iJRASET
International Journal For Research in
Applied Science and Engineering Technology

INTERNATIONAL JOURNAL
FOR RESEARCH
IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 5 Issue: XI Month of publication: November 2017

DOI:

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor:6.887
Volume 5 Issue XI November 2017- Available at www.ijraset.com

A Conceptual Review of Elastic Search – Survey Paper

Subhani shaik[1], Nalamothu Naga Malleswara Rao[2]

[1]Asst.professor, Department of CSE St. Mary's Group of Institutions Guntur, Chebrolu, Guntur, A.P, India
[2]Professor, Department of IT, RVR & JC College of Engineering, Chowdavaram, Guntur, A.P, India

Abstract: Elstic Search is a way to organize data and make it effortlessly accessible. In particular Elastic Search-a distributed full-text search engine-explicitly addresses issues of scalability, big data search and performance that relational databases were simply never designed to support. Elastic Search is a Server Based search developed in Java that is published as open source under the terms of the Apache Lucene. Elastic Search includes all advances in speed, security, scalability and hardware efficiency. It takes data and optimizes according to the language based searches and stores it in a sophisticated format. We plan to use Elastic Search's analytics tool to help improve the queried data. And also preprocess the query to make it faster before sending it to the server so as to reduce the time latency while being used over slower data connections. Whether it is searching a database of trade products by description or finding similar text in a body of crawled web pages, Elastic search is imaginary excellent. This paper presents the basics of what you need to know about Elastic search in order to use it.

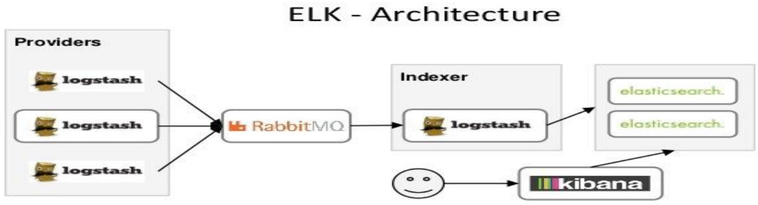
Keywords: Elastic search, ELK, Index, Shard

I. INTRODUCTION

A developer who uses relational databases, needed to search tables that had millions of records, resulting in exceedingly complex database views/stored-procedures and adding full text search on relational database fields. It made the database double the size and the speed was not optimum either. Relational databases are simply not built for such operations. With Elastic search we can achieve the speed we would like, as it leases us index millions of documents. Elastic search can perform queries across all those millions of documents and return accurate results in fraction of a second**.** Elastic search is a search engine based on Lucene. It provides a distributed, multitenant-capable full-text search engine with an HTTP web interface and schema-free JSON documents. Elastic search is developed in Java and is released as open source under the terms of the Apache Lucene. Elastic search is the most popular enterprise search engine followed by Apache Solr, also based on Lucene. Shay Banon formed the antecedent to Elastic search, named Compass, in 2004. While thinking about the third version of Compass he realized that it would be necessary to rewrite big parts of Compass to "create a scalable search solution". So he created "a solution built from the ground up to be distributed" and used a common interface, JSON over HTTP, suitable for programming languages other than Java as well. Shay Banon released the first version of Elastic search in February 2010. Elastic search BV was founded in 2012 to provide commercial services and products around Elastic search and related software.

A. What is ELK?

Elastic search s developed along with a data-collection and log-parsing engine titled Logstash, and an analytics and visualization platform titled Kibana. The three products are designed for use as an integrated solution, referred to as the "Elastic Stack" (formerly the "ELK stack"). The user interface to perform search and analytics is Kibana, an open source data visualization platform. With its instinctive, clean and responsive interface Kibana makes searching for data a joy. Logstash, another open source tool ensures the heavy lifting of consuming the logs from various systems and sends them to Elastic Search. Together, Elastic Search, LogStash and Kibana form the commonly known System ELK. Finally a powerful RESTFul API armors developers to do every action imaginable using the data set. Client libraries are also available for many languages



,

Figure 1.1: The ELK Architecture

II. ELASTIC SEARCH VS SOLR VS LUCENE

A. Prerequisites

In order to fully understand how Elastic Search works, especially when it comes to indexing and query processing, it is crucial to understand how Apache Lucene library works. Elastic search uses Lucene and tries to make all its features available through the JSON and Java API. It supports faceting and percolating, which can be useful for notifying if new documents match for registered queries. Elastic Search uses Lucene to handle document indexing. The same library is also used to perform search against the indexed documents.

Elastic search is a search engine based on Lucene. It provides a distributed, multitenant-capable full-text search engine with an HTTP web interface and schema-free JSON documents. Elastic search is developed in Java and is released as open source under the terms of the Apache License. Official clients are available in Java, .NET (C#), PHP, Python, Apache Groovy and many other languages. Elastic search is the most popular enterprise search engine followed by Apache Solr, also based on Lucene.

Solr is an source enterprise platform, written in Java, from the Apache Lucene project. Its major features include full-text search, hit highlighting, faceted search, real-time indexing, dynamic clustering, database integration, NoSQL features and rich document (e.g., Word, PDF) handling. Providing distributed search and index replication, Solr is designed for scalability and fault tolerance. Solr is widely used for enterprise search and analytics use cases and has an active development community and regular releases.

Solr's external configuration allows it to be tailored to many types of application without Java coding, and it has a plugin architecture to support more advanced customization. Apache Lucene is a source information software, originally written completely in Java by Doug Cutting. It is supported by the Apache Software Foundation and is released under the Apache Software License. Lucene has been ported to other programming languages including Object Pascal, Perl, C#, C++, Python, Ruby and PHP. Apache Lucene and Apache Solr are both produced by the same Apache Software Foundation development team since the two projects were merged in 2010. It is common to refer to the technology or products as Lucene/Solr or Solr/Lucene. Elastic Search and Solr have the same feature-set and address the same problem - that of building a fast, feature-rich search application on top of Apache Lucene.

B. Both provide search features such as

1) Java API and REST
2) faceting
3) highlighting
4) geo spatial search
5) replication

In fact, Solr and Elastic Search are so similar; there is even an ES plugin that allows you to use Solr clients/tools with Elastic Search.

For all practical purposes, there is no real reason to choose Solr over Elastic Search or vice versa. Both have mature codebases widespread deployment and are battle-proven. There are small disparities in the two, such as Elastic Search's percolation feature, and Solr's Pivot Facets.

III. ELASTIC SEARCH - BASIC CONCEPTIONS

A. The extensive features of Elastic search are as follows

1) Elastic Search is scalable up to megabytes of structured and unstructured data.
2) Elastic Search can be used as a standby of document stores like Mongo DB and Raven DB.
3) Elastic Search uses demoralization to improve the search performance.
4) Elastic Search is one of the popular enterprise search engines, which is currently being used by many big organizations like Wikipedia, The Guardian, Stack Overflow, Git Hub etc.

B. The basic concepts of what you need to know about Elastic search in order to use it are given below

1) *Indexing* –Elastic search is able to achieve fast search responses because, instead of searching the text directly, it searches an index instead. An Index is a collection of different type of documents and document properties. Index also uses the concept of shards to improve the performance. For example, a set of document contains data of a social networking application. This is like retrieving pages in a book related to a keyword by scanning the index at the back of a book, as opposed to searching every word of every page of the book. This type of index is called an inverted index, because it inverts a page-centric data structure (page->words) to a keyword-centric data structure (word->pages).Elastic search uses Apache Lucene to create and manage this inverted index.

2) *Document* − It is a collection of fields in a specific manner defined in JSON format. Every document belongs to a type and resides inside an index. Every document is associated with a unique identifier, called the UID. In Elastic search, a Document is the unit of search and index. An index consists of one or more Documents, and a Document consists of one or more Fields. In database terminology, a Document corresponds to a table row, and a Field corresponds to a table column.
3) *Schema* - Unlike Solr, Elastic search is schema-free. Whilst you are not required to specify a schema before indexing documents, it is necessary to add mapping declarations if you require anything but the most basic fields and operations.

C. The schema declare

1) what fields there are
2) which field should be used as the unique/primary key
3) which fields are required
4) how to index and search each field

a) *Type/Mapping:* It is a collection of documents sharing a set of common fields present in the same index. For example, an Index contains data of a social networking application, and then there can be a specific type for user profile data, another type for messaging data and another for comments data. In Elastic search, an index may store documents of different "mapping types". You can associate multiple mapping definitions for each mapping type. A mapping type is a way of separating the documents in an index into logical groups. To create a mapping, you will need the Put Mapping API, or you can add multiple mappings when you create an index.
b) *Node: It* refers to a single running instance of Elastic search. Single physical and virtual server accommodates multiple nodes depending upon the capabilities of their physical resources like RAM, storage and processing power.
c) *Cluster:* It is a collection of one or more nodes. Cluster provides collective indexing and search capabilities across all the nodes for entire data
d) *Shard:* Indexes are horizontally subdivided into shards. This means each shard contains all the properties of document, but contains less number of JSON objects than index. The horizontal separation makes shard an independent node, which can be store in any node. Primary shard is the original horizontal part of an index and then these primary shards are replicated into replica shards.
e) *Replicas:* Elastic Search allows a user to create replicas of their indexes and shards. Replication not only helps in increasing the availability of data in case of failure, but also improves the performance of searching by carrying out a parallel search operation in these replicas.

IV. ELASTIC SEARCH - WORKING MODEL

A. Processing Nodes

Once Elastic search node starts, it uses the discovery module to find the other nodes on the same cluster and connect to them. The master node reads the cluster state and goes into the recovery process. During this state, it checks which shards are available and decides which shards will be the primary shards. After this the whole cluster enters into a yellow state. This means that a cluster is able to run queries, but full throughput and all possibilities are not achieved yet. The next thing to do is to find duplicated shards and treat them as replicas. When a shard has too few replicas, the master node decides where to put missing shards and additional replicas are created based on a primary shard.

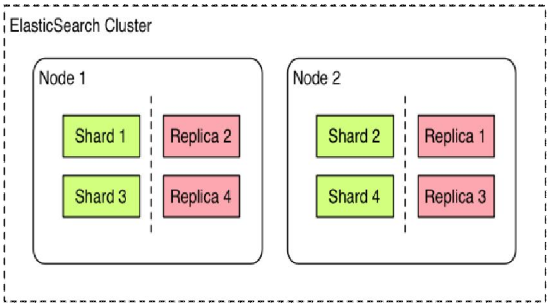


Figure 4.1:The Startup Process of Elastic Search

B. Indexing Data

There are many ways to send data to Elastic search. Index API is one of the easiest way. Rest API is another interesting way. Dot Net Client API libraries are also available which can be refered by a DOT NET application to read write data to Elastic search cluster

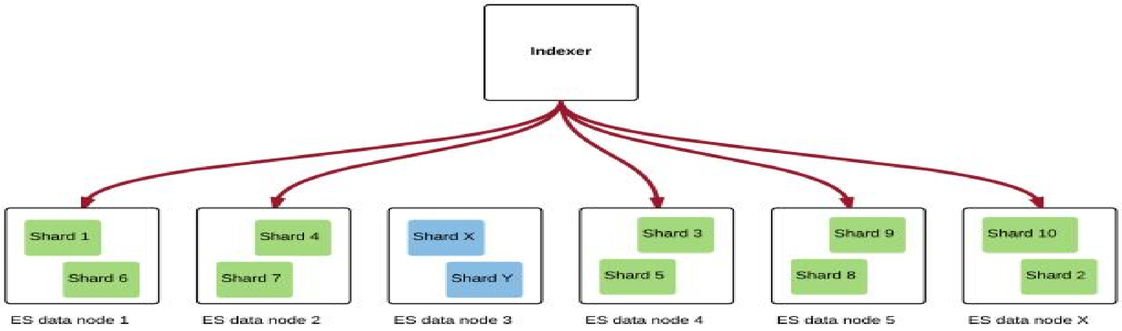


Figure 4.2:The Indexing of Elastic Search

C. Querying Data

The Query API is an important section of Elastic search API. The Query process is divided into two phase: the scatter phase is about querying all the relevant shards of the index and the gather phase is about gathering the results from the relevant shards, combining them, sorting, processing and returning to the client.

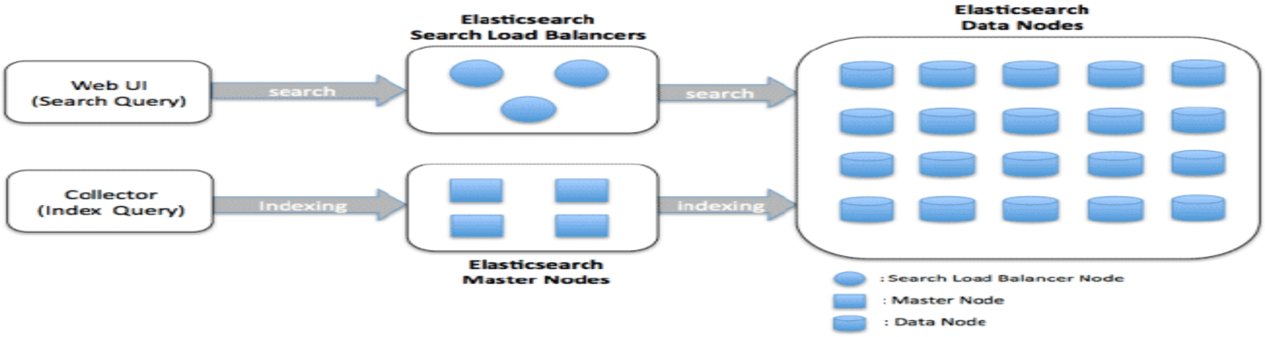


Figure 4.3: Querying of Data in Elastic Search

1) *Query DSL:* The query DSL is a flexible, expressive search language that Elastic search uses to expose most of the power of Lucene through a simple JSON interface. It makes your queries more flexible, more precise, easier to read, and easier to debug. The Query DSL is Elastic search's way of making Lucene's query syntax accessible to users, allowing complex queries to be composed using a JSON syntax. Like Lucene, there are basic queries such as term or prefix queries and also compound queries like the bool query.

A query clause typically has this structure:

```
{
  QUERY_NAME: {
    ARGUMENT: VALUE,
    ARGUMENT: VALUE,
  }
}
```

If it references one particular field, it has this structure:

```
{
  QUERY_NAME: {
    FIELD_NAME: {
      ARGUMENT: VALUE,
            ARGUMENT: VALUE,
         }
      }
}
```

In Elastic Search, searching is carried out by using query based on JSON. Query is made up of two clauses

2) *Leaf Query Clauses:* These clauses are match, term or range, which look for a specific value in specific field.
3) *Compound Query Clauses:* These queries are a combination of leaf query clauses and other compound queries to extract the desired information. Elastic Search supports a large number of queries. A query starts with a query key word and then has conditions and filters inside in the form of JSON object.

D. Elastic Search - Architecture

Elastic Search is a robust search and analytics tool that stores data in a document oriented data store. It is open source, meaning you can download, use and modify the program free of charge. The most popular usage of Elastic Search today is Log Management. Similar products in the market today are Splunk and Solr. Elastic Search is built on top of high performance open source search engine Apache Lucene. The documented oriented storage differs sharply from traditional table oriented RDBMS (Such as Oracle, MS SQL Server). With document oriented data storage, data is stored as structured JSON (JavaScript Object Notation) documents. Every field is indexed by default. This is why the search speed is incredible. The architecture of Elastic Search favors distribution, meaning you can scale your Elastic Search infrastructure massively and seamlessly.

Elastic search can be used to search all kinds of documents. It provides scalable search, has near real-time search, and supports multitenancy. Elastic search is distributed, which means that indices can be divided into shards and each shard can have zero or more replicas. Each node hosts one or more shards, and acts as a coordinator to delegate operations to the correct shard(s). Rebalancing and routing are done automatically". Related data is often stored in the same index, which consists of one or more primary shards, and zero or more replica shards. Once an index has been created, the number of primary shards cannot be changed.

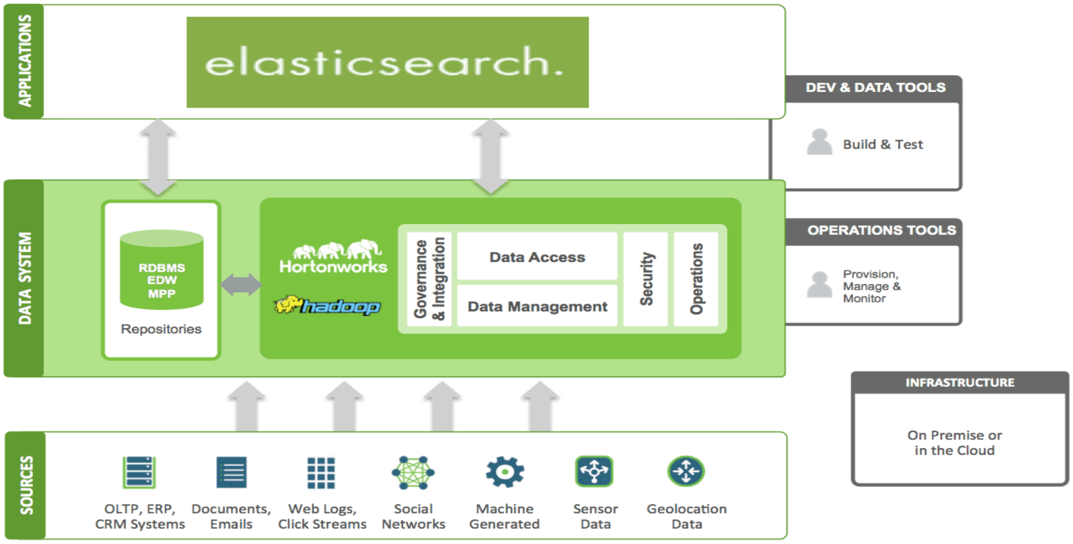


Figure 4.1: The Architecture of Elastic Search

Elastic search does not have tables, and a schema is not required. Elastic search stores data documents that consist of JSON strings inside an index. The field is like the columns of the SQL database and the value represents the data in the row cells.

Elastic search tries hard to hide the complexity of distributed systems. Here are some of the operations happening automatically under the hood

1) Separating documents into different shards, which can be stored on a single node or on multiple nodes.
2) Balancing shards across the nodes in the cluster to spread the indexing and search load.
3) Replicating each shard to provide redundant copies of the data, to prevent data loss in case of hardware failure.
4) Routing requests from any node in the cluster to the nodes that hold the concerned data
5) Seamlessly integrating new nodes as your cluster grows or redistributing shards to recover from node loss

V. ELASTIC SEARCH - USES AND ADVANTAGES AND DISADVANTAGES

A. Elastic Search - Uses

Elastic search can be used as a blog storage engine. Traditional SQL doesn’t readily give you the means to do that. Most software generates tons of data that is worth analyzing, Elastic search comes with Logstash and Kibanato give you a full analytics system. Finally, Elastic search can be used as Data ware house, where you have documents with many different attributes and non-predictable schemas. Since Elastic search is schema less, it won’t matter that you store various documents there, you will still be able to search them easily and quickly

B. Elastic Search - Advantages

1) Elastic Search is developed on Java, which makes it compatible on almost every platform.
2) Elastic Search is real time, in other words after one second the added document is searchable in this engine.
3) Elastic Search is distributed, which makes it easy to scale and integrate in any big organization. Creating full backups are easy by using the concept of gateway, which is present in Elastic Search.
4) Handling multi-tenancy is very easy in Elastic Search when compared to Apache Solr.
5) Elastic Search uses JSON objects as responses, which makes it possible to invoke the Elastic Search server with a large number of different programming languages.
6) Elastic Search supports almost every document type except those that do not support text rendering.

C. Elastic Search – Disadvantages

1) Elastic Search does not have multi-language support in terms of handling request and response data (only possible in JSON) unlike in Apache Solr, where it is possible in CSV, XML and JSON formats.
2) Elastic Search also has a problem of Split brain situations, but in rare cases.

VI. AMAZON ELASTIC SEARCH

Amazon Elastic Search Service makes it easy to deploy, secure, operate, and scale Elastic Search for log analytics, full text search, application monitoring, and more. Amazon Elastic Search Service is a fully managed service that delivers Elastic Search’s easy-to-use APIs and real-time analytics capabilities alongside the availability, scalability, and security that production workloads require. The service offers built-in integrations with Kibana, Logstash, and AWS services including Amazon Virtual Private Cloud (VPC), Amazon Kinesis Firehose, AWS Lambda, and Amazon Cloud Watch so that you can go from raw data to actionable insights quickly and securely. Amazon Elastic Search Service supplies all the resources for your domain and launches it. You can securely access the domain from your VPC or from a public endpoint. The service automatically detects and replaces failed Elastic Search nodes, reducing the overhead associated with self-managed infrastructure and the Elastic Search software. Amazon Elastic Search Service allows you to easily scale your cluster via a single API call or a few clicks in the console. With Amazon Elastic Search Service, you get direct access to the Elastic Search open-source API so the code and applications you’re already using with your existing Elastic Search environments work seamlessly.

A. Benefits

1) Easy to Use: Using Amazon Elastic Search Service you can deploy a production-ready Elastic Search cluster in minutes without having to worry about provisioning infrastructure or installing and maintaining Elastic Search software. It simplifies time-consuming management tasks such as software patching, failure recovery, backups, and monitoring
2) Supports Open-Source APIs and Tools: Amazon Elastic Search Service gives you direct access to the Elastic Search open-source API. Amazon Elastic Search Service supports Logstash, open-source data ingestion, transformation, and loading tool; and Kibana, an open-source visualization tool
3) Secure: Amazon Elastic Search Service can be accessed securely from VPC to keep all traffic between your VPC and Amazon Elastic Search Service within the AWS network. You can control access to your Amazon Elastic Search Service domain using security groups and AWS Identity and Access Management (IAM) policies. Amazon Elastic Search Service routinely applies security patches and keeps your domain up-to-date.
4) Highly Available; Amazon Elastic Search Service is designed to be highly available using zone awareness, which replicates data between two Availability Zones in the same region. Amazon Elastic Search Service monitors the health of your clusters and automatically replaces failed nodes

5) *Tightly Integrated with Other AWS Services:* Amazon Elastic Search Service offers built-in integrations with other AWS services such as Kinesis Firehose, AWS IoT, and Amazon Cloud Watch Logs for seamless data ingestion;
6) *Easily Scalable:* Amazon Elastic Search Service enables to monitor cluster through Amazon Cloud Watch metrics and resize the cluster up or down via a single API call or a few clicks in the AWS Management Console. You can configure your cluster to meet your performance requirements by selecting from a range of instance types and storage options including SSD-powered EBS volumes.

B. Use Cases

1) *Log Analytics:* Analyze un-structured and semi-structured logs generated by websites, mobile devices, servers, sensors, and more for a wide variety of applications such as digital marketing, application monitoring, fraud detection, ad tech, gaming, and IoT. Capture, pre-process, and load log data into Amazon Elastic Search Service using Amazon Kinesis Firehose, Logstash, or Amazon Cloud Watch Logs. You can then search, explore, and visualize the data using Kibana and the Elastic Search query DSL to gain valuable insights about your users and applications
2) *Full Text Search:* Amazon Elastic Search Service supports faceting, which allows customers to narrow their search results by value ranges for fields like price, product characteristics, and brands;
3) *Distributed Document Store:* Amazon Elastic Search Service provides a simple REST API, fast performance, powerful search capabilities and seamless scalability, to build highly perform ant applications that can store and retrieve billions of documents, with integrated replication across Availability Zones within a region
4) *Real-time Application Monitoring :* Capture activity logs across customer-facing applications and websites. Use Logstash to push these logs to Amazon Elastic Search Service domain. Elastic Search indexes the data and makes it available for analysis in near real-time (less than one second). By using built-in Kibana plugin we can visualize the data and perform operational analyses like identifying outages and problems. With Elastic Search's geospatial analysis, we can identify the geographical region where the problem is occurring. Troubleshooting teams can then search the index and perform statistical aggregations to identify root cause and fix issues.
5) *Clickstream Analytics:* Deliver rel-time metrics on digital content and enable authors and marketers to connect with their customers in the most effective way. Stream billions of small messages that are compressed, batched, and delivered to Amazon Elastic Search Service using Amazon Kinesis Firehose. Once the data is loaded to Amazon Elastic Search Service, you can aggregate, filter, and process the data, and refresh content performance dashboards in near real-time. For example, Hearst Corporation built a clickstream analytics platform using Amazon Elastic Search Service, Amazon Kinesis Streams, and Amazon Kinesis Firehose to transmit and process 30 terabytes of data a day from 300+ Hearst websites worldwide.

VII. CONCLUSION

Elastic search can scale out to hundreds or even thousands of servers and handle megabytes of data. Elastic-search is distributed by nature, and it is designed to hide the complexity that comes with being distributed. The distributed aspect of Elastic search is largely transparent. Elastic search is an open-source, broadly-distributable, readily-scalable, enterprise-grade search engine. Accessible through an extensive and elaborate API, Elastic search can power extremely fast searches that support data discovery applications. Elastic Search uses standard RESTful APIs and JSON. We also build and maintain clients in many languages such as Java, Python, .NET, and Groovy. Conventional SQL database managements systems aren't really designed for full-text searches, and they certainly don't perform well against loosely structured raw data that resides outside the database. On the same hardware, queries that would take more than 10 seconds using SQL will return results in under 10 milliseconds in Elastic search.

REFERENCES

[1]AmazonWebService.[https://aws.amazon.com/elasticsearch-service/
[2]ElasticSearchTutorial[http://www.elasticsearchtutorial.com/
[3]DataBigZone[https://dzone.com/articles/elasticsearch-getting-started]
[4]https://www.elastic.co/products/elasticsearch
[5]https://www.tutorialspoint.com/elasticsearch/elasticsearch_query_dsl.htm
[6] https://en.wikipedia.org/wiki/Elasticsearch
[7] https://en.wikipedia.org/wiki/Apache_Lucene
[8] https://en.wikipedia.org/wiki/Apache_Solr
[9] https://qbox.io/blog/what-is-elasticsearch
[10] Survey Paper on Elastic Search: https://www.ijsr.net/archive/v5i1/NOV152583.

ABOUT THE AUTHORS

Mr.Subhani shaik is working as Assistant professor in Department of computer science and Engineering at St. Mary’s group of institutions Guntur, he has 12 years of Teaching Experience in the academics.

Dr. Nalamothu Naga Malleswara Rao is working as Professor in the Department of Information Technology at RVR & JC College of Engineering with 25 years of Teaching Experience in the academics.