ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 4, Issue 3, pp: (23-26), Month: July - September 2016, Available at: www.researchpublish.com

ELASTIC SEARCH AS MIDDLEWARE SEARCH ENGINE FOR DASHBOARDS

Dashrath Mane[1], Rajinder Singh Pabla[2]

VESIT MCA Department, Chembur, Mumbai.

Abstract: **Organizations are generating, processing, and retaining data at a rate that often exceeds their ability to analyse it effectively. The outcomes and results derived from these large data sets helps in decision making and are often key to the success of the organizations. This helps in understand how to solve hard problems and thus gain competitive advantage. It has started being impractical to analyse using traditional offline, read-only relational databases because the data is so voluminous and increasing day by day. New "big data" technologies and architectures, like Hadoop and NoSQL databases, have been introduced to better support the needs of organizations analysing such data. In particular, Elastic search is a way to organize data and make it easily accessible. It is a server based search on Lucene. It is 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. In this paper, I reflect upon my own experience with Elastic search and how it can help organizations to manage their highly voluminous data.**

Keywords: **Elastic search, Dashboard, TFS (Team Foundation Server), Elastic search index.**

I. INTRODUCTION

Elastic search's search engine is basically based on another software project called Lucene. To understand Elastic search we can say it as a part of infrastructure built nearby Lucene's Java libraries. In Elastic search everything is related to storing optimized indexes of query terms is executed by Lucene and the actual algorithms for matching text. The API provided by Elastic search is highly scalability, and operational tools overhead Lucene's search implementation. Elastic search server is installation is quite simple, and the default configuration is sufficient for a standalone use without tweaking, although most users will eventually want to fine tune some of the parameters. A running instance of the Elastic search server is called a node, and two or more nodes can form the Elastic search cluster. To set up an Elastic search cluster, the only value that needs to be set in the configuration file is the name of a cluster; Elastic search will take care of discovering nodes on the network and binding them into a cluster.

II. CONCEPTS OF ELASTIC SEARCH

There are few concepts in Elastic search which are important and critical to fully understand how Elastic search works and operates.

1. Custer

A cluster is a set of one or more nodes (known as servers) that together holds your entire data and provides federated indexing and search capabilities across all nodes. A cluster is identified by a default unique name which is "Elastic search". This name is important because a node can only be part of a cluster if the node is set up to join the cluster by its name.

2. Node

A node is a single server that is part of your cluster, participates in the cluster's indexing and search capabilities and stores your data. Just like a cluster, a node is identified by a name which by default is a random Marvel character name that is assigned to the node at start up.

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 4, Issue 3, pp: (23-26), Month: July - September 2016, Available at: www.researchpublish.com

3. Index

An index is a collection of documents that have somewhat similar characteristics. Elastic search stores its data in one or more indices. Comparing Elastic search with SQL world, indexing is similar to a database. It is used to store the documents and read them from it.

4. Document

Document is the main entity in the Elastic search world. A document is a basic unit of information that can be indexed at the end. Document consists of fields, which are identified by its unique name and can contain one or multiple values. Each document may have different set of fields

5. Type

Within an index, you can define one or more types. This allows us to store various document types in one index and different mapping for different documents types.

6. Mapping

All documents are analyzed before being indexed. The input text is divided into tokens, which tokens should be filtered out, or what additional processing, such as removing HTML tags, is needed. This is where mapping comes into play; it holds all the information about the analysis chain.

7. Shard

Clustering allows storing information volumes that exceed abilities of a single server. To achieve this requirement, Elastic search spreads data to several physical Lucene indices these indices are called Shards, and all the parts of the index is called sharding. Elastic search can do this automatically and all the parts of the index (shards) are visible to the user as one big index.

8. Replica

Sharding allows pushing more data into Elastic search that is possible for a single node to handle. The idea is simple create an additional copy of shard, which can be used for queries just as original, primary shard.

III. WORKING OF ELASTIC SEARCH

1. About 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, if necessary, 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.

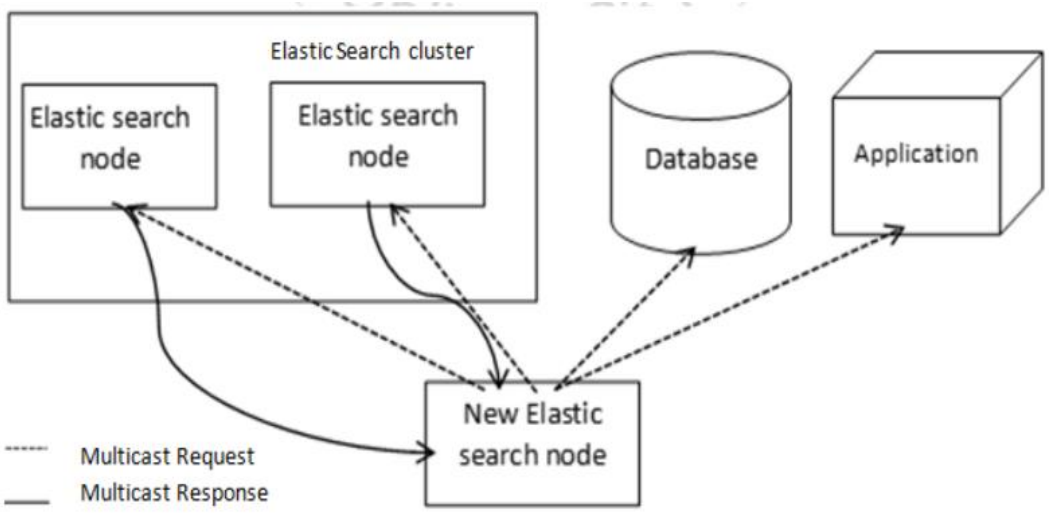

Figure 1: The Startup Process

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 4, Issue 3, pp: (23-26), Month: July - September 2016, Available at: www.researchpublish.com

2. 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

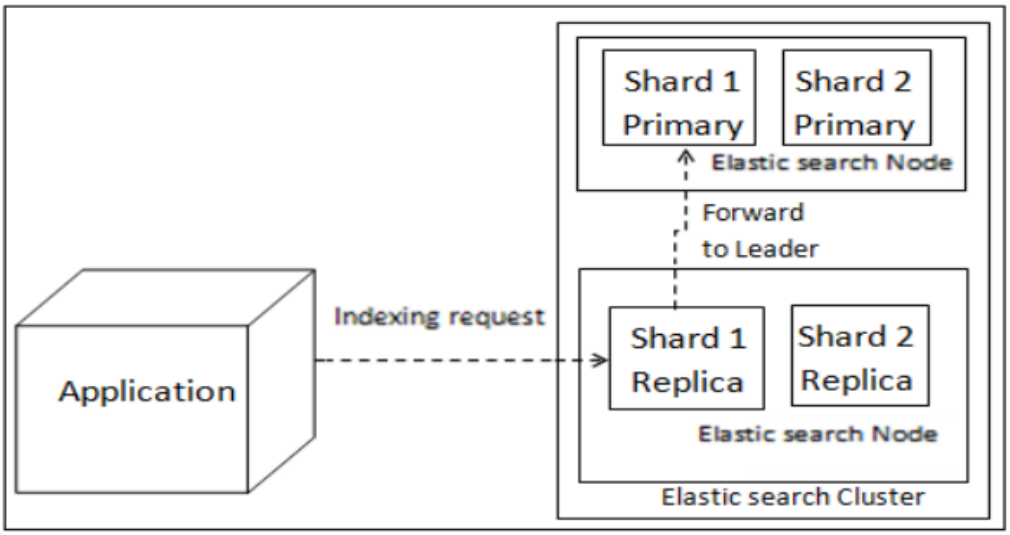

Figure 2: The Indexing Process

3. 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.

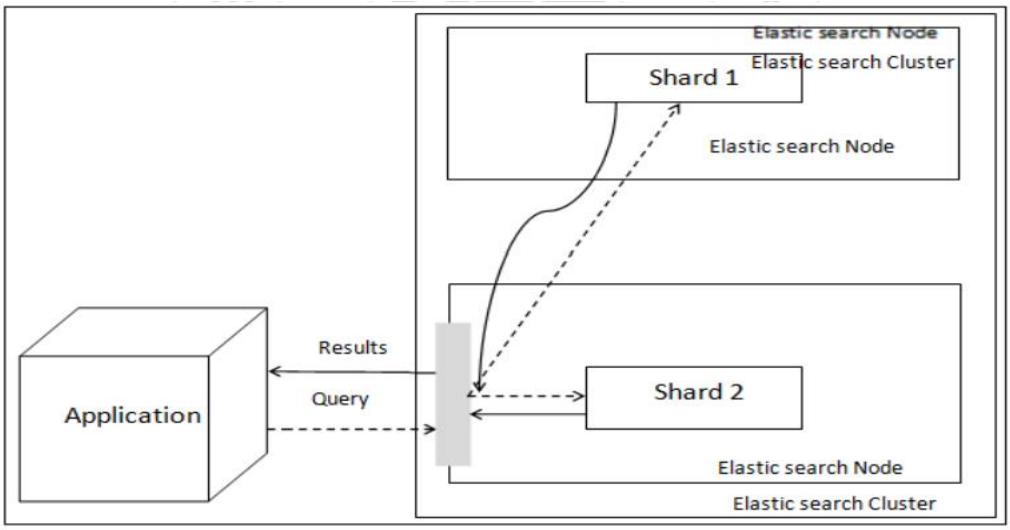

Figure 3: The Querying Process

IV. PROPOSED SYSTEM

The proposed system basically initiates the usage of Elastic search as a middleware search engine which is populated with data from TFS-Team Foundation server (a software configuration management tool by Microsoft). TFS is used in IT organizations to store data related to projects. It consists of the details like team members, members capacity etc. for any given project. This data can be fetched and inserted in Elastic search and can be used for analysing the performance of the team and state of tge projects. This data can also be used for creating reports and charts helping the managers in decision making and understanding the progress of the projects.

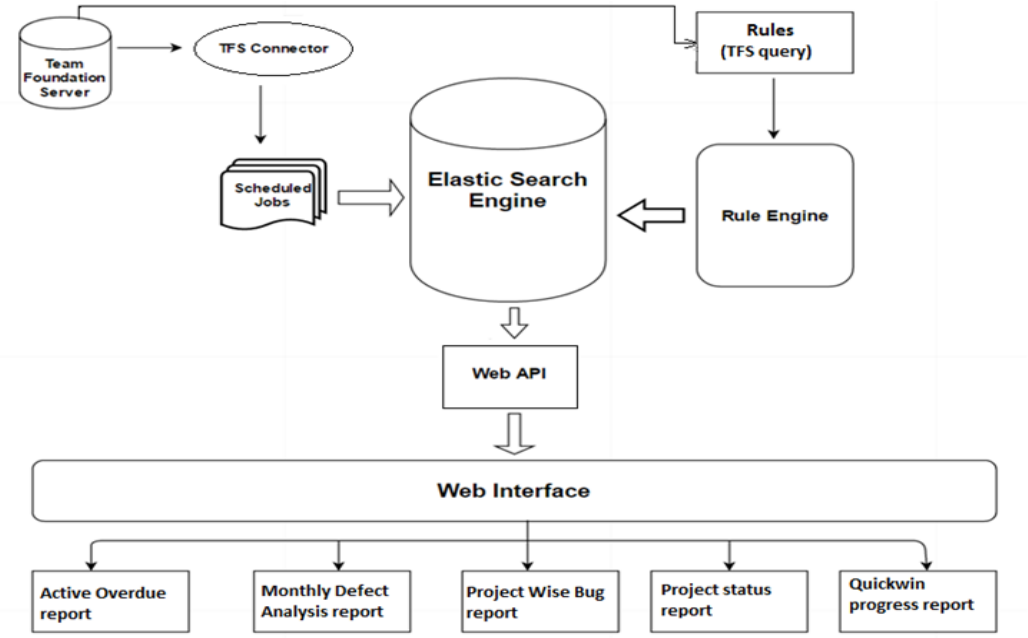

Figure 4: Architecture Diagram

Figure 4 is a highlevel architecture of the proposed system. Based on the rules defined in the TFS, projects data is fetched from TFS using TFS connector and inserted into Elastic search index. Now data from the Elastic search is retrived using web API and desplayed as reports, graphs and charts using Web Interface. These reports

V. ACKNOLEDGEMENT

I would like to thank my supervisor Mr. Dashrath Mane for his continuous support in my project and his willingness to bring his breadth of experience to this project. I would like to thank my lecturers and colleagues in VESIT, MCA program for the great learning experiences and interaction we shared which helped me in my project to a great extent.

VI. CONCLUSION

As the projects are incresing the managers wants to be updated with the state and progress of the projects. The essence of this proposed system is to provide an at a glance view of all the pojects and their status, information about team enegagement and will help the managers in planing and decision making. Ealsticsearch is an essential part of the proposed system as it acts as a data store for the application.

REFERENCES

[1] Sematext. Elastic search refresh interval vs indexing performance. http://bit.ly/1iZoPGc, July 2013.

[2] Mining Modern Repositories with Elastic search: https://cs.uwaterloo.ca/~okononen/msr2014.pdf

[3] Use of Elastic Search for Intelligent Algorithms to Ease the Healthcare Industry:http://www.ijsce.org/attachments/File/v3i6/F2013013614.pdf

[4] Survey Paper on Elastic Search: https://www.ijsr.net/archive/v5i1/NOV152583.pdf