International Journal of Engineering Research and General Science Volume 3, Issue 2, Part 2, March-April, 2015
ISSN 2091-2730

Design and Implementation of Rapid Searching Framework for Relational Database

Ms Sonam Borhade[1], Prof. Pankaj Agarkar[2]

1. PG student of Computer Engineering, Dr. D.Y Patil School of Engineering Lohegaon, Pune, India,

borhade.sonam @gmail.com, 8600646751

2. Prof. Pankaj Agarkar, Dr. DYP School of Engineering Lohegaon, Pune, India, pmagarkar@gmail.com.

Abstract— Information explosion opens a challenge for researcher in information retrieval and search. As emergence of digital repositories and information explosion there is clear need of technique to organize information. This need day by day attracts researchers to give better techniques in area of information retrieval and search for large database. Organizations depend on relational database for transaction processing and they prefer selection operation to process in large database. Therefore, when data goes beyond few million records selection operation take lot of time to process such database. In this operation, indexes build on columns and if there are number of tables then it require more time to process whole transaction and result in performance degradation. Searching framework must be nice choice for this kind of transaction processing.

Apache LUCENE is nowadays most widely used popular searching framework found in many applications. Toward this end, paper addresses shortcomings of selection operation: the clear need to faster record fetching form big database and offload selection work and with that, it suggests one alternative as LUCENE Indexing. Here we implemented fast searching framework with help of lucene for relational database. Records searched in lucene and then used in mysql for further transaction. Performance of both mysql search and lucene search get evaluated.

Keywords— Performance Evaluation, Information Retrieval and Search, Lucene Indexing.

INTRODUCTION

Nowadays emergence of digital libraries and information exchange in information technology area there is clear need for improved techniques to organize large quantities of information. Organizations mostly use relational database for transaction processing, which use selection operation for searching records. However, if data goes beyond few million records selection operation take more time to process such data to retrieve records from large dataset and put load on database. To do good performance it requires more number of servers on price of increased licensing cost. Searching framework is one nice choice to search records in such case. To offload selection work from database and to achieve faster record retrieval experience there is need of better searching framework. As relational database require in organizations for transaction processing, cannot replace it with searching framework. So here clear need of an external searching framework, which helps to achieve better performance on selection and sequential search for relational database. Apache LUCENE is nowadays most widely used popular searching framework used in many applications. This paper focus on LUCENE technology which well known for its searching capabilities. Records searched in LUCENE and then used in database.

Main objective of this work is to study LUCENE technology for indexing data and using this technology to implement searching and indexing of 1 million files from raw content set. Resulting index optimized for searching. Raw content set is a collection of user's information. Purpose of this work is to allow users a convenient full text search method and evaluate performance of relational database and LUCENE indexing to analyze which give best results. As discussion focused on LUCENE technology, some question has come in to mind that what is actually LUCENE? Why choose LUCENE as solution? Moreover, how it is suitable for our need? Therefore, reminder discussion gives answer of these questions.

A. What is LUCENE?

LUCENE originally written by Dough Cutting and is now licensed under Apache software license [3].A common misconception with LUCENE is, it is ready to use search application. In fact, LUCENE is an information retrieval library entirely written in java a free open source project. LUCENE is scalable and high performance IR library for full text indexing and searching and can add easily to any application. We can build application on top of LUCENE [3]. LUCENE having several built in analyzers, which handle spell correction, compound words, case sensitivity. For fast record retrieval it first index available data then apply search on index. . LUCENE is format independent so it can index and search files available in any format. For indexing it, extract text from data available in any format. To achieve better search results parsing and analysis play a supplementary role with indexing and searching [3,4]. LUCENE stores indexing internally in documents form, documents contain fields internally and every field have field name. Every field name having value associated with it [3]. Document contains multiple fields. To search, search on field of documents. Therefore, when we add database entries, each row treated as document and document as fields. Every document stored by unique document id. For searching in LUCENE document Id is required. LUCENE provides only document IDs for searching. Document Id

also used to retrieve document. The document IDs are unique but not permanent. We can add documents and delete documents entries. Therefore, IDs for deleted documents do remain unused. LUCENE periodically performs compression of index. In compression, LUCENE collects document IDs of deleted documents for reuse, where document IDs change, that is why one cannot depend on LUCENE document Ids.

B. Why Choose LUCENE?

Rather than LUCENE, there are technologies available like Xpian, which is search engine library written in C++, Minion is a search engine written in java from Sun Labs, Egothor is much similar to LUCENE, is java library for full text indexing and searching. However, all of them are not suitable for this proposed work and not able to customize and build in existing application. In addition, solr from apache LUCENE project is open source search platform for enterprise. This is for web based project and based on LUCENE for indexing. Solr can be nice choice in future cause easily migrates from LUCENE to solr. LUCENE is promising choice for this proposed work because apache developer's community writes it entirely in java that used widely and supported therefore we are using LUCENE.

C. How LUCENE suitable for proposed work?

As discussed in above lines about what is LUCENE we found answer of our third question is that searching is where LUCENE is strong. Therefore, to build external searching framework it is better suitable for our need. We are using Apache LUCENE framework to provide searching facility [1]. We are trying to evaluate the framework in which the records searched in LUCENE [2], and then they used in database operations. Performance of both LUCENE indexing and database operation evaluated for performance. Therefore, LUCENE is better suited for our need. In this paper, searching module used as plugin to evaluate performance. In addition, we can use this in existing projects.

RELATED WORK

We surveyed some real world search applications mentioned below which switched toward lucene to achieve better searching and working superb until now.

A. *NetFlix* [3]

NetFlix is one of the open source search application designed by Reed Hastings. This application offer movies on very less rent and without late fee. Idea of NetFlix was born because of video stores asking for too much rent and late fee. NetFlix powered by solr, which is LUCENE search server. Fuzzy search can add in LUCENE [4], in NetFlix fuzzy search added to LUCENE. LUCENE provide auto-completion feature in NetFlix for movie names as well as spelling correction of misspelled actor names. When user enters query NetFlix suggests movie titles, which are relevant to query. For this application, LUCENE regularly index and search millions of movie records in sub seconds and free from licensing cost.

B. *Monster* [3]

Monster is very well known job and carrier search engine used worldwide. Monster posts number of jobs any one time. Up to 2008 monster having 150 million resumes in its database and per month, 63 million job seekers served. Today, approximately 300 to 400 queries have been running on 40 milliseconds of average response time. To provide better service to both customers and employees and to stay in market monster have requirements like: High volume of data management, Need to maintain constant updates of inventory, Need to provide faster results on users query, Refine search for end users without any performance degradation ,Overcome technology barriers which limit information scope, Security concerns to maintain users privacy. Integrate scalable and flexible approach. To complete such increasing need monster uses LUCENE is as solution. LUCENE makes clusters of high volumes data to reduce size of index. For faster results of query LUCENE, use real time indexing here for freshers. To enable refined search LUCENE gives faceted search and 'single click' filters. LUCENE having intuitive search capabilities that helps for deeper searching of resumes and jobs browsing. In addition, it provides security controls for users information management and having unlimited scalability and customization based on open source software.

C. *LinkedIn* [3]

Linked in have challenges: To manage varying database where every second new users join and add their profiles, Achieve real time indexing of unstructured data, Providing immediate response to query in crucial hours, Giving linguistic support and institutive browsing, easy integration to different platforms like web 2.0 tools to make user profiles which takes data form number of sources. Therefore, to overcome such challenges LinkedIn chose LUCENE as solution to implement search. LUCENE provides index segmentation through which large data managed by limiting index base and achieve faster response for query. Here LUCENE provides faceted search through some advanced features like auto completion of contacts and better view of search. LUCENE give relevant records that are helpful here which sort between users profile and profiles, which does not match to that profiles and give result in sub seconds. LUCENE provide solution by integrating with recent web tools, which gives flexibility; for example can add videos in search results.

D. *U.S Food and Drug Administration* [3]

Food and Drug Administration (FDA) is one agency in Health and Human Services department under U.S government. This agency control and inspect safety of medical foods, veterinarian products, cosmetics and tobacco. FDA has big library of information in which many records available from optical character recognition to recent digital formats. So to manage such kind of information, FDA switched to semantic mining framework using LUCENE. Major requirement of FDA was to integrate petabytes of data available in different intranet of enterprises. Therefore, in such situation LUCENE is providing way to search across different enterprise repositories and detect data duplication. Here LUCENE's Content Analysis and query analysis algorithm help fast and relevant record retrieval. Filter in LUCENE manage security and access control policies. By using Lucne integration is possible with existing enterprise infrastructure to reduce TCO.

E. *MUFIN* [5]

In 2011, research accomplished to integrate LUCENE search in MUFIN [5]. The primary purpose was to study LUCENE technology and implement content-based image retrieval for indexing large data. MUFIN is existing Multi feature indexing network has basic requirement to organize available 100 million collection of image based on CoPhIR data set. Therefore, by using LUCENE, user interface built for users and integrated into existing MUFIN website.

There are many indexing types available in LUCENE [6] basic types are NIO FS Directory, Simple FS Directory, and RAM Directory. Indexes stored in actual system through Simple FS Directory. This is useful for large indices. NIO FS Directory stores index to hard disk through JAVA IO API. RAM Directory is way to store all indexes in RAM. This is suitable for smaller indices that can be fully loaded in memory and destroyed after termination of application. As the index held in memory, it is comparatively faster [7]. Search results in LUCENE shown in relevant manner and first it gives relevant results. In this present work for transaction processing all records are required. Therefore, we do not need relevance. For multiple fields search, LUCENE search queries used. Research in [8] gives new approach to access information, which solves problem where user required to submit whole query and to find information from results users always face try and see approach. Author proposes interactive fuzzy search through which system able to search on the fly as user types keywords of query. Small input errors neglected.

PROGRAMMER'S DESIGN

A. Schematic Representation of Proposed Work

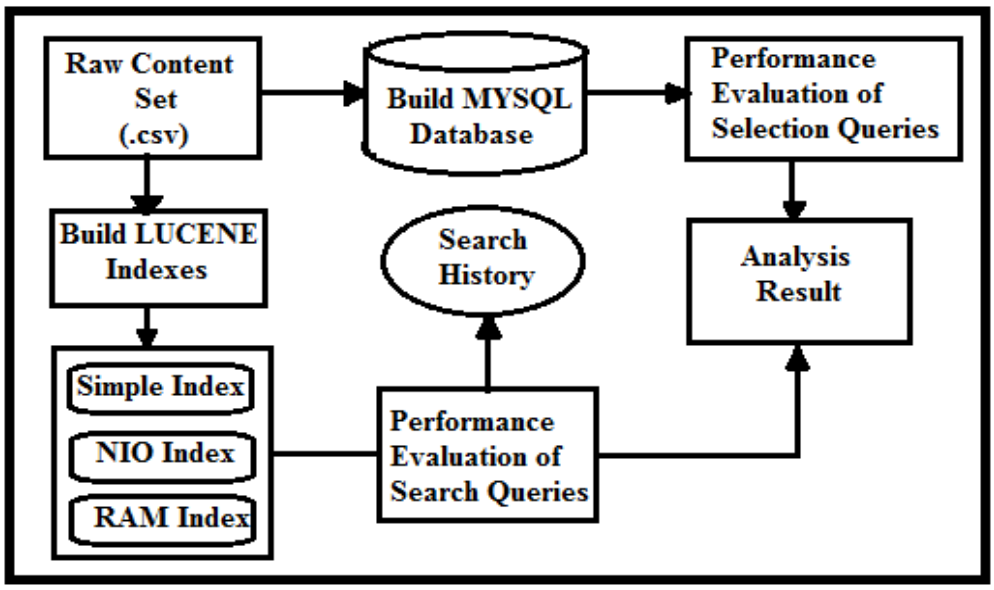

Figure1. Evaluation Workflow

In this proposed work, our focus on performance evaluation of LUCENE indexes and MYSQL database for record fetching experience. In addition, here we are trying to give better alternative for selection operation held in large database to search records faster. Therefore, indexing from apache LUCENE used, with that we are building three LUCENE indexes and will test them with different records to check which suitable for our purpose. At the same time, we are also building MYSQL database and will test it. Searching Module will apply on both database and LUCENE Indexes and search history will maintain. For outcome, performance of both LUCENE indexes and MYSQL database will compare and generate results.

Architecture of proposed work consists four major phases in first phase creation of MYSQL database and LUCENE indexes from available raw content. Then search for records on LUCENE indexes and MYSQL database. Then performance evaluation will be on LUCENE indexes and MYSQL database. In Fourth phase, all the outcomes of MYSQL database and LUCENE indexes analyze for results.

B. Mathematical Model

System:
S=System
S=I, O, BD, BLI, PED, PEL, R
Input to System:
I=RF
RF=Raw File Input.
Output:
O=DB, SI, NI, RI, R
DB=Relational Database
SI=Simple File System Directory Index.
NI=NIO File System Directory Index.
RI=RAM Directory Index.
R=Analysis Result.
Functions:
BD (RF) RF -> DB
Build MYSQL database from raw data.
BL (RF) RF -> SI, NI, RI
Create LUCENE Indexes from raw data.
PED (DB, Q) DB, Q -> Td
Achieve query time (performance measure) for selection queries from MYSQL database.
PEL (SI, NI, RI, Q) SI, NI, RI, Q -> Tl
Achieve performance measure in LUCENE for distinct indexes.
A (Td, Tl) Td, Tl -> R
Make Analysis Result from two performance measures.

We are evaluating the performance of record retrieval from apache LUCENE indexes with relational database. Using LUCENE, one can search records faster so, can use in database for different transactions. The search history maintenance is implemented which can be used by anyone who wants to maintain user search history. Per user, search history maintainer will allow to show search history of particular user. This will enhance the LUCENE searching feature. This system evaluates the use of LUCENE searching framework to perform data searching and retrieval from big databases. The searching performance evaluation will held on MYSQL database and different LUCENE indexes. Then comparison and analysis will held to see that which arrangement gives best performance. We are using LUCENE indexes Ram Index, Simple Index, and NIO Index. Then we will check performance of these three indexes with different records to analyze, which is best.

Exact match keywords displayed by Lucene scoring formula as below:

$$\sum t \text{ in } q \; (t\,f\,(t \text{ in } d\,)\; idf\,(t)\; boost(t.field \text{ in } d\,)\; lengthNorm(t.field \text{ in } d\,))\; coord(q,d\,)\; queryNorm(q)$$ [6].

Here we can evaluate different lucene query which exact match on keywords will help to give same results as mysql so reduce overhead of processing of extra results.
Below given the system configuration used to take performance measure.
Processor: 1.7 GHz, Core i5
RAM: 4GB, Hard disk: 160 GB,
OS: Fedora 17, java environment: openjdk-1.7.
Apache Lucene 3.6, mysql 5.5

Dynamic Programming and Serialization
Different users have different queries or can be common queries.

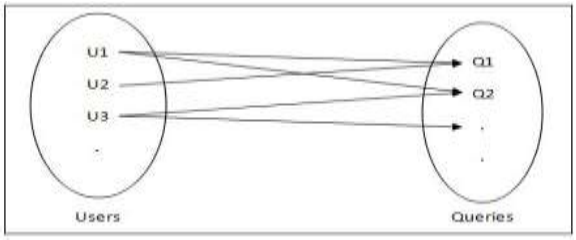

Figure 2: Dependencies between user and queries.

To evaluate performance of each index, each query get tested on every index.

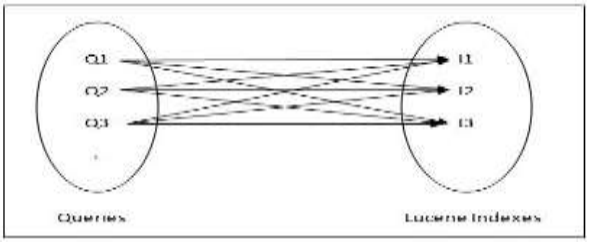

Figure 3: Dependencies between queries and LUCENE indexes.

RESULTS AND DISCUSSION

In performance evaluation the multi threading is used for performing searching as running multiple threads simultaneously and the searching in LUCENE is performed faster so large requests from users are handled properly. The number of input request to input response time is constant in range so the graph will be almost linear.

MYSQL search performance

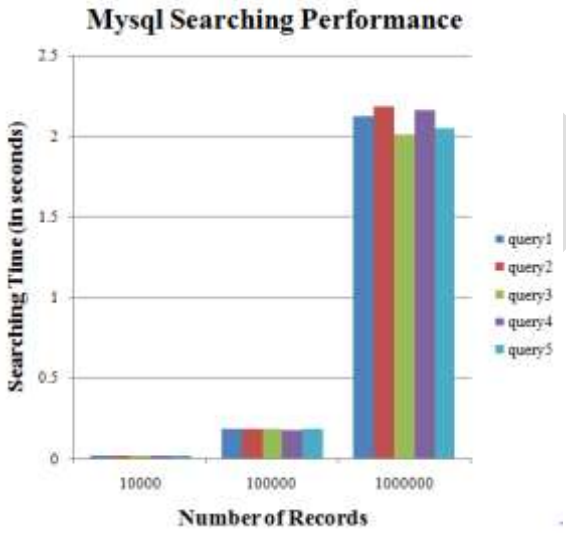

Figure 4: MYQL searching performance

Foe this evaluation we used 1 million records general user information with 28 columns, which is inserted in mysql as well as used to create index in lucene. The user ID is used as primary key to identify particular user information record. The searching is performed in three phases using 10,000 records, 100,000 records and 1 million records.
Figure 4 above showing the mysql searching performance for three phases.

Regardless of search query the searching time in mysql is nearly constant, because mysql searches entire table for any search query which requires the same time for same number of records. The sharp increase in searching time shows that searching time is linearly proportional to number of records. Searching time is not related to how many number of records matches our search query.

Lucene Index Building Performance:

Figure 5 and figure 6 shows Lucene Indexing performance. In figure 5 while running single thread for indexing we can analyze that RAM index takes less time as compare to Simple and NIO index.

Figure 6 shows Lucene Indexing performance while running 4 threads. Building Lucene Indices required slightly less time than single threading approach. With multithreading approach size of Index increases slightly. As RAM index stores data completely in physical memory the RAM index with stored fields is not practical for indices greater than few hundred megabytes in size.

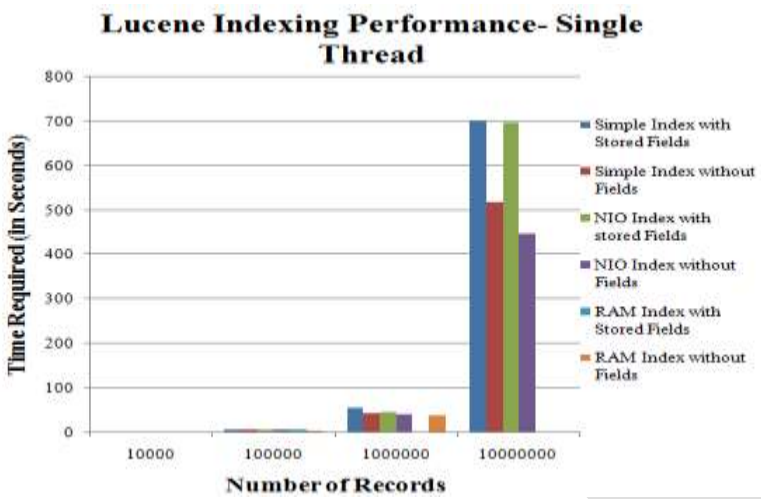

Figure 5: Lucene Indexing performance using single thread

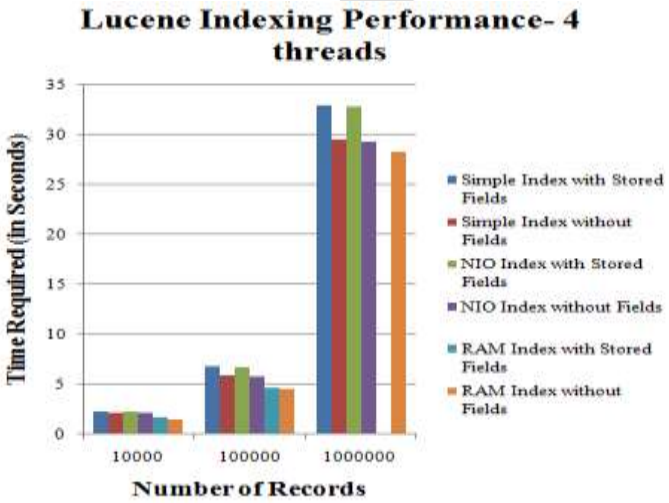

Figure 6: Lucene Indexing performance using 4 threads

Lucene Searching Performance

Figure 7,8,9 shows below Lucene Searching Performance for 10000, 100000, 1 million records. By this analysis it articulate that as compare mysql searching lucene takes very less time. For any type of query RAM searching is best. But it has limitation of physical memory present on the system.

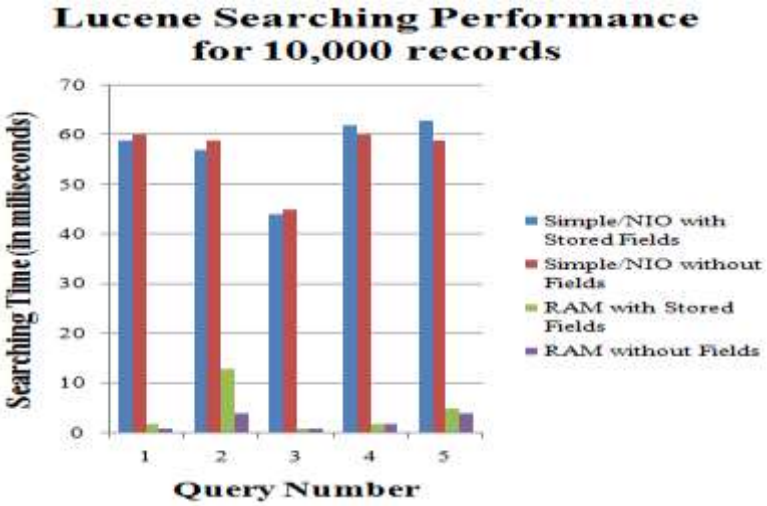

Figure 7: Lucene searching performance for 10,000 records

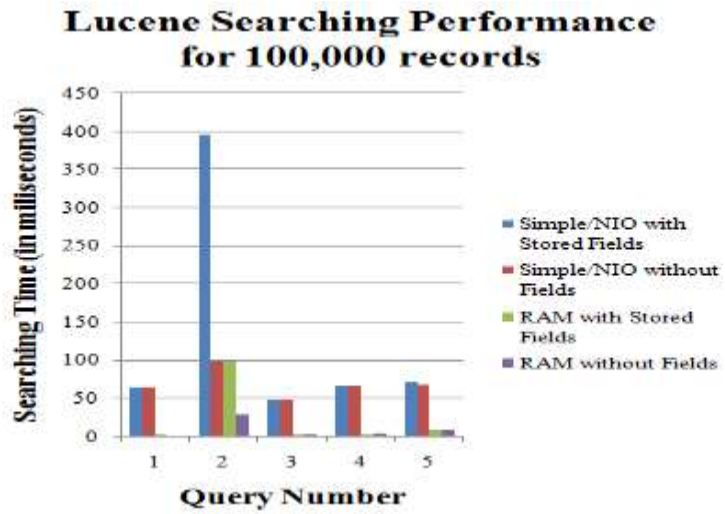

Figure 8: Lucene searching performance for 100,000 records

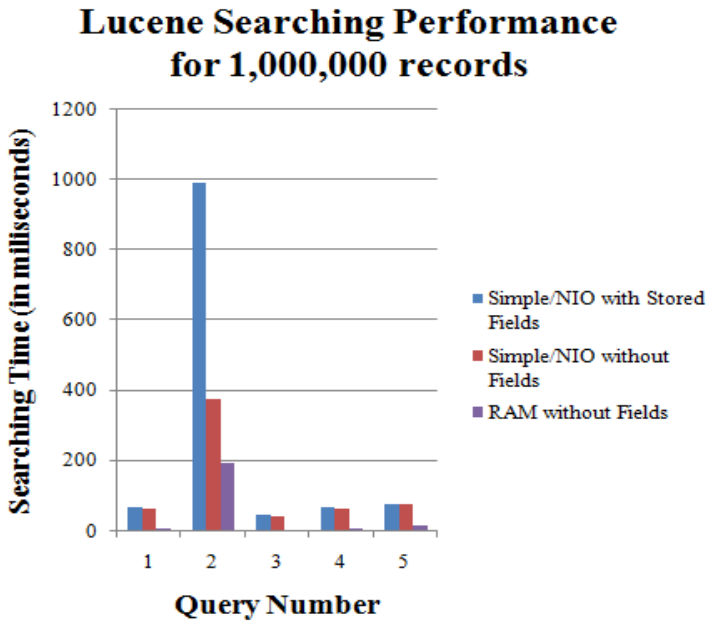

Figure 9: Lucene searching performance for 1 million records

ACKNOWLEDGMENT

I take this opportunity to express my deep sense of gratitude towards my esteemed **guide Prof. Pankaj Agarkar** for giving me this splendid opportunity to select and present this dissertation topic and for encouragement & providing me with the best facilities for my presentation work.

I would like to thank TechModi pvtltd, pune who sponsor this project work and gave me opportunity to work with them, I always thankful for their indispensable support, priceless suggestions and for most valuable time lent as and when required.

CONCLUSION

Lucene searching is very fast, so we can use it to retrieve records in large databases. Although inserting is slow multiple threads can be used to build index. Index with data stored give less performance than index without data stored, and in our scenario data storage is not required as we can retrieve data from database.

Further we can evaluate indexing capability provided by mysql, also bulk search performance need to evaluated using multithreading we can differentiate performance between Simple and NIO indexes. Memory mapped index is new index which can be evaluated for searching. The search history maintenance module provides additional feature to lucene searching framework. Module can be optionally added as plugin at any time to existing system.

This system uses multithreading for searching multiple queries simultaneously on the single server. We can develop the multi server searching which will use multiple servers for processing searching queries. For index building the same server is used. We can use multiple servers for index creation, which will duplicate the indexes.

REFERENCES:

[1] Apache Team, "Apache documentation", Online At http://LUCENEn.apache.org/core/3_6_1/index.html (as of 7 January, 2014)
[2] Deng Peng Zhou, "Delve inside the LUCENE indexing mechanism", Online At http://www.ibm.com/developerworks/library/wa-LUCENE/ (as of 14 January 2014)
[3] A Lucid Imagination White Paper "The Case for LUCENE/Solr: Real World Search Applications" January 2010.
[4] Guaoliang Li, Shenguye Ji, Chen Li, Jianhua Feng, "Efficient fuzzy full-text type-ahead search", The VLDB Journal (2011) 20:617–640 DOI 10.1007/s00778-011-0218-x.
[5] Lucie Molkova, "Indexing Very Large text Data", Brno, spring 2011.
[6] Michael McCandless, Erik Hatcher, Otis Gospodnetic, "LUCENE in Action" 2nd edition, Manning Publications Co, 2010.
[7] Amol Sonawane, "Using Apache LUCENE to search text" Online At http://www.ibm.com/developerworks/ opensource/ library/os-apache-LUCENEsearch/" (as of 11 December 2013)
[8] Shengyue Ji, Guoliang Li, Chen Li, Jianhua Feng, "Efficient Interactive Fuzzy Keyword Search", Apr 20-24, 2009, ACM 978-1- 60558-487-4/09/04