# Utilization Of Bloom Filter For Database Query Optimization

**Victor Hugo Cahyo[1], Normalisa[2], and Yudi Triyana[3]**

[1]Computer Science, International University Liaison Indonesia, Intermark, BSD City, 15310
e-mail: [1] victor.cahyo@stud.iuli.ac.id; [2]normalisa@iuli.ac.id; [3]yudi.triyana@iuli.ac.id

**Abstract. In modern days, NoSQL database system is slowly replacing the traditional relationalbased database system. One of the NoSQL database type that is widely used is a document store databases type as it mimics the design of traditional relational database system, while provides more flexibility on the scheme. Some traditional database query optimization methods can be utilized in a documentstore databases, but the flexibility of the scheme, and lack of a strict consistent structuremakes some query optimizations to be impossible to perform. Bloom filter is a probabilistic data structure that is capable to determine the member- ship of an element in a set. Bloom filter always produce true negative, but may producea false positive. This qualitative study is conducted in order to determine the viability of bloom filter for an optimization method to reduce the time taken to process a query in a NoSQLdocument store databases. The results from this study determine that the implementation of bloom filter with the scheme described in this experiment is only applicable to create a slightly better performance of a query in only some specific cases. Meanwhile in the majority of other cases the bloom filter either does not create a significant performance different, or hinders theperformance of the query.**

**Keywords: NoSQL, Bloom Filter, Query Optimization**

## 1. INTRODUCTION

In these days, a three tiered architecture where the application logic and the database logic are separated within their own self encapsulated module. There are a lot of advan- tages of a modular design, but there are also some disadvantages.

Because of the separation between the application logic module and the database logic module, an IPC communication scheme provided by the Operating System is re- quired to have a proper communication between the modules. The communication pro- tocol is typically located in the application layer of the OSI model of telecommunication, which is the highest layer in the OSI model.

Additionally, to communicate securely within the network, some form of network security and authentication system must be implemented. A data security system is also required to reduce the direct impact from data breach. The additional security system could burden the network and the database machine further.

One of the newest database type introduced in 2007 was NoSQL[4]. NoSQL database has been taking a lot of ground in the database market and slowly replacing relational database as the database in use[7]. NoSQL is a type of database that stores the records in non relational manner, have flexible scheme, and typically horizontally scalable[11]. One type of NoSQL database is a document databases, where each record is a structured data, and has their own independent distinct field[10]. The field in the document database is similar to the column in the traditional relational based database system. The document databases type of NoSQL database has a new type of problem that does not exist, where a field on one record may not existed in the other data.

In a document databases, requesting the content of an invalid field in the record is guaranteed to either fail or have null records. The system designer could create a design with a strict limits on the fields, and enforce it to the programmers during the develop- ment of the system, but it reduces the flexibility of the database scheme, which is one of the selling point of NoSQL. Additionally some programming errors or bugs during the creation of the database query are inevitable, hence an improvement method that could detect, report, and reduce the impact from such bugs is a welcomed feature.

Some optimization methods for database have been utilized, but mostly improve on reducing the initial overhead of database connection[9]. In this research, additional database query improvement based on the reduction of invalid field from the database request is analyzed.

## 2. LITERATURE REVIEW

In August 2017 in a conference named International Conference on Engineering Applications of Neural Networks[8], Dritsas et. al. presented their research and findings about the utilization of bloom filter on an relational SQL database in optimizing the execution time of SQL query. In the research, a bloom filter is used to filter out data before search- ing the data in the record index of the database. The researchers tested the following operations: In, Inner Join, Left Join, Right Join, Exists and Top.

The experiment is done by making a comparison for processing time for a query sent to an SQL server with and without bloom filter. The testing is done with a database containing between 1.000.000 and 10.000.000 data with increment of 1.000.000. The finding from the experiment is that for all SQL commands, the bloom filter overloaded the system, and took longer to respond to the query for small number of records. In the experiment with 8.000.000 or more data entries, the bloom filter started to create an advantages and reduce the execution time significantly. The authors of the paper demonstrated the capability of the bloom filter to detect nonexistent records before looking up the records in the index table. The bloom filter was also implemented as an optimization scheme for SQL based relational database in the server side.

A paper written by Christian Antognini is published in 2008 [3]. The paper details the usage of bloom filter in the RDBMS database system developed by Oracle. In the paper, the author explained that the bloom filter is mainly employed in three situations:

1. To reduce data communication between slave processes in parallel joins.
2. To implement join-filter pruning.
3. To support result caches.

Joining is an action to take records from multiple tables, and join them together in a single SQL query. Parallel join is a join operation where the database engine spawns multiple processes, and then delegates the operation of generating the result set to the slave processes. The join operation is done by a process located higher in the hierarchy before sending the result set to the query coordinator process. The processes that are located the lowest level in the hierarchy are responsible to read the records from the table, and generate new sets containing the records from each table. The sets are further filtered by the reader processes, and passed to the joiner process. The records in the set sent by the reader process may be dropped because it does not fulfill the condition required for the join operation.

The query coordinator decides whether to use the bloom filter based on the join selectivity of the query. When a bloom filter is deemed appropriate, the joiner process creates a shared bloom filter. One of the reader process populates the bloom filter with its own data, and the other reader probes the bloom filter before including the records to the result set. The bloom filter in the Join-Filter pruning procedure is used by the optimizer when the database system receives a request for a join operation, followed by a filter operation. The optimizer scans the query request and tries to prune redundant parameter from the filter operation request. Up to Oracle Database version 10g Release 2, an optimization named subquery-pruning is used to optimize the query. This optimization method is used in limited manner, because parts of the query are scanned and executed twice during the optimization process. Meanwhile Oracle Database version 11g introduces the Join-filter pruning optimization to replace the subquery-pruning. In join-filter pruning method, a bloom filter is used to avoid the double execution, which makes it far more viable to use.

## 3. METHODOLOGY

The experiment is consisted with loading the front page of the management e-commerce site, which contain all of the data groups. The time taken from the call of the database query until the data ready to be used by the client application is measured. Before any database query being done, the server computes the bloom filter and sent it to the client outside the measured time.

The measurement is taken by obtaining the timestamp with Javascript's build in Per- formace interface before the query request call, and after the records is ready to be served by the client. The different between the timestamp is computed and used as the data point. The independent variable in the experiment is the number of records in the database, and the number of optional fields in the records. The dependent variable is the time taken between the call of the client side function to query the database, and the time taken for the content of the records to be ready to be used. The number of records is increasing by exponent of 2 in range from 26 until 211. The detail of the distribution of the number of records can be seen in table 1.

Table 1. Number of Records

| No. | Number of records | Number of records |
|---|---|---|
| 1 | $2^6$ | 64 |
| 2 | $2^7$ | 128 |
| 3 | $2^8$ | 256 |
| 4 | $2^9$ | 512 |
| 5 | $2^{10}$ | 1024 |
| 6 | $2^{11}$ | 2048 |

The other independent variable in the experiment is the number of maximum op- tional fields as shown in the table 2. Every record in the database has randomized number of optional fields, but the number of the optional fields can not exceed the max- imum limit. The name of the fields is randomly selected with the format as explained in the section 3.1. The grouping for the upper limits is based on a linear increment of 10 between each experiment groups.

The control group is a measurement with the bloom filter function being bypassed, while the test group is a measurement with the bloom filter fully functional to filter out the optional fields that is not the member of the set. To increase the accuracy of the data, the measurement is repeated 3 times with different dummy code to simulate the randomness of the real world data, and the result is averaged to get the result data point for the experiment.

Table 2. Maximum Number of Optional Fields

| No. | Maximum Number of Optional Fields |
|---|---|
| 1 | 10 |
| 2 | 20 |
| 3 | 30 |
| 4 | 40 |

To perform the optimization, firstly a new array named buffer is created, and initial- ized with an empty array. Then the id array is checked to determine whether it is empty or not. If the id is empty, then the buffer is passed as the return value of the subroutine. If the id still contain at least one element, the element is taken out from the id array, and stored as a new variable called current. A new buffer called optimizedField is also created and initialized with an empty array.

Another check to find out the content of current.field is performed. If the current.field is empty, then the content of current.field is replaced with optimizedField, and the cur- rent variable is pushed to the buffer array. Then the subroutine loops back to the step where it checks whether the id is empty. Otherwise a bloom filter check is performed with the first member of the current.field. If the first member of the current.field is a member of the filter, the first member of the current.field is inserted to the optimizedField array. Finally the first member of current.field is removed, and the program loops back to the step where it checks whether the current.field is empty. The loop back is the iterative approach version in a flowchart to perform an action for all element of the set, or also known as fold algorithm.

## 4. RESULT

There was 3 repetitions for each permutation of the independent variables, and the result value is obtained by taking the mean average of the raw data. The averaged results of the experiment after can be seen in the table 3 for the control group with the bloom filter bypassed, and in the table 4 for the test group with the bloom filter utilized. The averaged error rate of the query can be seen in the table 6

The following tabel 3 and tabel 4 are the result data from the experiment in mil- lisecond. The row in the table denotes the axis for the maximum number of possible optional fields, with the top row as the number of possible fields. The column indicates the axis for the number of records in the database, with the leftmost column as the num- ber of records.

The difference between the time taken for query with and without bloom filter is shown in the table 5 The time difference is the time saved on the database query by the bloom filter, and determine the success of the bloom filter in optimizing the database query.

Table 6 displays the error rate for each query in percent. Error rate is computed by counting the number of field that has value of null. The number of null value is then reduced by the number of mandatory fields, and then divided by the total number of fields requested by the query to get the rate of error.

Table 3. Time Taken for Query Without Bloom Filter in milliseconds

| | 10 | 20 | 30 | 40 |
|---|---|---|---|---|
| 64 | 29 | 21.7 | 22.8 | 23.8 |
| 128 | 41.8 | 35.8 | 34.1 | 43.8 |
| 256 | 63.5 | 62.7 | 63.7 | 65.3 |
| 512 | 113.6 | 114.1 | 98 | 92.6 |
| 1024 | 214.5 | 208.4 | 209.7 | 226.4 |
| 2048 | 474.3 | 557.5 | 469.8 | 523.2 |

Table 4. Time Taken for Query with Bloom Filter in milliseconds

| | 10 | 20 | 30 | 40 |
|---|---|---|---|---|
| 64 | 23.4 | 25 | 24.8 | 29.3 |
| 128 | 42.9 | 43.6 | 44.6 | 46.8 |
| 256 | 54.6 | 55.7 | 77.1 | 76.1 |
| 512 | 104.5 | 106.1 | 123.4 | 140.8 |
| 1024 | 221.8 | 262.5 | 295.6 | 308.8 |
| 2048 | 515.7 | 595.5 | 691.9 | 758.5 |

Table 5. The Difference of Time Taken Between Query in the Test Group and the Control Group in milliseconds

| | 10 | 20 | 30 | 40 |
|---|---|---|---|---|
| 64 | 5.6 | -3.3 | -2.0 | -5.6 |
| 128 | -1.0 | -7.7 | -10.5 | -2.9 |
| 256 | 8.9 | 7.0 | -13.5 | -10.8 |
| 512 | 9.1 | 7.9 | -25.4 | -48.2 |
| 1024 | -7.3 | -54.1 | -85.9 | -82.4 |
| 2048 | -41.4 | -37.9 | -222.1 | -235.3 |

Table 6. Error rate of the respon query

| | 1 0 | 2 0 | 3 0 | 4 0 | Reff |
|---|---|---|---|---|---|
| 64 | 0.27 % | 1.32 % | 2.77 % | 4.33 % | 42.7 % |
| 128 | 0.21 % | 1.07 % | 2.91 % | 4.63 % | 41.3 % |
| 256 | 0.22 % | 1.12 % | 2.74 % | 4.73 % | 40.6 % |
| 512 | 0.22 % | 1.12 % | 2.60 % | 4.52 % | 40.3 % |
| 1024 | 0.21 % | 1.14 % | 2.71 % | 4.57 % | 40.16 % |
| 2048 | 0.22 % | 1.13 % | 2.72 % | 4.81 % | 40.08 % |

## 5. CONCLUSION

In conclusion, the hypothesis of this experiment is not supported by the findings of the experiment. Despite the capability of the bloom filter to optimize the query in certain cases, the gain obtained from the filter is not that substantial. Meanwhile in the majority of cases, the bloom filter fails to provide perceivable optimization, or even hampers the query itself.

[1] Burton H. Bloom. “Space/Time Trade-Offs in Hash Coding with Allowable Errors”. In: *Commun. ACM* 13.7 (1970-07), p. 422426. ISSN: 0001-0782. DOI: 10 . 1145 / 362686. 362692. URL: https://doi.org/10. 1145/362686. 362692 (cit. on p. 7).

[2] J. Postel. *The TCP Maximum Segment Size and Related Topics*. Tech. rep. RFC Editor, 1983-11. DOI: 10. 17487/RFC0879. URL: https://www. rfc-editor. org/rfc/ rfc879 (cit. on p. 33).

[3] Christian Antognini. “Bloom Filter”. In: (2008-06). URL: https://antognini. ch/ papers/BloomFilters20080620. pdf (cit. on p. 5).

[4] Neal Leavitt and Lee e Garber. “Will NoSQL Databases Live Up to Their Promise?”In: *Computer* 43 (2 2010-02-08), p. 1214. DOI: 10. 1109/MC. 2010. 58 (cit. on p. 1).

[5] D. Borman. *TCP Options and Maximum Segment Size (MSS)*. Tech. rep. RFC Editor, 2012-06.

DOI: 10.17487/RFC6691. URL: https://www.rfc-editor.org/rfc/rfc6691.txt (cit. on p. 33).

[6] Meenu Dave. "SQL and NoSQL Databases". In: *International Journal of Advanced Research in Computer Science and Software Engineering* (2012-08) (cit. on pp. 9, 10).

[7] Matt Asay. "NoSQL databases eat into the relational database market". In: (2015-03-04). URL: https://www.techrepublic.com/article/nosql-databases- eat-into-the-relational-database-market/ (cit. on p. 1).

[8] Eirini Chioti et al. "Bloom Filters for Efficient Coupling Between Tables of a Database". In: 2017-08, pp. 596–608. ISBN: 978-3-319-65171-2. DOI: 10.1007/978-3-319- 65172-9_50 (cit. on p. 5).

[9] Michael Aboagye. "Improve database performance with connection pooling". In: (2020-10-14). URL: https://stackoverflow.blog/2020/10/14/improve-database-performance-with-connection-pooling/ (cit. on p. 2).

[10] MonggoDB. URL: https://www.mongodb.com/document- databases (cit. onp. 1).

[11] Lauren Schaefer. *What is NoSQL?* URL: https://www.mongodb.com/nosql-explained (cit. on p. 1).

[12] Christof Strauch. *NoSQL Databases*. URL: https://www.christof-strauch.de/nosqldbs.pdf (cit. on pp. 9, 11).