ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 12, December 2015

Program Optimization for Query Submission

Sagar Maruti Pujari [1], Anilkumar Warad [2,]

P.G. Student, Dept. of Computer Engineering, Dhole Patil College of Engineering, Pune, Maharashtra, India[1]

Assistant Professor, Dept. of Computer Engineering, Dhole Patil College of Engineering, Pune, Maharashtra, India[2]

ABSTRACT: The performance of application is depending on backend calls like database calls and/or web services. As number of calls increases at backend side, the performance of application decreases as it calls and/or executes same query and/or procedure/functions multiple times. So it degrades performance. This problem can be solved by creating a batch of parameters, and by rewriting the query/procedure. It is based on program analysis and rewrite rules, for automate the generation of batched forms of procedures and replace iterative database calls within imperative loops with a single call to the batched form. Such manual rewriting is time-consuming and error prone. The program transformation method depends on data flow analysis which handles query executions within loops. At runtime, it can combine multiple asynchronous requests into batches. The author has used asynchronous query submission and batching technique together to improve performance gain. The author has presented DBridge, a novel static analysis and program transformation tool to optimize database access. It implements these program transformation techniques for Java programs that use JDBC to access database. It uses the SOOT optimization framework for static analysis. The author has presented SOOT framework for optimizing Java bytecode.

KEYWORDS: program analysis, query optimization, program transformation, sub queries.

I. INTRODUCTION

Many database applications perform queries and updates from within procedural code that contains business logic. Parameter batching is an important technique to speedup iterative execution of queries and updates. It allows the choice of efficient set-oriented plans for queries and updates. Database applications perform queries and updates from within procedural code that contains business logic. The author has proposed a program transformation approach for rewriting applications to use batched parameter bindings. The transformation rules make use of information about inter-statement data dependencies gathered from static analysis of the program. The proposed program transformation rules are powerful enough to rewrite a large class of loops involving complex control flow and arbitrary level of nesting. DBridge is a program transformation tool based on Java applications that use JDBC API to access database. The tool performs static analysis of the input program and identifies opportunities for replacing iterative database access with set oriented access. Then it rewrites the application code and the embedded queries together for set oriented processing. DBridge is designed to be a source-to-source transformation tool, and to this end, it ensures readability and maintainability of the transformed code.

Iterative execution of parameterized SQL queries can be replaced by a batched form or set oriented form of the query. The performance of applications can be improved by asynchronous submission of queries. The author has proposed techniques based on program analysis and rewriting to prefetch the results of queries or Web service requests that are subsequently issued by a program.

In many applications, queries and updates to a database are frequently executed repeatedly, with different values for their parameters. Iterative execution plans of queries and updates are often very inefficient. It increases the network round trip so it degrades the performance of application. The performance of application can be improved by rewriting nested queries using set operations or query optimizer. The author has presented an approach which is based on program analysis as well as transformation, to automatically generate batched forms of procedures. An important technique to speed up repeated invocation of such procedures/functions is parameter batching which allows the choice of efficient set-oriented plans for queries and updates.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 12, December 2015

To perform effective optimizations, use method inlining and static virtual method call resolution which reduces the use of bytecodes. Java's execution is safe which do not allow to access illegal memory accesses are checked for safety before execution. In some case it can be determined at compile-time. If we use annotation mechanism, then the Java Virtual Machine could speed up the execution by not performing these redundant checks. The SOOT framework provides a set of intermediate representations and APIs for optimizing Java bytecode. The SOOT framework produces bytecode from a source like javac compiler. It transforms and/or optimizes the code and produces new class files. This new bytecode can then be executed using any standard JVM implementation.

II. RELATED WORK

The program plans represents control and data dependences in a modularized form in which loops have been converted to recursion. The data dependence graphs are designed for the hierarchical analysis of dependence relations in programs. The control dependence representation has one major advantage over the data dependence graphs, in that the need to convert control dependence into guards is eliminated. The Data Flow Graph represents global data dependence at the operator level is called as atomic level. The Extended Data Flow Graph represents control and data dependence. It represents structured programs.

Java tool which performs significant optimizations on bytecode and produces new class files is Jax. Jax is application compression which removes unused methods and fields and the class hierarchy is compressed. It's used to speed up optimizations. Java tools for manipulating bytecode like JTrek, Joie, Bit and JavaClass are constrained to manipulating Java bytecode in their original form. They do not provide convenient intermediate representations such as BAF, JIMPLE or GRIMP for performing analyses or transformations. To package Java application, there are a number of tools like Jax, DashO-Pro and SourceGuard which consists of code compression and/or code obfuscation. The author has not yet applied SOOT to this application area. He has plans to implement this functionality. The tools in Java native compilers category take Java applications and compile them to native executables. They build 3-address code intermediate representations, and some perform significant optimizations. Toba which produces unoptimized C code and relies on GCC to produce the native code. Harissa produces C code but performs some method devirtualization and inlining first. Vortex is a native compiler for Cecil, C++ and Java which contains a complete set of optimizations. Marmot is also a complete Java optimization compiler. There are two types of Java compiler infrastructures namely Flex and Suif compiler system. Flex is a Java compiler infrastructure for embedded parallel and distributed systems which uses a form of SSA intermediate representation for its bytecode called QuadSSA. Suif compiler system possesses a front-end which translates Java bytecode to OSUIF code which is an object oriented version of Suif code which can express object orientation primitives

The indexed nested loops operation is not modified, and continues to issue synchronous IO operations; the AIO performed earlier basically prefetches data, which reduces the probability of the synchronous IO blocking, and allows the disk subsystem to optimize fetching of data. The buffer can be refilled when it is empty (batch mode), or each time a record is removed from the buffer on completion of an asynchronous request (sliding window mode). Graefe also mentions that Microsoft SQL Server and IBM DB2 keep a fixed set of unresolved IO requests and IO is started concurrently for all these elements. Graefe [4] does not provide any experimental results. In contrast to the results in [4], our asynchronous iterator model allows entire subplans to be non-blocking.

III. PROGRAM TRANSFORMATION

The author has selected Java as the target language, SOOT optimization framework and JDBC as the interface for database access. To implement the examples, it needs to perform data flow analysis of the given program and build the data dependence graph. The main task of our program transformation tool present in the Apply Async Trans phase. The runtime library works as a layer between database API and application. It provides asynchronous as well as batched query submission. Features like thread management and cache management are handled by libraries.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 12, December 2015

IV. EXPERIMENTAL RESULTS

Experiment 1: Auction application.

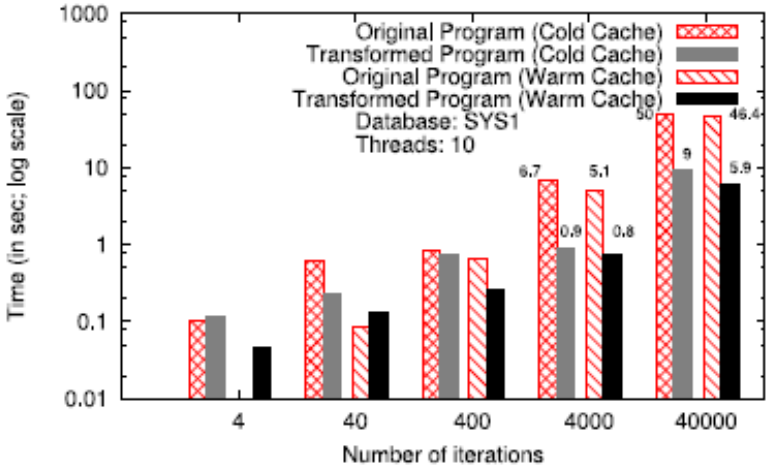

Fig. 1- Experiment 1 with varying number of iterations.

Fig. 1 shows the performance of this program before and after the transformations with warm and cold caches in log scale. The y-axis denotes the end to end time taken for the loop to execute, which includes the application time and the query execution time. For a small number of iterations, the transformed pro-gram is slower than the original program. If number of iterations increases then it will give transformations benefit.

Experiment 2: Experiment 1(Auction application) with varying number of threads

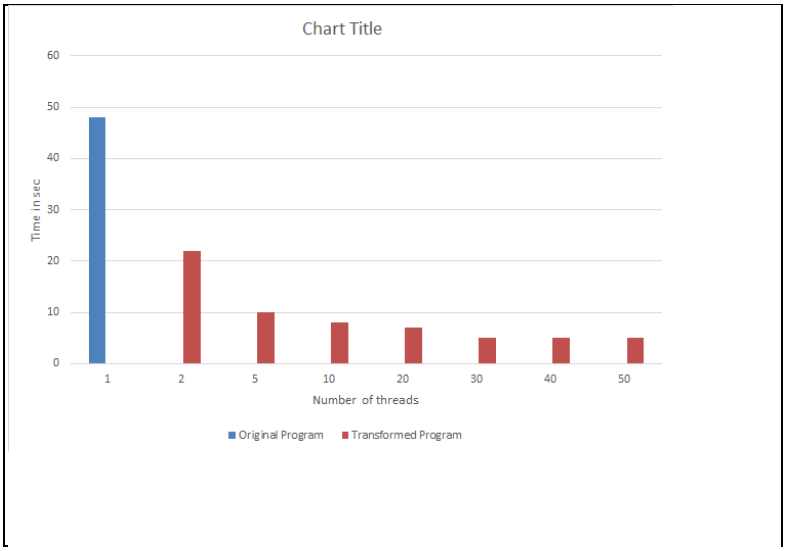

Fig. 2- Experiment 1 with varying number of threads.

As the number of iterations increases, next, we keep the number of iterations constant (at 40,000) and vary the number of threads. The results of this experiment are shown in Fig. 2. The execution time (for both the warm and cold

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 12, December 2015

cache) drops sharply as the number of threads is increased, but gradually reaches a point where the addition of threads does not improve the execution time.

Experiment 3: Bulletin board application

For this experiment, author has considered the scenario of listing the top stories of the day, along with details of the users who posted them. Fig. 2 shows the results of our transformations with different number of iterations. Although the transformed loop in the program takes slightly longer time for small number of iterations, the benefits increase with the number of iterations (note the log scale of y-axis).

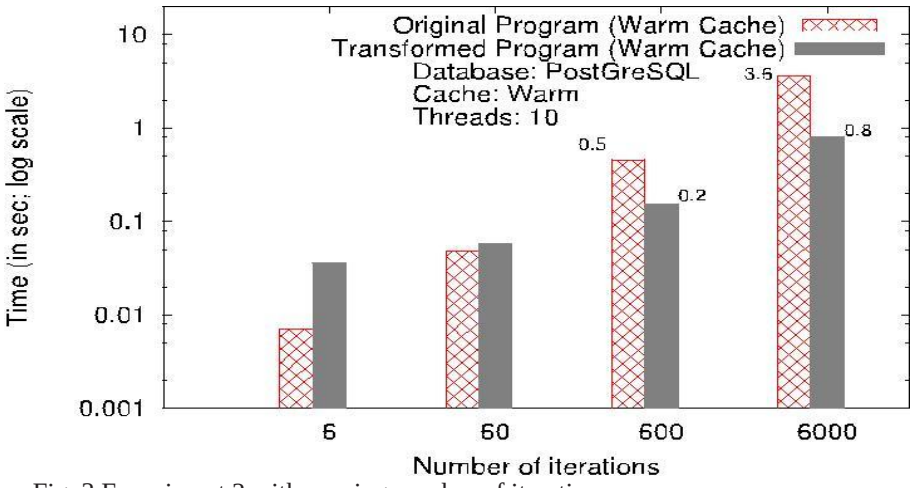

Fig. 3 Experiment 2 with varying number of iterations.

Fig. 3 shows the performance of this loop in the program before and after applying our transformation examples. As in the earlier example, author has fixed the number of threads and varies the number of iterations. The author has performed this experiment with ten threads, on a warm cache on SYS1. The results are in accordance with our earlier experiments. He has observed that the number of threads is an important parameter in such scenarios. This parameter is influenced by several factors, such as the number of processor cores available for the database server and the client, the load on the database server, the amount of disk IO, CPU utilization etc.

V. CONCLUSION

Query optimization is based on program analysis and transformation techniques. It gives significant benefits for database applications. This paper proposes a program analysis and transformation based approach to automatically rewrite database applications to exploit the benefits of asynchronous query submission. They also described a novel approach to combine asynchronous submission with batching. It saves total execution time versus time to kth response, reducing network round trips (by batching multiple requests) versus overlapping execution of queries and reducing memory consumption (By using iterative query execution) versus set oriented execution of the query. The author has presented DBridge, a tool that combines program and query transformations to make query execution set oriented. It optimizes database access by performing optimizations. SOOT framework which simplifies the task of optimizing Java bytecode.

REFERENCES

[1] R. Guravannavar and S. Sudarshan, "Rewriting procedures for batched bindings," in Proc. Int. Conf. Very Large Databases, 2008 pp. 1107–1123.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 12, December 2015

[2] M. Chavan, R. Guravannavar, K. Ramachandra, and S. Sudarshan, "DBridge: A program rewrite tool for set-oriented query execution," in Proc. IEEE 27th Int. Conf. Data Eng., 2011, pp. 1284–1287.
[3] K. Ramachandra, R. Guravannavar, and S. Sudarshan, "Program analysis and transformation for holistic optimization of database applications," in Proc. ACM SIGPLAN Int. Workshop State Art Java Program Anal., 2012, pp. 39–44.
[4] M. Chavan, R. Guravannavar, K. Ramachandra, and S. Sudarshan, "Program transformations for asynchronous query submission," in Proc. IEEE 27th Int. Conf. Data Eng., 2011, pp. 375–386.
[5] R. Guravannavar, "Optimization and evaluation of nested queries and procedures," Ph.D. dissertation, Dept. Comput. Sci. Eng., Indian Inst. Technol., Bombay, India, 2009.
[6] J. Ferrante, K. J. Ottenstein, and J. D. Warren, "The program dependence graph and its use in optimization," ACM Trans. Program. Lang. Syst., vol. 9, no. 3, pp. 319–349, 1987
[7] K. Kennedy and K. S. McKinley, "Loop distribution with arbitrary control flow," in Proc. ACM/IEEE Conf. Supercomput., 1990, pp. 407–416.
[8] A. Manjhi, C. Garrod, B. M. Maggs, T. C. Mowry, and A. Tomasic, "Holistic query transformations for dynamic web applications," in Proc. IEEE 25th Int. Conf. Data Eng., 2009, pp. 1175–1178.
[9] G. Graefe, "Executing nested queries," in Proc. 10th Conf. Database Syst. Business, Technol. Web, 2003.
[10] M. Elhemali, C. A. Galindo-Legaria, T. Grabs, and M. M. Joshi, "Execution strategies for SQL subqueries," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2007, pp. 993–1004.
[11] S. Iyengar, S. Sudarshan, S. Kumar, and R. Agrawal, "Exploiting asynchronous IO using the asynchronous iterator model," in Proc. Int. Conf. Manage. Data, 2008, pp. 127–138.
[12] A. Dasgupta, V. Narasayya, and M. Syamala, "A static analysis framework for database applications," in Proc. IEEE Int. Conf. Data Eng., 2009, pp. 1403–1414.
K. Ramachandra and S. Sudarshan, "Holistic optimization by prefetching query results," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2012, pp. 133–144.