Billing Performance Enhancement of a Pricing and Billing Engine

Hari Rajan A
Student
Computer Science and Engineering Department
Government Engineering College, Thrissur, Kerala, India

***Abstract-* miRevenue is a pricing and billing engine that focus on enabling banks to deliver sustainable revenue enhancement to their targeted business needs. miRevenue has billing process that runs on the required accounts on the specific date. As part of the billing process the system processes Billing Groups. Billing Group is a collection of Charge Records for the purposes of preparing data for generating Settlements, and indirectly, Statements. A Billing Group is related to a Billing Preference through the Billing Profile. A Billing Profile is structure for associating the Billing Profile Key to Billing Groups, Statement Groups, and various preferences for the purpose of enabling a flexible configuration for the Settlement and Statement processes. A Billing Preference specifies the settlement and statement frequencies and a billing process offset, settlement offset and statement offset. Currently, during the billing process, the preference associations associated with the accounts are fetched on the fly. The system currently maintains a hierarchy which starts from Entity Group and then Customer and then Portfolio and then Account. So, if a specific account does not any preference associations attached it looks in its parents and uses the preference association it gets from there. This process continues till the root level i.e., the Entity Group. This preference association fetching can take time depending up on the level at which each account has its association configured i.e., billing process is dependent on this. The proposed work is to do a research on how the pricing and billing engine works now and there by design a new algorithm, which is in line with the working principle of miRevenue, which would have all the preference associations ready before hand itself. Once this design is implemented, the billing process will be independent of the preference associations of the accounts processed and will be completed in the required time interval. The purpose of the research is to gain banking domain knowledge and enhance the pricing and billing engine. The methodology that would be adopted to do the research would be as follows:**
1) Learn the working principle of the product.
2) Learn how the billing process currently runs.
3) Learn how the preference associations are attached or fetched now.
4) Design a new algorithm that would have the preference associations ready before hand (i.e, before the billing process)

I. About miRevenue

miRevenue, focus on enabling banks to deliver sustainable revenue enhancement to their targeted business needs.Today, miRevenue goes beyond relationship banking, which is composed of basic relationship pricing and consolidated billing functionality. miRevenue fulfils the function of a Product and Pricing Lifecycle Management platform, which requires additional functionalities that banks need in the areas of product management, profitability simulation, and offer management and on boarding. As such, miRevenue is the first complete Product and Pricing Lifecycle Management (PPLM) platform, delivering a “start-to-finish” view of the way banks bring products to market quickly and easily with a focus on revenue enhancement. miRevenue pricing and billing platform has been architected for handling high volume batch and online processing volumes being executed on an ongoing basis. miRevenue uses a batch framework.
There are a number of fundamental principles behind the batch processing architecture.They are:

- Division of work into multiple independent pieces to enable scalability
- True parallel processing requires that no two Worker Threads need access to the same resource to complete an operation.
- Each Worker Thread must not require, nor exclude, the existence of a master control thread to complete its assigned work.
- An out-of-band mechanism must exist that allows control over the Worker Threads.
- Information that is used to coordinate, recover and manage the processing must reside in the database.
- A batch is composed of the following levels of work:
- A single Work Item (e.g. a banking transaction or a customer update)
- A Commit Set groups a number of Work Items to be operated on within a single database transaction (controlled by the batch commit frequency)
- A Work Chunk with a number of Commit Sets to be processed between testing for control instructions.
- A Task which groups all common Work Items into a Work List.
- A Job which is divided into sequential Tasks.
- The Batch which is divided into Jobs which may be executed in parallel.

II. TERMINOLOGY

Below are some of terminologies related to this product and which are used further in this document.

- Billing - The process of producing Statements and sending Settlements to the core banking systems.
- Billing Group - A collection of Charge Records for the purposes of preparing data for generating Settlements, and indirectly, Statements.
- Billing Preference - Various common preferences that drive both the Statement and Settlement processes.
- Billing Profile - A structure for associating the Billing Profile Key to Billing Groups, Statement Groups, and various preferences for the purpose of enabling a flexible configuration for the Settlement and Statement processes
- Charge Account Preference - Specifies the debit and credit accounts that calculated Charge Records of a given Transactional Entity would be settled against for each Charge Code.
- Revenue Account Preference - Specification of the bank's revenue account to be used for the contra entry that calculated Charge Records of a given transaction account would be settled against.
- Statement Preference - Configuration preferences specific to Document generation, primarily including Document Subscriptions.

III. OBJECTIVES

The area selected to do the research work is to enhance the billing process performance of the product. The billing process requires huge amount of calculations to be done which are needed later for statement generation and settlements. For these calculations to be performed, the product requires to know preference associations related to each account. Preference association decides what criteria needs to be applied for these calculations.

Currently, in the billing process, the algorithm performed for fetching these preference associations is in such a way that the associations are fetched on the fly i.e. if an entity, say, Account is being processed, then the association associated with that entity is fetched by firing an SQL. If that entity does not have a direct association, then the association is fetched from its parent. This process is continued till an association is obtained for that specific entity. This process is dependant up on the volume of data to be processed and also on the levels at which the associations are configured. If the volume of data is too large, the processing time increases which further violates the time limit agreed upon for the completion of the process.

IV. LITERATURE REVIEW

Below papers were referred and studied to know more about the technology scopes and thereby arrive at a solution for the problem.

1) ***SQL Query Optimization Methods of Relational Database System***

Oracle provides several options to aid performance, such as partitioning large tables, using materialized views, storing plan outlines, and many others. This chapter examines how DBAs can use these techniques to aid developers' efforts to increase the efficiency of their application code. This chapter introduces the SQL Tuning Advisor to help you tune SQL statements. You can then use the recommendations of this advisor to rewrite poorly performing SQL code. I begin the chapter with a discussion of how to approach performance tuning. More than the specific performance improvement techniques you use, your approach to performance tuning determines your success in tuning a recalcitrant application system.

Published in:

Computer Engineering and Applications (ICCEA), 2010 Second International Conference on (Volume:1)

Date of Conference:

19-21 March 2010

2) ***Towards Detecting Thread Deadlock in Java Programs with JVM Introspection***

Abstract-Deadlock is a common error for multithread Java programs. Existing Java thread deadlock detection solutions either require source code, or are built on non-official JVMs. In a consequence, a great number of Java programs cannot be evaluated with these solutions. This paper proposes a new Java thread deadlock detection approach, namely JDeadlockDetector. JDeadlockDetector is built on the official Java Virtual Machine (JVM), viz., OpenJDK's HotSpot. Compared to existing methods, JDeadlockDetector achieves three unique advantages, i.e., application transparency, detection accuracy and minimized performance overhead. Our functionality evaluation shows JDeadlockDetector achieves no false negative and minimized false positive while the performance evaluation shows the workloads generated by SPECjbb2005 achieve 96.7% of official JVM speed on average.

Published in:

Trust, Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th International Conference on 16-18 Nov. 2011

3) ***Design and Implementation of the Hibernate Persistence Layer Data Report System Based on J2EE***

The process of research and development of data report system for financial management is described in this paper. It chooses J2EE as platform, JasperReport as development tools, for the realization of the report system with hibernate to meet the requirement of users. In the process of project development, we pay more attention to the study of the mechanism of the report generation by JasperReport. The advantages of object-oriented (O/R) database are described, as well as the superiority of Hibernate, finally, we design the layout of the database with IReport. Traditional relational database is transformed into durable objects with the O/R mapping tools through Hibernate, so the database can be operated by the form of object, which will be used

to develop the persistent objects mapped by the underlying data structure, and provide the service, such as querying, searching, and generating reports for the advanced data.
Published in:
Circuits, Communications and Systems, 2009. PACCS '09. Pacific-Asia Conference

V. METHODOLOGY

The dependency of the billing process on the volume of data for which the preference association fetching needs to be done has to be eliminated. The work involved here is to study the current algorithm followed to perform the billing and identify the area which impacts the time. Once the area is identified, a new algorithm which is independent of volume of the data is to be proposed and implemented. The new algorithm should have a mechanism in which the preference association fetching is done outside the billing process time limit.

Below are the steps of the algorithm proposed to better the billing process.
Direct Associations
- Delete all the resolved preference associations for the current billing profile.
- Fetch all the direct associations associated.
- If there are any child entity group elements for the nodes, make them eligible to be resolved.

Stage Unresolved Billing Profiles
- Create a new Task Processor.
- The above task processor stages all the entries which needs to be processed.
- The staging would populate the grouping key of the staging table with the linked entity type values (for ex: Portfolio, Account etc.)
- A new task is added which uses this processor

Processed Staged Entries
- Create a new processor
- The above processor should process the staged entries
- The processing should happen in the required hierarchical order starting from Entity group first and then downwards. This hierarchical processing can be achieved by providing the Grouping key as a task attribute.

For example if the hierarchy of processing to be followed is Entity Group, Customer and Account in the order. We would have entries in the staging table with Grouping_key with values for EG, customer and account. The Job Definition would have below three tasks in the sequential order so that the processing happens as required.

Correcting Existing System
- A task processor which truncates the table which contains the preference associations and populates all the billing profiles into it with all the preference associations as null.
- A task which uses this processor.

New Job
- A new job which would use all the above tasks is need. This is the job which runs before the billing process. In this job, each task would be ordered as required and hence when this job is completed, all the required preference associations are resolved and stored in the database.

VI. RESULT

The job proposed above in the methodology section would be ran before the billing process. Once the job completes, all the required associations are fetched and is made readily available in a separate table.
As a result of the proposed solution, the preference association fetching will be moved out of the billing process i.e. the before the billing process runs, all the required preference associations required will be fetched and stored in a separate table. Hence, it is only required to directly fetch the data from the table and thereby reducing the overhead of firing the SQL query and finding the preference associations during the billing process.
Doing direct selects on a table is very much more efficient that firing complex SQL queries to find the preference associations associated a specific Entity.
Once this solution is implemented, the product is expected to perform the billing process much faster that it does now and hence it would not be an issue even if the volume of data to be processed increase in future.

VII. REFERENCES

[1] http://docs.oracle.com/javase/tutorial/essential/concurrency/
[2] http://en.wikipedia.org/wiki/Batch processing
[3] http://sqltuning.com/oraclesqlperformancetuning/
[4] http://spring.io/guides/gs/maven/
[5] http://en.wikipedia.org/wiki/Nested set model
[6] Database programming using Java, Swain, M. , Anderson, J.A. ; Korrapati, R. ; Swain, Nikunja K.
[7] Research and Practice of SQL Optimization in ORACLE, Jiyuan Shi