Optimizing Concurrent M^3-Transactions: A Fuzzy Constraint Satisfaction Approach

Peng LI
Department of Computer Science, Western Kentucky University
Bowling Green, KY 42101, USA

and

Zhonghang XIA
Department of Computer Science, Western Kentucky University
Bowling Green, KY 42101, USA

and

I-Ling YEN
Department of Computer Science, University of Texas at Dallas
Richardson, TX 75080, USA

ABSTRACT

Due to the high connectivity and great convenience, many E-commerce application systems have a high transaction volume. Consequently, the system state changes rapidly and it is likely that customers issue transactions based on out-of-date state information. Thus, the potential of transaction abortion increases greatly. To address this problem, we proposed an M^3-transaction model. An M^3-transaction is a generalized transaction where users can issue their preferences in a request by specifying multiple criteria and optional data resources simultaneously within one transaction.

In this paper, we introduce the transaction grouping and group evaluation techniques. We consider evaluating a group of M^3-transactions arrived to the system within a short duration together. The system makes optimal decisions in allocating data to transactions to achieve better customer satisfaction and lower transaction failure rate. We apply the fuzzy constraint satisfaction approach for decision-making. We also conduct experimental studies to evaluate the performance of our approach. The results show that the M^3-transaction with group evaluation is more resilient to failure and yields much better performance than the traditional transaction model.

Keywords: Transaction Processing, Fuzzy constraint satisfaction, Criteria-based transaction, Group evaluation, Preference query

1 INTRODUCTION

Due to the high connectivity of the Internet, on-line transaction processing systems are anticipated to deal with highly concurrent accesses that involve multiple database systems. Recent researches in transaction processing have focused on efficient execution of such systems. Among the potential methods in improving performance for transaction processing, reducing transaction failure rate can be an effective approach. A major cause for transaction failure is due to the application logic in the transaction itself. For example, a transaction issued based on out-of-date information can cause its abortion.

Consider a typical on-line transaction flow. Usually, the user first browses the database to obtain the state information. Then she issues an update transaction to obtain the desired resources. If the state information becomes out-of-date when the update transaction arrives to the database, the transaction is likely to be aborted. Subsequently, a user may be forced to modify the query repeatedly, resulting in a low level of customer satisfaction and high communication cost. This problem can be exacerbated when the system has a high volume of concurrent accesses. Also, the impact of transaction failure becomes more significant when we consider widely distributed systems or access from mobile devices. In both cases, the high communication cost can significantly raise the request reissuing cost. In the mobile transactions cases, the probability of transaction aborts increases since the requests are more likely to have been issued based on the stale information cached on the client devices where caching is a common practice due to frequent disconnections.

Some approaches have been proposed to improve the potential problem with excessive transaction aborts. The basic idea is to put more information into a transaction to let a user express more sophisticated preferences. The flexible transaction model allows a user to define alternative sets of subtransactions [5]. If a more preferential choice cannot be committed, the next preferential one will be executed and, hence, it is less likely to abort. The preference query model proposed in [9] allows users to define multiple preferences in one query. All search conditions of a preference query is first evaluated, and then the results are ranked according to the preferences. The most preferred nonempty set of results is then returned to the client. The advantage of this mechanism is that the users can specify their intents more expressively and, hence, are released from reformulating successive requests for reaching their acceptable responses.

Putting more information into queries not only reduces the

chances of transaction aborts, but also offers the opportunities for global optimization. The system can consider a group of concurrent queries at the same time and try to satisfy as many of them as possible by looking at a global picture. Consider an example of ticket selling for a sports event. Assume that there is only one row, row *r*, with 5 consecutive seats in a section of the stadium for the event; while there are several rows with 3 consecutive seats. Customer *X* requests 5 consecutive seats and customer *Y* requests 3 consecutive seats. If *Y*'s transaction arrives first, then it is possible that the system assigns the seats in row *r* to *Y*, especially if *Y* puts row *r* as its first preference. Subsequently, when *X*'s transaction arrives, there will be no 5 consecutive seats available to satisfy its request. If transactions for customers *X* and *Y* arrive almost at the same time and the system can analyze them and make appropriate decision for the seat assignment, the transaction failure can be avoided and customer satisfaction can be maximized. However, to the best of our knowledge, there is no research work on optimizing concurrent update transactions.

In this paper, we proposed a new transaction model, M^3-Transaction (M^3 stands for independent Multi-relation, Multi-criterion, and Multi-quantification) as an integrated mechanism to enhance the conventional models for high volume on-line transaction processing. Multi-criterion construct allows users to specify preferences in a transaction to reduce transaction abort rate. Multi-quantification construct, IM-Top *m–n* (specifies that the first *L* tuples satisfying the criterion are to be output, $m \le L \le n$) facilitates users in specifying the desired cardinality of the target relation in a highly flexible way. Also, independent multi-relation construct provides expressive power and convenience for users to define additional source relations. With the ability of defining multiple preferences and alternate data sources, an M^3-Transaction is less likely to fail and can achieve better customer satisfaction. To further optimize customer satisfaction level and further reduce transaction abort rate, we consider group evaluation, i.e., consider evaluating multiple transactions at the same time. M^3-transaction facilitates the group evaluation since it supports criteria based selection and allows easy extraction of criteria.

In this paper, we discuss an optimization mechanism in processing M^3-transactions. Our approach can be applied to other criteria-based transaction models with preference specifications. In the next section, the decision-making mechanisms we use for the group evaluation will be discussed. In Section 3, we show the experimental results of our approach and compare different decision-making algorithms. Section 4 concludes the paper.

2 GROUP EVALUATION FOR M^3-TRANSACTIONS

Although M^3-transactions can be processed individually, group evaluation can further increase the customer satisfaction and reduce the transaction failure rate. We consider M^3-transactions that request for consumable resources in databases. A *consumable resource* is a data entry in a table that, once assigned to one transaction, cannot be assigned to another transaction. Clients may issue transactions with conflicting resources demands. In group evaluation, we need to assign resources in an optimized way that resolves most conflicts in client requests. We separate group evaluation for M^3-transactions into two steps, transaction grouping and resource assignment decision-making.

2.1 Transaction grouping

The rule for transaction grouping is based on the data set to be retrieved by concurrent transactions. A transaction, T_i, is assigned to a transaction group *TG*, if there is a transaction, T_j, in *TG* such that the data set retrieved for update by T_i is overlapping with that of T_j. Thus, the critical problem of grouping is how to determine the data set to be retrieved by each transaction. We adopt the approach in [8] to build approximation of data sets to be accessed by analyzing the predicates of criteria, i.e. use the search predicates to characterize the data sets. Then, whether the data sets are overlapping is estimated by determining the jointness of their search predicates. A naïve approach to analyze a criterion is to ignore all join predicates, just extract the search predicates and convert them into disjunctive normal form for efficient conflict testing. For example, in the criterion "Dept.budget > 10000 and Dept.dno = Employee.dno and Employee.age > 40", the "Dept.budget > 10000" and "Employee.age > 40" are extracted and the join predicate "Dept.dno = Employee.dno" is disregarded. Obviously, this criterion is joint with another criterion "Employee.age > 30". So the two transactions with them should be divided into one group. More sophisticated approaches can be found in the predicate lock literature [7] [4].

2.2 Mapping the M^3-transaction criteria to the FCSP

After concurrent transactions are divided into groups, the system will find an optimal resource assignment for each group. Consider a transaction group, *TG*, of *t* transactions, where $TG = \{T_1, T_2, \ldots, T_t\}$. Let MC_i denote the multi-criteria search condition of T_i, where MC_i contains p_i levels of preferences and $MC_i \equiv C_i^1$ **or-else** C_i^2 **or-else** … **or-else** $C_i^{p_i}$ (C_i^j is a preference/criterion specified in MC_i and C_i^j has a higher priority than C_i^k, $1 \le j < k \le p_i$). Also, assume that every transaction T_i in *TG* specifies a bounded quantity using the IM-Top *m–n* construct and m_i and n_i denote the lower and upper bounds respectively. Let SS_i denote the set of all possible solutions that can satisfy T_i, $SS_i = \{ S_i^{l_i,k_i} \mid S_i^{l_i,k_i}$ satisfies $C_i^{l_i} \wedge |D(S_i^{l_i,k_i})| = k_i \wedge m_i \le k_i \le n_i,\ 1 \le l_i \le p_i\}$, where $S_i^{l_i,k_i}$ is the solution that satisfies the l_i-th criterion, $C_i^{l_i}$, of T_i with a cardinality k_i (k_i tuples are assigned to T_i), $D(S_i^{l_i,k_i})$ is the set of data items involved in $S_i^{l_i,k_i}$ and $|D(S_i^{l_i,k_i})|$ returns its cardinality. We measure the satisfaction level of a solution $S_i^{l_i,k_i}$ to the transaction T_i by its satisfied criterion level l_i and its cardinality k_i. We say the solution S_i^{1,n_i} *fully* satisfies T_i since it satisfies the first level of criterion with n_i tuples. Other solutions $S_i^{l_i,k_i}$ ($\neq S_i^{1,n_i}$), $m_i \le k_i \le n_i$, $1 \le l_i \le p_i$, only *partially* satisfy T_i. In the extreme, $S_i^{p_i,m_i}$ provides the lowest satisfaction level to T_i. Obviously, due to the conflict resource demands and the limited resource availability, not all concurrent transactions can be fully satisfied; some of them have to be partially satisfied or even not satisfied at all. The group evaluation is to find a data assignment solution which can maximize the satisfaction level of all

transactions. More precisely, let $\mathbf{s} = \{s_i \mid 1 \le i \le t\}$ denote the final solution set for transaction group TG, where $s_i = S_i^{l_i^* k_i^*}$, and l_i^* and k_i^* are the level of the criterion and the cardinality selected in the final solution for T_i, respectively. So the group evaluation is to find an optimal solution $\mathbf{s}^*$ that minimizes l_i^* and maximizes k_i^* for each transaction T_i.

In order to guide the search for the optimal solution, we use the fuzzy constraint satisfaction problem (FCSP) approach to model the group evaluation. We first map the transaction space to a set of variables. Let $X = \{x_1, x_2, \ldots, x_t\}$ be the set of variables and each x_i represent a transaction T_i. SS_i (the set of solutions for T_i) is the domain for x_i, and s_i, where $s_i \in SS_i$, is the final solution for x_i. The requirement that no data item can be assigned to more than one transaction is expressed as a crisp constraint on X, i.e. $D(s_i) \cap D(s_j) = \phi$, for all i, j, $i \neq j$. We associate fuzzy constraints on each $x_i \in X$ to express the satisfaction level of the solution $s_i \in \mathbf{s}$ to each transaction T_i. Here we first describe the three objectives, O_1, O_2, and O_3, for group evaluation. We then define the corresponding fuzzy constraints for these objectives in the following subsections.

Objective O_1: Satisfy as many transactions within a group as possible. A solution a dominates another solution b if $TN(a) > TN(b)$, where $TN(x)$ returns the number of transactions satisfied in solution x.

Objective O_2: Satisfy as high a criterion level of a transaction as possible. Here, we use the leximin ordering approach [3] to define this objective. The leximin ordering approach combines the egalitarianism, which maximizes the minimal individual utility, and utilitarianism, which maximizes the sum of the individual utilities. It first compares the solutions according to egalitarianism, and then discriminates more among them via utilitarianism. Let $L(x) = \{l_i^x \mid 1 \le i \le t\}$, $1 \le l_i^x \le p_i$, denote the set of final criterion levels selected in solution x. We order the elements in $L(x)$ decreasingly to obtain a sorted list $SL(x) = (sl_1^x, sl_2^x, \ldots, sl_t^x)$ where $sl_i^x \ge sl_{i+1}^x$. A solution a dominates another solution b if $\exists i \, \forall j < i$, $sl_j^a = sl_j^b$ and $sl_i^a < sl_i^b$.

Objective O_3: Satisfy as high a cardinality requested by a transaction in its IM-Top statement as possible. Similar to O_2, we use the leximin ordering to define O_3. Let $N(x) = \{k_i^x \mid 1 \le i \le t\}$, $m_i \le k_i^x \le n_i$, denote the number of resources obtained by each transaction in the solution x. We order the elements in $N(x)$ increasingly to obtain a sorted list $SN(x) = (sn_1^x, sn_2^x, \ldots, sn_k^x)$ where $sn_i^x \le sn_{i+1}^x$. A solution a dominates another solution b if $\exists i \, \forall j < i$, $sn_j^a = sn_j^b$ and $sn_i^a > sn_i^b$.

2.2.1 Fuzzy modeling based on priorities of transactions

When data resources are scarce and not all transactions can be satisfied, some transactions need to be dropped during the evaluation. The dropping process is carried out according to the transaction priorities. We use a heuristic function Φ_D to determine the priorities of transactions and, then, decide which transactions are to be satisfied. Φ_D is defined as follows,

$$\Phi_D(x_i) = \frac{m_i \times conf(T_i)}{|D(T_i)| \times p_i}$$

where x_i is the variable representing the transaction T_i, m_i is the bottom bound of IM-Top function specified in the transaction T_i, $|D(T_i)|$ is the number of the data items available involved by T_i, $conf(T_i)$ is the conflict degree of T_i and p_i is the number of criteria in MC_i. The conflict degree is determined by the number of transactions, which is estimated by the predicate jointnesses between criteria of transactions, conflicting with T_i. The number of data available for each transaction can be estimated from the data state information maintained by the system. The rationale of Φ_D is that the more resources a transaction intends and the scarcer the resources are, the more likely the transaction cannot be satisfied. And the more a transaction conflicts with other transactions, the more data competition could be reduced when the transaction is dropped. Also, the fewer criteria MC_i contains, the less flexible the transaction is. Therefore, the higher the value $\Phi_D(x_i)$ is, the lower the priority will be assigned to transaction T_i. Let $Rank(\Phi_D(x_i)) \rightarrow r_i$, $1 \le r_i \le t$, denote the rank of $\Phi_D(x_i)$ where Rank 1 is the highest priority while Rank t is the lowest one. The priorities are uniformly mapped to $[b_1, b_2]$ by a function Φ_R as follows,

$$\Phi_R(x_i) = \frac{b_1 - b_2}{1 - t} * (r_i - t) + b_2, \; 1 \le r_i \le t, 0 \le b_2 < b_1 \le 1$$

where b_1 and b_2 are the upper and the lower priority bounds respectively.

We use the fuzzy model and prioritized constraints [2] to model the transaction priorities for O_1. Consider a crisp constraint, cc_i on each $x_i \in X$ to require that the transaction T_i should be satisfied. In the case that not all cc_i, $1 \le i \le t$, can be satisfied, some of them have to be dropped. To decide which one is preferred, we define the priority of each cc_i according to the priority of its corresponding transaction T_i. Let $P(cc_i)$ denote the priority of cc_i, where $P(cc_i) = \Phi_R(x_i)$. More specifically, $P(cc_i)$ indicates the level of importance that cc_i is satisfied. Now, the prioritized constraint $(cc_i, P(cc_i))$ can be represented by a fuzzy constraint pc_i such that,

$\mu_{pci}(s_i) = 1$ if a solution s_i satisfies cc_i

$= 1 - P(cc_i)$ if a solution s_i violates cc_i

Thus, the constraint cc_i is considered as satisfied at least to degree $1 - P(cc_i)$, no matter whether s_i satisfies it or not.

2.2.2 Fuzzy modeling based on transaction criteria

Here, we apply the fuzzy model to model O_2. A fuzzy constraint sc_i is associated to each variable $x_i \in X$ to express the satisfaction level of solutions in SS_i in terms of the selected criteria levels of T_i. Each sc_i is defined by a membership function μ_{sci} that maps each solution set $S_i^{l_i,*} = \{ S_i^{l_i,k_i} \mid S_i^{l_i,k_i} \in SS_i \wedge m_i \le k_i \le n_i \}$ to a membership value in [0, 1], where $S_i^{l_i,*}$ is a subset of SS_i and all its members satisfy $C_i^{l_i}$. The membership function μ_{sci} is defined as follows,

$$\mu_{sc_i}(S_i^{l_i,*}) = l_i^{-\frac{p_i}{a_1 * p_i - 1}}, \quad 1 \le l_i \le p_i$$

where a_1 is a constant. The rationale of constructing μ_{sci} as a power function is that we assume the shape of the customer satisfaction curve for his preferences follows the curve of a power function, i.e. a customer may consider that the satisfaction degree of the second preference is much less than the first one, but the satisfaction degree of the 10th preference has no big difference with that of the 9th one. We also assume that the more preference levels an MC_i contains, the less will be the differences between satisfaction levels. Thus, we use p_i, the number of criteria in the MC_i, as a parameter in μ_{sci} for this purpose. To further control the drop rate of the curve, we

introduce a_1 as a factor which can be adjusted by the user.

2.2.3 Fuzzy modeling based on requested cardinalities

Similar to the fuzzy modeling for criteria, we apply the same approach to the requested cardinalities. A fuzzy constraint rc_i is associated to each $x_i \in X$ to express the satisfaction levels of the solutions in SS_i in terms of the requested cardinality of T_i. Each rc_i is defined by a membership function μ_{rci} that maps each solution set $S_i^{*,k_i} = \{ S_i^{l_i,k_i} \mid S_i^{l_i,k_i} \in SS_i \wedge 1 \le l_i \le p_i\}$ to a satisfaction level in a linearly ordered set K (with top denoted 1 and bottom denoted 0), where S_i^{*,k_i} is the subset of SS_i and the cardinality of its members is k_i. Similar to μ_{sci}, we define the membership function μ_{rci} as follows,

$$\mu_{rc_i}(S_i^{*,k_i}) = (n_i - k_i + 1)^{\frac{m_i - n_i - 1}{a_2 * (n_i - m_i + 1) - 1}},\ m_i \le l_i \le n_i$$

where a_2 is a constant and is to control the curve drop rate.

Note that the FCSP is known to be NP-complete and its computational cost is in proportion to the scale number of the valuation set [2]. Regarding the requested cardinalities, the big potential difference between n_i and m_i can make the number of different scales in the valuation set very big, resulting in high computation cost for group evaluation. To control its cost, we further map the valuation set K to another set F with coarser scales as follows,

$$\Phi_F(\mu_{rc_i}(S_i^{*,k_i})) = \frac{i}{F} \quad \text{for } \forall i, \text{ s.t. } if\ \frac{i}{F} - \mu_{rc_i}(S_i^{*,k_i}) < \frac{1}{2 * F}$$

$$or\ \mu_{rc_i}(S_i^{*,k_i}) - \frac{i}{F} \le \frac{1}{2 * F}, 0 \le i \le F$$

In this way, each value in K is rounded to a scale in F which is closest to it.

2.2.4 Decision making process

In order to reduce the search space for group evaluation, we divide the search process into two steps. In the first step, we fix the request cardinalities at the lowest level and search for an optimal group solution based on the fuzzy modeling for the criteria and priorities of transactions. In the second step, we fix the criteria selections obtained in the first step and find optimal cardinalities for all transactions based on the fuzzy modeling on requested cardinalities.

We first use the prioritized fuzzy constraints [2] to integrate the two fuzzy constraints, for multiple criteria and priorities of transactions, together. That is, we define the priority for each fuzzy constraint sc_i by applying the approach proposed in Sub-section 2.2.1 and then attach the priority P(sc_i) computed to each sc_i. This process is modeled by the prioritized fuzzy constraint fc_i as follows.

$$\mu_{fci}(S_i^{l_i,*}) = \max(1 - P(sc_i), \mu_{sci}(S_i^{l_i,*}))$$

The meaning of $\mu_{fci}(S_i^{l_i,*})$ is that the satisfaction level for a solution set $S_i^{l_i,*}$ is at least 1− P(sc_i), regardless of to what extent it satisfies the criteria of T_i. Essentially, the fuzzy constraint set, $\{fc_i \mid 1 \le i \le t\}$, models O_1 and O_2 together.

We now define the objective function for the first step. The satisfaction level of a group solution $\boldsymbol{s} = \{s_i \mid 1 \le i \le t\}$ to a transaction group TG is denoted by $Sat_{step1}(\boldsymbol{s})$ and measured by the satisfaction degree of the least satisfied fuzzy constraint, i.e. $Sat_{step1}(\boldsymbol{s}) = \min_{1 \le i \le t}(\mu_{fci}(S_i^{l_i,k_i}))$, where k_i is a cardinality with the lowest satisfaction level for T_i in F, i.e. $\neg\exists\ j_i$ s.t. $0 < \Phi_F(\mu_{rc_i}(S_i^{*,j_i})) < \Phi_F(\mu_{rc_i}(S_i^{*,k_i}))$. Thus, the optimal group solution $\boldsymbol{s}^*$ obtained in this step is a solution that maximizes the satisfaction level of the least satisfied constraint, i.e. $Sat_{step1}(\boldsymbol{s}^*) = \max_{\boldsymbol{s}} \{\min_{1 \le i \le t}(\mu_{fci}(S_i^{l_i,k_i}))\}$.

To search $\boldsymbol{s}^*$, we use the algorithm proposed in [2], which is based on the Depth-First Branch and Bound approach. The search space of this algorithm is a tree where the root is an empty assignment, internal nodes denote partial solutions of variables and leaves are the complete ones. During the search, variables are instantiated one by one. Solutions for each variable x_i are incrementally computed based on the state information and indexed for efficient retrieving. For each solution $\boldsymbol{s}$ represented by a leaf, its $Sat_{step1}(\boldsymbol{s})$ is computed. The current best $Sat_{step1}(\boldsymbol{s})$ is used as a lower bound, b_{low}. In addition, $Sat_{step1}(\boldsymbol{s}_{partial})$ of the partial solution $\boldsymbol{s}_{partial}$ denoted by the current internal node is computed and compared with b_{low}. When $Sat_{step1}(\boldsymbol{s}_{partial}) \le b_{low}$, the branches with the current node as root are pruned.

Based on the optimal solution obtained in the first step, denoted by $\boldsymbol{s}_{step1}^*$, we further try to increase the satisfaction level in terms of cardinalities of the transactions, while abiding to the criterion levels determined in $\boldsymbol{s}_{step1}^*$. We first define the objective function for the second step. The satisfaction level of a group solution $\boldsymbol{s}$ is measured by $Sat_{step2}(\boldsymbol{s}) = \min_{1 \le i \le t}(\Phi_F(\mu_{rci}(S_i^{l_i^*,k_i}))$, where l_i^* is the criterion level determined by $\boldsymbol{s}_{step1}^*$ and $m_i \le k_i \le n_i$. Thus, the optimal group solution $\boldsymbol{s}^*$ obtained in this step is a solution with the maximal $Sat_{step2}(\boldsymbol{s})$, i.e. $Sat_{step2}(\boldsymbol{s}^*) = \max_{\boldsymbol{s}} \{\min_{1 \le i \le t}(\Phi_F(\mu_{rci}(S_i^{l_i^*,k_i})))\}$. The search algorithm of this step is the same as that of the first one.

2.3 The Leximin ordering approach

The min-optimal approach discussed in Section 2.2.4 uses the egalitarianism approach that only takes into account the satisfaction level of the least satisfied transaction and does not consider the satisfaction level of other transactions in the group and, hence, the group solutions generated are coarse. Here, we adopt the *leximin Ordering* (LO) [3] to refine min-optimal solutions. Let $SD(\boldsymbol{s}) = \{sd_i^s \mid sd_i^s = \mu_{fci}(s_i), 1 \le i \le t\}$ denote the set of satisfaction degrees of s_i in the solution $\boldsymbol{s}$ generated in the first step. We order the elements in $SD(\boldsymbol{s})$ in an increasing order to obtain a sorted vector $SSD(\boldsymbol{s}) = (ssd_1^s, \ldots, ssd_t^s)$ where $ssd_i^s \le ssd_{i+1}^s$. A solution a dominates another solution b, denoted as $a \ge_{lex} b$, if $\exists i\ \forall\ j < i\ ssd_j^a = ssd_j^b$ and $ssd_i^a > ssd_i^b$. The solution refinement for the second step is the same. To search a leximin-optimal solution, we use the algorithm proposed in [11], which is also based on the Depth-First Branch and Bound approach. In this algorithm, the leximin ordering is used to compare the satisfaction degree vectors of the partial or complete solutions. The satisfaction degree vector of the current best solution is taken as the lower bound, b_{low}. The satisfaction degree vector of

the current one, denoted as SD_{cur}, is used as an overestimation of the maximum vector of satisfaction degrees among leaves descending from the current node. When $SD_{cur} \leq_{lex} b_{low}$, pruning occurs.

Note that the computational cost of the leximin optimal problem is much higher than the corresponding FCSP because its computational cost is in proportion to the number of possible different leximin vectors, which grows exponentially with both the number of constraints in the FCSP and the number of scales in the valuation set [6]. Therefore, in order to guarantee the response time when the system load is high, suboptimal solutions have to be considered, e.g. setup a time out and seek the best suboptimal solution within the time bound or set an adaptive upper bound to stop the search earlier.

3 SIMULATION RESULTS

We have conducted preliminary experiments to evaluate the performance of the decision making algorithms for the M^3-transaction and compare it with the traditional transaction. The simulation was conducted on a single database system. We choose the example of tickets selling for our simulation. The sports stadium consists of 50,000 seats, divided into 25 sections. Each section consists of 100 rows with 20 seats in each row. The information for the stadium and all the seats are stored in an Oracle 9i database. The system consists of one server and multiple clients. Clients communicate with the server using the HTTP protocol and the server is programmed in Java and interacts with the database through JDBC. The clients are simulated by a client program, which issues synthetic M^3-transactions simulating multiple clients. The inter-arrival rate of the transactions generated by the client program follows the Poisson distribution.

The client program keeps a view of the database and refreshes it periodically. This is to simulate the scenario that the clients may get out-of-date information due to the large number of concurrent accesses or data caching. The time between two successive refreshes is called "View Refresh Time". Clients issue transactions based on their current view of the database. Each M^3-transaction is specified by the following parameters: total number of seats desired, number of consecutive seats desired among them, section preference (indicates the preferred range of sections), and row preference (indicates the preferred range of rows). The number of seats and number of consecutive seats desired are randomly selected from a range based on a uniform distribution. The 25 sections in the stadium are divided into three segments to simulate the price-based seat selections. The desired segment in a transaction is generated randomly. To generate the row range preferences, a row and the corresponding section within a segment are chosen such that they have a possibility of satisfying the seating requirement and are as close to the front as possible. This is done to ensure that the regions at the front have a higher chance of being selected than the ones at the back, which simulates the real world behavior of a client preferring a front row to the one at the back.

We designed two experimental setups to evaluate the performance of the M^3-transaction model and compare it with that of the traditional transactions.

Clients Send Traditional Transactions. This setup simulates the traditional transaction processing systems. First, the client generates a traditional transaction that obtains the most preferred set of data items based on the state of the database (which may be out-of-date). If the transaction fails, the client generates another transaction to obtain an alternative set of data based on the user preference and the state of the database (which may have been renewed). This continues till one transaction succeeds. Note that with each time of the new transaction creation, the number of consecutive seats is relaxed by one until no requirement for the consecutive seats.

Clients Send M^3-transactions. In this setup, the client first generates the multi-criteria selection condition, *MC*, based on the state of the database to specify the client's alternative preferences. Then, the corresponding M^3-transaction (with selection condition *MC*) is sent directly to the server. The server processes the M^3-transactions using the FCSP approach and the LO approach. If an M^3-transaction fails, the client will generate new M^3-transactions iteratively until one succeeds.

For the second setup, we only collect transactions when the number of newly arrived transactions is larger than a trigger value. The parameters of the mapping functions are set as follows, $b_1 = 0.8$ and $b_2 = 0.2$ for Φ_R, $a_1 = 3/2$ for μ_{sci}, $a_2 = 3/2$ for μ_{rci} and $F = 5$ for Φ_F. To improve the performance of the FCSP and the LO algorithms, we integrate the simple BackJumping algorithm [1] into the Branch and Bound search and use some heuristic to guide the search process. The heuristic prunes inferior choices and selects transactions with lower priorities and solutions with the highest satisfaction degree . Also, we set up an adaptive upper bound for the search.

The performance metric used for the purpose of comparison is the average response time *T*. The response time of a request is the total processing times of all transactions issued for the request, including the failed transactions and the last transaction that obtains the desired data items successfully. Figure 1 shows the experimental results and compares the performance of traditional transaction processing and M^3-transactions processing at various transaction arrival rates, with view refresh times of 500 msec and 1000 msec (Figure 2). We set the collection trigger value to 2 when the arrival rate is less than 6 ps, 3 when the arrival rate is 6-8 ps, 4 when the arrival rate is 10-14 ps, and 6 when the arrival rate is 16 ps or higher. There is an exponential increase in *T* for traditional transactions. This could be attributed to the fact that at higher transaction arrival rates, more clients are likely to see the same view of the database and thus potentially issue requests with intersecting choice sets, increasing the failure rate of transactions. Also from the figure, we can see that the higher the view refresh times, the larger the values of *T*. The reason for this is that as view refresh time increases, more clients issue transactions based on a staler view of the database and, hence, increasing the likelihood of failure. From the experimental results, we can see that M^3-transactions (processed by the FCSP approach) are more resilient to failure and yield much better performance.

On the other hand, from the figures, we can see that there are sudden increases in *T* for group evaluation based on the LO approach. This is due to the high computation cost incurred in this approach. When the transaction arrival rate increases to a

certain level (30 ps with the view refresh time 500 msec and 26 ps with view refresh times 1000 msec), the number of transactions collected in a transaction group increases, to a certain extent (about 11 in this study), and, hence, the time needed for decision making becomes very long. As a result, many transactions are forced to wait for the decision making process. After several rounds, hundreds of transactions may be suspended and, hence, overloading the system. However, the experimental results (not shown in the figures) show that the LO approach yields limited improvements to the solutions obtained from the FCSP approach (only 0.2% - 0.6% transaction groups can be improved). This could be attributed to the heuristic which always tries to select the solution with the highest satisfaction level. So, in this study, the FCSP method is much more efficient than the LO approach, and it can obtain high quality solutions when appropriate heuristic is used.

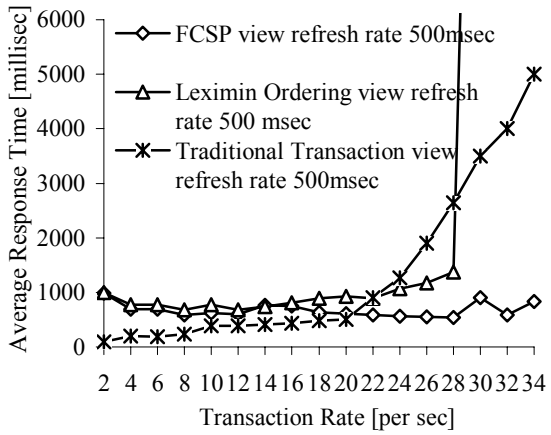

Figure 1. performance comparison (500 msec refresh rate)

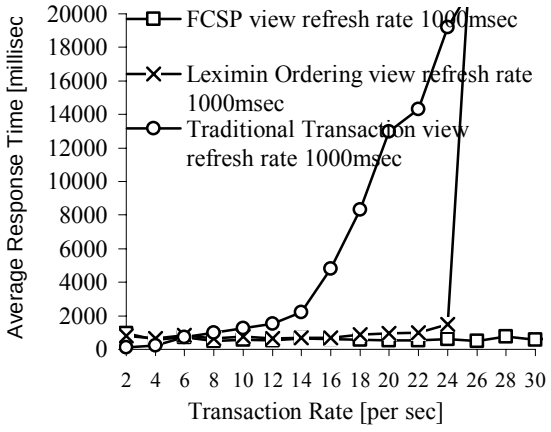

Figure 2. Performance comparison (1000 msec refresh rate)

We have also conducted a simulation to further measure the performance of the decision making process. We set an upper bound for group size. If the number of transactions in a group exceeds the given bound, the system divides the group into smaller groups. Figure 3 shows the average decision time incurred with various group-size bounds, with arrival rate of 16 ps, view refresh times of 500 msec, and the collection trigger values of 20 and 30. Figure 4 shows the average response time of the same simulation. The results show that the average decision time increases exponentially as the maximum group size increases. The average response time is dominated by the collection delays when the group size is less than 9 for LO and 14 for FCSP. But when the maximum group size increases, the average response time increases exponentially due to the significantly increased decision time. So, from this simulation, we can see that the system should adjust the collection trigger value dynamically according to the system loads since the collection delays affect the response time significantly. Also, the maximum group size should be restricted even though it implies decreased degree in optimization.

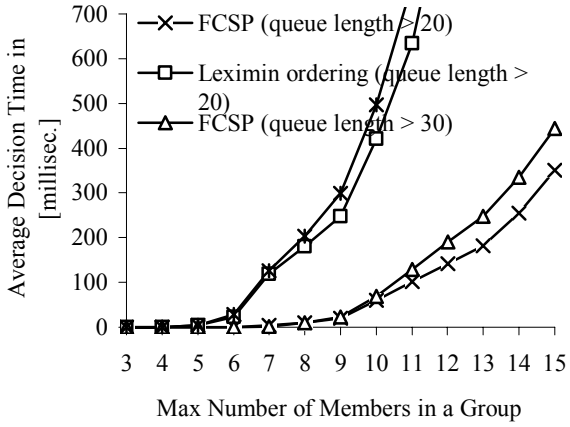

Figure 3. Decision Time comparison

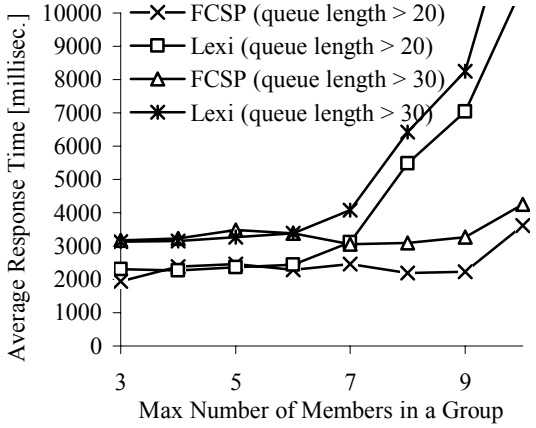

Figure 4. Respone time comparison

4 SUMMARY AND FUTURE WORK

In our previous work [10], we have introduced the M^3-transaction model which can reduce transaction failure rate and increase the customer satisfaction. An M^3-transaction allows customers to specify their preferences in advance and, hence, avoids iteratively submitting transactions in the case of failure.

In this paper, to increase the customer satisfaction and further reduce transaction failure rate, we propose the group evaluation technique for the M^3-transaction. We consider a group of M^3-transactions that arrive to the system within a short duration together and decide the optimal data allocation for them. The decision-making mechanism is based on the

FCSP model. The experimental results show that the M^3-transactions are more resilient to failure and yield much better performance. The results also show that the FCSP method is efficient and can achieve solutions with high quality when appropriate heuristic is used.

Future work for the M^3-transaction research is considered in two directions. First, the techniques for searching solutions for each criterion only basing on the database state information need to be further studied. Second, we will extend the group evaluation for the multidatabase system.

5 REFERENCES

[1] E. Tsang, "Foundation of constraint satisfaction", **Academic Press**, 1993.

[2] D. Dubois, H. Fargier and H. Prade, "Possibility theory in constraint satisfaction problems: handling priority, preference and uncertainty", **Applied Intelligence**, Vol. 6, 1996, pp. 287-309.

[3] D. Dubois, H. Fargier and H. Prade, "Refinements of the maxmin approach to decision making in a fuzzy environment", **Fuzzy Sets and System**, Vol. 81, 1996, pp. 103-122.

[4] C. Elkan, "A decision procedure for conjunctive query disjointness", **Proc. of the 8th ACM Symp. on Principles of Database Systems**, 1989, pp. 134--139.

[5] A.K. Elmagarmid, Y. Leu, W. Litwin, and M. Rusinkiewicz, "A multidatabase transaction model for interbase", **Proc.16th Intl. Conf. on VLDB**, 1990, pp. 507-518.

[6] M.P. Joao, M.P. Fernando, A.R. Rira, "Structure and properties of leximin FCSP and its influence on optimization algorithms", **Proc. of the Intl. Conf. on IPMU**, July 1998, pp. 188-194.

[7] A. Klug, "Locking expressions for increased database concurrency", **Journal of the ACM**, Vol. 30, No. 1, January 1983, pp. 36-54.

[8] W. Schaad, H.-J. Schek, G. Weikum, "Implementation and performance of multi-level transaction management in a multidatabase environment", **RIDE**, Taipeh, 1995.

[9] M. Lacroix, P. Lavency, "Preferences: putting more knowledge into queries", **Proc.13th Intl. Conf. on VLDB** 1987, pp. 217-225.

[10] P. Li, I. Yen, "M^3-transaction: a new transaction model for on-line applications with high access rate", Dept. of Computer Science, the Univ. of Texas at Dallas, **Technical Report No. UTDCS-22-02**, Oct. 2002.

[11]P. Meseguer, J. Larrosa, "Solving fuzzy constraint satisfaction problems", **Proc. of the 6th IEEE Intl. Conf. on Fuzzy Systems**, Vol. 3, 1997.