REDUCING DISRUPTION IN TIME-TABLED CONDITION MONITORING

Binling Jin and Suzanne M. Embury
Department of Computer Science,
University of Manchester,
Oxford Road, Manchester M13 9PL
United Kingdom
{bjin, sembury}@cs.man.ac.uk

Abstract Condition monitoring typically has a high processing overhead and can thus disrupt the processing of business transactions by the monitored systems. Time-tabled condition monitoring aims to overcome this disadvantage by allowing the system owners to specify the times at which condition monitoring may take place, and the times when the system is too busy with revenue generating (or otherwise mission critical) processing. Unfortunately, however, when this approach is applied to a distributed system, the complex interactions between the component sites during monitoring mean that some interference with normal business processing does occur. In this paper, we present five methods for reducing this interference at the expense of the accuracy of the results of condition monitoring. We describe the outcome of an experimental evaluation that has helped us to characterise each method in terms of its effect on disruption and accuracy. We conclude by presenting heuristics to help in choosing which method to adopt in a given situation.

Keywords: Condition monitoring, distributed query evaluation, integrity monitoring.

Introduction

Organisations are becoming increasingly dependent on their information systems (ISs) in supporting many key business functions. In addition to the traditional uses of information in facilitating and guiding the day-to-day operations of the business, it is also being applied in other contexts, such as marketing, customer relationship management and strategic decision making. However, access to information comes with an associated cost. Execution of queries and transactions (especially in high volumes) can require non-negligible amounts of time. Be-

cause of this, the number and variety of these longer term applications of data that can be supported by a company's ISs are limited by the amount of processing resources that can be spared from the task of managing mission critical/revenue generating business processes.

One solution to this problem is to replicate the data in a new information system (e.g. a data warehouse [1]) but this is expensive, both in terms of the initial set-up and the continued maintenance of the replicated data. Few applications will warrant the expense of full replication, even though they may be very worthwhile and of potential value to the organisation.

However, less drastic solutions are also possible. The owner of an IS may be reluctant to allow unlimited access by a new application (since it may interfere with the rate at which revenue generating transactions can be processed, for example). However, he or she may be more willing to release some processing resources, on the proviso that control over exactly when this happens and for how long remains in his or her hands. For example, the manager of a busy IS may be very reluctant to take on additional processing burdens during office hours, but may be happy to allow a new application access to the data between the hours of 3.00am and 5.00am, once the business of the day has been concluded and the backup procedures have run to completion.

Time-tabled condition monitoring (TTCM) is a method of tracking the state of conditions expressed over a distributed information system (DIS) that attempts to make use of these less-heavily-loaded time periods. It allows the owner of each component system (or portion of data) to specify when that system is available for processing queries relating to condition monitoring. Effectively, the system owner provides a timetable (of arbitrary length) that states when the system is busy with other, more important tasks, and when it is free to undertake work for the TTCM engine. The aim is to allow the long term monitoring of conditions over distributed systems without also disrupting the normal business processing rates at the component ISs [11].[1]

The price to be paid for the resulting lack of disruption is that the record of the data items that satisfy the condition over time will not be accurate. Since the TTCM engine must wait until the component systems are free for use, some changes in conditions will not be detected until after they have occurred. If a change is short-lived, it may not be

[1] Although at present there is little demand for distributed condition monitoring in industry, we have focused on distribution as this is the more general case, with centralised systems being the special case. Moreover, with the recent rise of data sharing technologies such as the Grid, we believe that the need to monitor conditions that span several distinct systems will grow.

detected at all. At one end of this spectrum, we have the situation where the owner is so concerned about disruption to the system that he or she forbids all condition monitoring. Naturally, the levels of accuracy in this case will be very poor. Similarly, we can achieve extremely accurate results if we are not concerned about how much disruption we cause to other forms of information processing on the system in question (for example, monitoring the conditions as soon as updates occur, using an active rule mechanism).

Ideally, of course, we would like to maximise the accuracy of the results while also minimising the amount of disruption that occurs at each site. In our previous work, we have explored the use of temporal logic techniques for increasing accuracy, and have characterised them according to the degree of disruption they impose on the DIS [12]. However, the ability to trade disruption for better accuracy is only half of the story. In order to allow full control over the TTCM system, we also need to provide the system administrator with ways to reduce disruption at the expense of poorer accuracy. It is this latter form of control that we turn our attention in this paper.

The remainder of the paper is organised as follows. In Section 1, we discuss the various approaches to condition monitoring proposed in the literature and the means by which researchers have attempted to limit the processing overheads they impose. Next, in Section 2, we describe the TTCM approach in more detail and pinpoint the causes of disruption and inaccuracy within it. Section 3 presents the five different methods we have proposed for reducing disruption, and characterises them in terms of their expected effects on both disruption and accuracy. The results of an experimental evaluation of the five methods are discussed in Section 4, while Section 5 concludes.

1. Approaches to Condition Monitoring

The term *condition monitoring* refers to the continuous scrutiny of an IS in order to discover whether any data items exist within it that satisfy a particular condition. In some cases, the purpose of this scrutiny is to raise a warning when some undesirable or illegal condition has arisen (for example, monitoring of medical equipment). In other cases, the properties of data items which satisfy the condition are recorded, along with the times at which they were entered into and removed from the IS, in order to analyse long term trends (for example, in fraud detection and quality control). In this paper, we are concerned with this second kind of condition monitoring.

Experience has shown that condition monitoring can be highly expensive in terms of the processing resources it demands. Since the monitoring is *continuous,* we must re-evaluate the query that searches for matches with the condition every time a (relevant) update occurs. To make matters worse, since updates happen most frequently during periods of peak business activity, condition monitoring imposes its worst overheads at times when processing resources can least be spared.

A number of researchers have studied the problem of reducing the overheads of condition monitoring. The most common approach is to evaluate the condition query incrementally relative to each update that occurs, rather than repeating the work of checking data items that cannot possibly have been affected by the change [4, 14, 16, 19, 20]. Another common approach focuses on avoiding unnecessary recomputation of unchanged parts of the condition view by materialising it (either in whole or in part) [2, 5, 8, 17, 18].

Efficiency is an even greater concern when the condition to be monitored is distributed across several databases, because of the significant overheads imposed by the need to ship data sets between sites. Most proposals for improvements to distributed condition monitoring have therefore focussed on reducing the costs of data communications. For example, Mazumdar [13] proposed a technique for reducing the number of sites involved in the distributed query evaluation, on the grounds that this would also reduce the need for data shipping. Other authors (e.g. [10, 9, 15]) have focused on the more fundamental problem of how to keep the data sets that must be communicated between sites small and manageable.

All the methods described above assume that accuracy of condition monitoring is paramount and therefore that conditions should be checked immediately when data is updated (or transactions committed). Even taking into account the efficiency measures we have mentioned, the overhead imposed by immediate condition monitoring is too great for many system owners to feel that the benefits are worth the resultant costs. In some cases, however, we may be prepared to accept some limited reduction in accuracy, if by that means we can reduce the disruption to key business functions. A small number of authors have considered this line of argument, and have proposed a periodic approach to condition monitoring, in which condition re-evaluation is not triggered by each and every update [3, 6, 7]. Instead, the conditions may be monitored at some frequency set by the user (e.g. every hour or after every 100 transactions) or perhaps only when the user requests the current value.

These methods provide system owners with some measure of control over the resources that are given over to condition monitoring, but

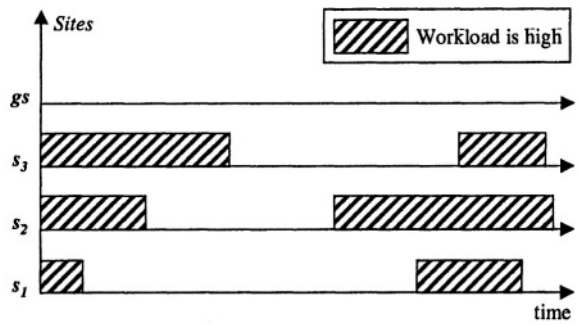

Figure 1. **An Example Set of Timetables Showing when TTCM may Operate**

only in a rather coarse-grained fashion. Moreover, they do not consider the additional problems raised by distributed condition monitoring, where each component site imposes its own constraints on resource usage. Time-tabled condition monitoring (TTCM) takes the notion of periodic monitoring a stage further, in allowing the owner of each site involved in distributed condition monitoring to specify exactly when that system can perform work on behalf of the TTCM system and for how long [11]. The TTCM system then attempts to choose an optimal plan for distributing the work of query evaluation to the component sites, in such a way that accuracy of the results will be maximised. However, because of the complex interactions between the timetables specified by the individual sites, it is not always possible to avoid all disruption at all sites. In the following section, we will present a more detailed description of the workings of TTCM and outline the causes of this unwanted disruption.

2. Time-Tabled Condition Monitoring

Figure 1 shows example timetables for a DIS consisting of three component sites (s_1, s_2 and s_3). A fourth site (gs) is also shown, representing the TTCM system itself (where the results of condition monitoring are recorded). The blank periods at each site indicate times when condition monitoring queries may be evaluated by that site, while the shaded areas indicate periods when the system is busy with other work.

When a query is submitted to the TTCM system for monitoring, it is translated into an execution plan, which describes how the work of each of the operators within the query are to be allocated across the various sites of the system. By implication, therefore, the execution plan also determines which data sets will have to be shipped between sites for con-

dition monitoring to take place. Each site maintains an operation queue, which indicates the operations that are due for re-evaluation at that site. Operations are ordered within the queue according to decreasing length of time between their last evaluation and the most recent update to their input relations. As well as the usual relational algebra operations, operation queues may also include ship operations (which request that a local relation be transmitted to another site) and load operations (which request that data shipped from a remote site be loaded into a local relation). The final results of condition re-evaluation are sent to the TTCM site, as a set of timestamped tuples, where they are recorded for future analysis.

The reasons for the inaccuracy of the results from TTCM should now be clear. Data items which satisfy the condition will often be introduced into the IS during the periods when it is busy with normal business processing (i.e. when update rates are high). The TTCM system, however, must wait until the next free period before it can begin the work that will detect the satisfaction of the condition. Therefore, the timestamp that will be associated with the new data item will be later than the time of its actual entry into the system. Similarly, when updates to a data item mean that it no longer satisfies the monitored condition, the timestamp recorded by TTCM for its removal will also be inaccurate. It is even possible that the presence of data items that satisfy the condition may be missed altogether, if they are removed again before the TTCM system has a chance to identify them. In our previous work [12], we have shown that it is possible to reduce these inaccuracies by the inclusion of temporal reasoning in the query evaluation process, but some inaccuracy will still remain.

The causes of disruption, on the other hand, are less obvious to the casual observer. By disruption in this context we mean a reduction in the amount of non-condition monitoring work that the component sites are able to undertake while TTCM is in operation. If the timetables for TTCM have been set accurately by the system owners, and very little business processing occurs during the periods marked as "free" for TTCM, then we would expect TTCM to have no negative effect on the transaction rates exhibited by the monitored system. Yet our experiences have shown that such disruption does occur.

Why should this be so? On investigation, we discovered that the problem arises due to the complex interactions between the component systems, each of which may be operating to a different timetable for the purposes of TTCM. We can distinguish two significant forms of disruption:

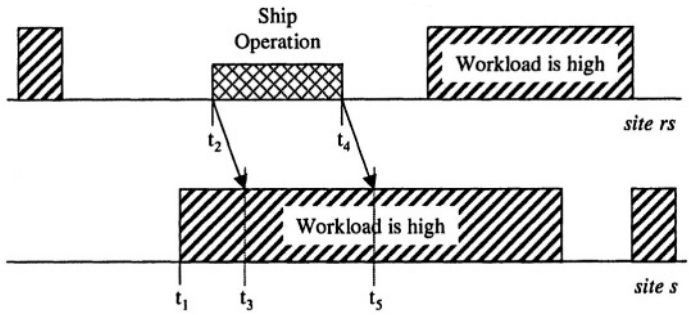

Figure 2. Disruption Caused by Remote TTCM Processing During Busy Periods

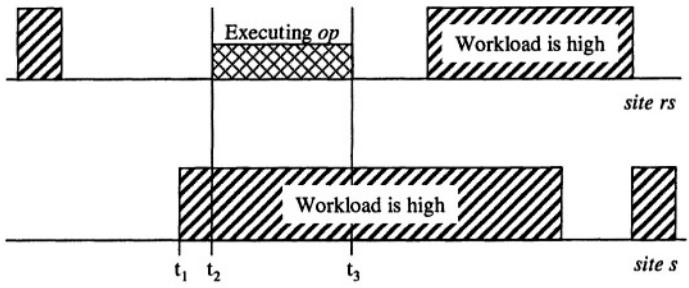

Figure 3. Disruption Caused by Delays in Processing Requests for Remote Data

- *Direct Disruption,* which is caused when a ship operation arrives at its destination site during a busy period for that site (Figure 2). Although the receiving site would not attempt to load the shipped data into the database (since TTCM is not enabled during busy periods), there is still some small amount of overhead involved in handling the receipt of the data and storing it locally on disk for later processing.

- *Indirect Disruption,* which is caused when a component system that is performing normal business processing (i.e. is in a busy period) needs to access data held at another site that is currently engaged in TTCM. In such a case, the processing of the transaction at the busy site will be delayed until the site which holds the data completes its current TTCM operation (Figure 3).

3. Methods for Reducing Disruption

One way to combat the disruption inherent in TTCM is to make a careful choice of the allocation of work to the various sites. However, such an approach cannot adapt to the local conditions that occur at runtime. The alternative is to modify the way in which TTCM operations are executed by each individual site, so that each site has available the complete context in which to make decisions about the best way to avoid disruption. We will now present five run-time strategies for reducing disruption in this way, and will discuss how far each of them can help to avoid disruption and what effect they each have on the accuracy of the final TTCM results.

Method 1: Ship Data Only When Both Sites are Free. The first method is aimed at reducing direct disruption. As we have seen, this form of disruption results when data is shipped to a site at which the workload is high. This suggests the simple expedient of delaying ship operations until the destination site has entered a free period and is ready to receive the data. Where this can be done, the direct disruption should be completely eliminated.

The main advantages of this approach is that it is very simple and straightforward to implement. The only information required to determine whether to ship data or not is the timetable of the destination site, which can be cached locally for easy access. The main disadvantage is the effect on accuracy, which is likely to be significant unless both sites involved in the ship operation have many free periods in their timetables. If they do not, then there is a chance that the shipment of data may have to be delayed for a very long time until the timetables of the sending and the receiving site happen to coincide.

Method 2: Ship Data via a Free Intermediate Site. The second method is a variant on the first. It aims to reduce delays in shipping data while still eliminating the direct disruption. As with the first method, method 2 begins by examining the timetables of the sending and receiving sites and calculating the next time at which both will be free. However, it also looks for a third site that can potentially be used as an intermediate holding site for the data while waiting for the destination site to become free. Once again, data is only shipped when both sites involved are free for TTCM. The difference this time is that we have two ship operations to consider: the transfer from the sending site to the intermediate site, and the transfer from the intermediate site to the destination site. If the use of an intermediate site will cause the

data to arrive at the destination site sooner than with method 1, then the necessary ship operations are inserted into the operation queues at the source site and the intermediate sites.

Since delays in evaluation are the cause of inaccuracy, where there are several candidate intermediate sites, we choose the one that will result in the earliest arrival time for the data at the destination site. In a complex system, with very few free periods, it might also be worthwhile to consider the use of multiple intermediate sites, although we have not explored this option ourselves (since the search space size grows rapidly with the number of hops considered and the anticipated benefits are not all that great).

There is also the added complication that, using this method, it is possible that an older version of the relation might arrive at the destination site *after* the arrival of a newer version (which has travelled by a different route). However, this problem can be solved by shipping timestamps with each version of the relation, so that the destination site can check whether the data that has just arrived is more recent than that received previously or not.

The advantage of this method is that it should result in more accurate condition monitoring than method 1, while having much the same effect on disruption. The disadvantage is that it is slightly more complicated to implement.

Method 3: Monitor Conditions Periodically. This third method aims to reduce indirect disruption by cutting down on the number of times that condition monitoring operations are performed during TTCM. The rationale for this is that if the TTCM is issuing fewer queries then there will be more bandwidth available for handling requests for data from other (busy) sites.

Ideally, we would prevent only those executions of operations that do not result in any new information regarding the data items that satisfy the condition. For example, suppose during a free period at site s_1, the relation r is shipped twice to the destination site s_2. (This could happen if r was updated just after the first ship operation had been completed, for example.) However, at the time of the first ship, s_2 is busy and it does not process the new set of tuples for r until after the second ship operation has occurred. In this case, the first ship operation was a waste of time, because its results were not used by the TTCM system.

Unfortunately, identifying such redundant operations before they occur is impossible because we do not know when the next update to the input relations of each operation will occur. We must therefore adopt some more artificial mechanism for determining how often each oper-

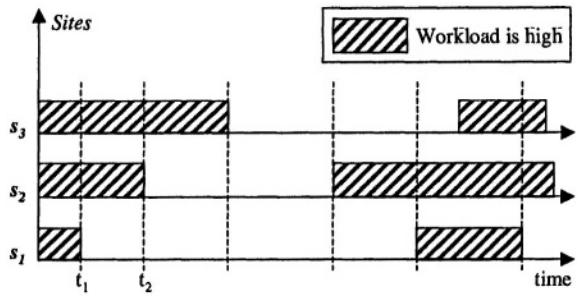

Figure 4. **Example Timetables Showing the Points at which Re-evaluation of Operations is Allowed**

ation can productively be re-evaluated. Since the majority of updates occur during busy periods, one approach is to re-evaluate each operation only when some site in the system has changed its status from busy to free. However, we also need to take into account changes that are made as a result of TTCM itself, so we should also be prepared to re-evaluate operations when a site changes from free back to busy again.

This strategy is best explained by an example. Figure 4 shows the timetables of three sites involved in condition monitoring. Initially, all the sites are busy but soon site s_1 becomes free and begins to process the operations in its queue. If an operation *op* is evaluated when s_1 becomes free at time t_1 then, under this method, we will artificially block its re-evaluation until time t_2, when one of the other sites changes its state from busy to free. The vertical lines in Figure 4 indicate the times after which operations may be re-evaluated thanks to a change in the status of some site or other.

In general, one would expect this method to have only a limited effect on disruption, since it is still possible for an operation to be evaluated many times during any one free period. However, by the same token, the negative effects on accuracy can also be expected to be limited, since the condition is still being monitored on a regular basis. Moreover, although this method was designed with the intention of reducing indirect disruption, it should also show some beneficial effects in terms of direct disruption, since the number of ship operations executed during any one free period may also be reduced.

Method 4: Monitor Conditions once per Free Period. If method 3 is expected to have only a limited effect on disruption, then an obvious way to improve it is to impose a more stringent criterion for re-evaluation that further reduces the frequency with which operations

are executed. Method 4 does exactly this by ensuring that each operation is executed at most once in each free period.

The principal advantage of this method is its simplicity. However, its beneficial effects could also be significant, especially in systems which have a high update rate and/or timetables containing many long periods of free time. By the same token, we would expect to see a corresponding reduction in accuracy of condition monitoring with this method.

Method 5: Monitor Conditions During Global Free Periods. The final method aims to reduce both types of disruption, by taking the rather drastic step of only performing condition monitoring work when *all* sites are free at the same time. If our understanding of the causes of the disruptive effects of TTCM is correct then this method should eliminate both direct and indirect disruption completely. However, unless the timetables for all systems include an unusually high proportion of overlapping free time, this method is also expected to have an extremely severe effect on the accuracy of condition monitoring.

4. Experimental Evaluation

Having implemented the five methods within the TTCM system, we next undertook an experimental evaluation to determine whether our predictions as to their effectiveness were correct. In order to do this, it was necessary to define quantitative measures of the two characteristics of interest: namely, disruption and accuracy. We define the disruption of a method in terms of its effect on the total number of business transactions that are processed over the course of an experiment. We refer to the number of transactions executed during an experiment as the *TransactionNumber*.

We define the *inaccuracy* of a method to be equivalent to the total amount of time during the experiment for which the condition monitoring system has recorded a false positive or a false negative result for the condition. A false positive result occurs when the condition monitoring systems states that, for some period of time, the condition was true, when in fact it was false. Similarly, a false negative result occurs when the condition monitoring system states that the condition was false when it was in reality true. By totalling up the length of time during which an inaccurate result was recorded, we can gain some indication of the inaccuracy inherent in the condition monitoring method used.

In order to measure these quantities when the methods were in action within the TTCM system, we developed a simple order handling system, distributed over three sites. We also developed an experimental framework which allowed us to execute transactions against the database and

record the actual state of the monitored condition, while using one of our methods to record the state of the condition as seen by the TTCM system for comparison. Each run of the system lasted for a period of 12 hours.

We used this experimental framework to assess the benefits and costs of each of the five methods presented above. In order to provide upper and lower baselines against which to compare our methods, we also measured the disruption and inaccuracy in two extreme cases. Lower bounds were provided by measuring the effects of running the basic TTCM evaluation method, which executes the condition monitoring operations during free periods without making any additional attempt to control disruption. For convenience, we refer to this as method 0.

The upper bounds were found by executing the sequence of business transactions with TTCM switched off (method 6). The results of this process gave us the maximum number of transactions that could be processed during the 12 hour period of the experiment. Similarly, since no condition monitoring was taking place, the record of the data items that satisfy the condition will be empty. The inaccuracy of an empty TTCM result does not tell us what the maximum possible level of inaccuracy is since, for many data items, an empty condition satisfaction record is an entirely accurate result. However, it does give a useful benchmark figure against which to compare the inaccuracy that results from the other methods, since it is almost certainly going to be worse than trying to perform some kind of TTCM, however limited.

Since the success of each individual method is dependent upon the exact form of the timetable in use at any one time, we have performed two sets of experiments with two different sets of timetables, so that we could get some initial idea of the stability of our results. The key factor in determining how hard or how easy it is to avoid disruption seems to be the ratio of:

- the average length of the free periods over all sites (*AvgLengthLFP*), to
- the average cost of condition monitoring for each site (*AvgCostCM*).

The two sets of experiments use timetables which were deliberately designed to give very different values for the ratio of these two quantities (*AvgLengthLFP/AvgCostCM* = 17 and = 2.5). Figures 5 and 6 illustrate the results in each case. In the graphs, inaccuracy is given in terms of minutes, while transaction number is a simple count of the transactions that were completed.

As we predicted, method 5 does indeed result in the least disruption, coming close to the upper bound in terms of the number of transactions

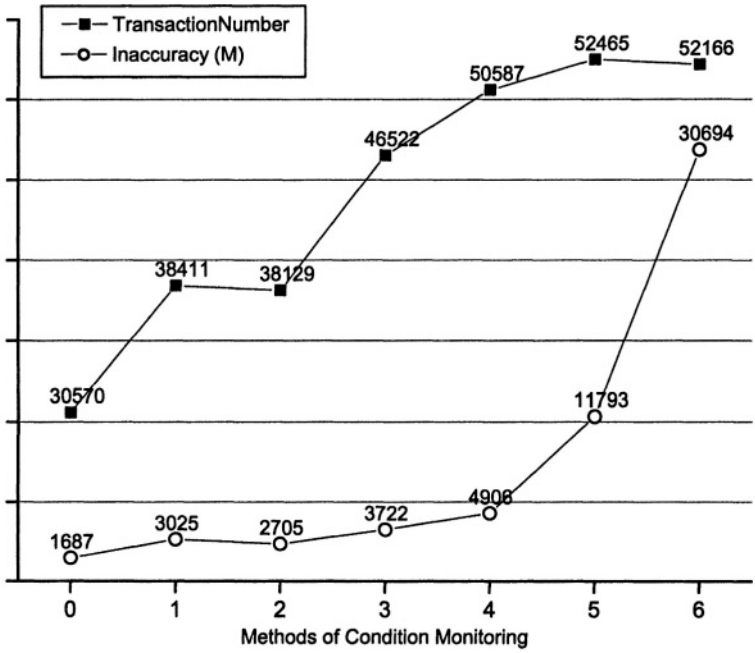

Figure 5. **Disruption and Inaccuracy Resulting from the Five Methods:** $AvgLengthLFP/AvgCostCM = 17$

that could be executed. However, this benefit comes at a heavy cost in inaccuracy, which is four to six times the inaccuracy of the basic TTCM method (method 0).

Both methods 1 and 2 have a positive effect on disruption and a surprisingly small negative effect on accuracy. On the basis of these results, there is very little to choose between them, which perhaps suggests that the extra complications of method 2 are not worth the time and trouble it takes to implement them.

Methods 3 and 4 are both extremely successful when *AvgLength-LFP/AvgCostCM* is higher; that is, when the free periods are significantly longer than the amount of time required to check the condition as a whole. The number of transactions that can be processed when using these methods is almost as high as that when no condition monitoring is taking place. In addition, there is a surprisingly small effect on inaccuracy, with the amount of false positives and false negatives being

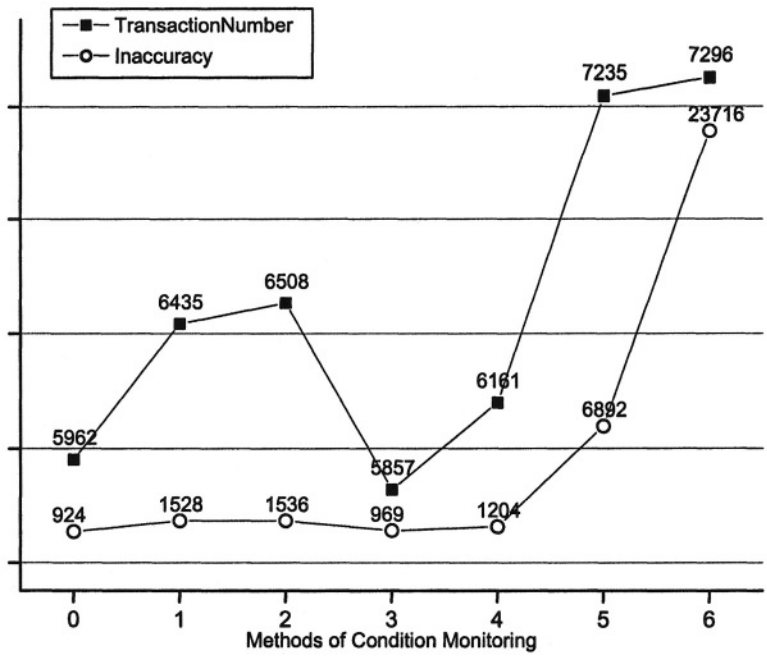

Figure 6. Disruption and Inaccuracy Resulting from the Five Methods: $AvgLengthLFP/AvgCostCM = 2.5$

scarcely larger than the lower baseline figure itself. However, neither method is very successful in the context of a lower value for *AvgLength-LFP/AvgCostCM*.

This is reasonable. When the free periods are only just long enough to allow the execution of one TTCM operation, then attempts to reduce the frequency of execution of those operations is unlikely to make much of a difference on the overall outcome. Effectively, in this experiment set, methods 3 and 4 were operating very much like the basic TTCM evaluation strategy; hence the limited improvement in disruption and the low levels of increased inaccuracy.

5. Conclusions

We have presented five different methods for reducing the disruption caused by time-tabled condition monitoring, and have outlined how we have evaluated their varying effectiveness using our experimental frame-

work. None of the methods emerges as a clear winner from this evaluation. Either the positive effect on disruption is limited or else there is a substantial negative effect on the accuracy of the results. Methods 3 and 4 performed excellently in the first experiment, but were much less successful in the second. Presumably, it would be possible to construct scenarios in which each of these methods could perform well. The question as to which method to use in practice, therefore, depends in part on the particular characteristics of the DIS to which TTCM is to be applied and in part on whether lack of disruption or high accuracy is of most importance to the owners of the system.

For example, if absence of disruption is the key priority, and accuracy is considered to be of secondary importance, then method 5 is the obvious choice. It will eliminate almost all the disruption, but the results of condition monitoring will contain many phantoms and omissions. If, on the other hand, accuracy of results is a major concern, over and above disruption, then the basic TTCM method should be adopted.

More typically, however, system owners will be looking to achieve a compromise between accuracy and disruption. In such a case, perhaps the best approach is to try to determine the principal causes of disruption within the system and to choose the most appropriate method for reducing it. In addition, system owners should take into account the characteristics of their workload timetables and the conditions that are to be monitored and thus determine the value of *AvgLengthLFP/AvgCostCM* for their system. If this value is high (say, greater than 10) then there is a good chance that methods 3 and 4 may be effective in reducing disruption without too serious an effect on disruption. Otherwise, these methods should probably be avoided. Tables 1 and 2 summarise our recommendations for choosing a specific method based on system characteristics and stakeholder priorities.

SuggestedMethod / Accuracy / TransactionNumber	*VeryImportant*	*Important*	*NotImportant*
VeryImportant	3 or 4	3 or 4	5
Important	0	3 or 4	5
NotImportant	0	0	*

Table 1. Choice of Method when $AvgLengthLFP/AvgCostCM > 10$

SuggestedMethod / Accuracy / TransactionNumber	*VeryImportant*	*Important*	*NotImportant*
VeryImportant	2	2	5
Important	0	2	5
NotImportant	0	0	*

Table 2. Choice of Method when $AvgLengthLFP/AvgCostCM \leq 10$

Acknowledgments

The authors wish to thank Mark Greenwood and Ben Senior for their helpful discussions and suggestions on the work described in this paper.

References

[1] R. Barquin, and H. Edelstein. *Building, Using, and Managing the Data Warehouse.* Prentice Hall, 1997.

[2] Philip A. Bernstein, Barbara T. Blaustein, and Edmund M. Clarke. Fast maintenance of semantic integrity assertions using redundant aggregate data. In *Sixth International Conference on Very Large Data Bases, October 1-3, 1980, Montreal, Quebec, Canada, Proceedings,* pages 126–136. IEEE Computer Society, 1980.

[3] Stephanie J. Cammarata, Prasadram Ramachandra, and Darrell Shane. Extendeing a relational database with deferred referential integrity checking and intelligent joins. In James Clifford, Bruce G. Lindsay, and David Maier, editors, *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data, Portland, Oregon, May 31 - June 2, 1989,* pages 88–97. ACM Press, 1989.

[4] Stefano Ceri and Jennifer Widom. Deriving production rules for constraint maintainance. In Dennis McLeod, Ron Sacks-Davis, and Hans-Jörg Schek, editors, *16th International Conference on Very Large Data Bases, August 13-16, 1990, Brisbane, Queensland, Australia, Proceedings,* pages 566–577. Morgan Kaufmann, 1990.

[5] Surajit Chaudhuri, Ravi Krishnamurthy, Spyros Potamianos, and Kyuseok Shim. Optimizing queries with materialized views. In Philip S. Yu and Arbee L. P. Chen, editors, *Proceedings of the Eleventh International Conference on Data Engineering, March 6-10, 1995, Taipei, Taiwan,* pages 190–200. IEEE Computer Society, 1995.

[6] Latha S. Colby, Timothy Griffin, Leonid Libkin, Inderpal Singh Mumick, and Howard Trickey. Algorithms for deferred view maintenance. In H. V. Jagadish and Inderpal Singh Mumick, editors, *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, June 4-6, 1996,* pages 469–480. ACM Press, 1996.

[7] Armin B. Cremers and G. Doman. Aim - an integrity monitor for the database system ingres. In Mario Schkolnick and Costantino Thanos, editors, *9th International Conference on Very Large Data Bases, October 31 - November 2, 1983, Florence, Italy, Proceedings,* pages 167–170. Morgan Kaufmann, 1983.

[8] Stefan Grufman, Fredrik Samson, Suzanne M. Embury, Peter M. D. Gray, and Tore Risch. Distributing semantic constraints between heterogeneous databases. In Alex Gray and Per-Åke Larson, editors, *Proceedings of the Thirteenth International Conference on Data Engineering, April 7-11, 1997 Birmingham U.K,* pages 33–42. IEEE Computer Society, 1997.

[9] Ashish Gupta. *Partial Information Based Integrity Constraint Checking, PhD. Thesis.* Stanford University, USA, 1994.

[10] Ashish Gupta and Jennifer Widom. Local verification of global integrity constraints in distributed databases. In Peter Buneman and Sushil Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 26-28, 1993,* pages 49–58. ACM Press, 1993.

[11] Binling Jin and Suzanne M. Embury. Non-intrusive assessment of organisational data quality. In E.M. Pierce and R. Katz-Haas, editors, *The 6th International Conference on Information Quality (IQ-2002), Cambridge, MA, USA, November 2001,* pages 398–411, 2001.

[12] Binling Jin and Suzanne M. Embury. Increasing the accuracy of time-tabled condition monitoring. In *Integrity and Internal Control in Information Systems V, IFIP TC11/WG11.5 Fifth Working Conference on Integrity and Internal Control in Information Systems (IICIS), November 11-12, 2002, Bonn, Germany,* volume 251 of *IFIP Conference Proceedings,* pages 21–35. Kluwer, 2003.

[13] Subhasish Mazumdar. Optimizing distributed integrity constraints. In Song C. Moon and Hideto Ikeda, editors, *Proceedings of the 3rd International Conference on Database Systems for Advanced Applications (DASFAA), Daejeon, Korea, April 6-8, 1993,* volume 4 of *Advanced Database Research and Development Series,* pages 327–334. World Scientific, 1993.

[14] Jean-Marie Nicolas. Logic for improving integrity checking in relational databases. *Acta Informatica,* 18(3):227–253, 1982.

[15] Xiaolei Qian. Distribution design of integrity constraints. In Larry Kerschberg, editor, *Expert Database Systems, Proceedings From the Second International Conference, Vienna, Virginia, USA, April 25-27, 1988,* pages 205–226. Benjamin Cummings, 1989.

[16] Xiaolei Qian and Gio Wiederhold. Incremental recomputation of active relational expressions. *IEEE Transactions on Knowledge and Date Engineering,* 3(3):337–341, September 1991.

[17] Eric Simon and Patrick Valduriez. Design and implementation of an extendible integrity subsystem. In Beatrice Yormark, editor, *SIGMOD'84, Proceedings of Annual Meeting, Boston, Massachusetts, June 18-21, 1984,* pages 9–17. ACM Press, 1984.

[18] Divesh Srivastava, Shaul Dar, H. V. Jagadish, and Alon Y. Levy. Answering queries with aggregation using views. In T. M. Vijayaraman, Alejandro P. Buchmann, C. Mohan, and Nandlal L. Sarda, editors, *VLDB'96, Proceedings*

of 22th International Conference on Very Large Data Bases, September 3-6, 1996, Mumbai (Bombay), India, pages 318–329. Morgan Kaufmann, 1996.

[19] Michael Stonebraker. Implementation of integrity constraints and views by query modification. In W. Frank King, editor, *Proceedings of the 1975 ACM SIGMOD International Conference on Management of Data, San Jose, California, May 14-16, 1975,* pages 65–78. ACM Press, 1975.

[20] Jennifer Widom, Roberta Cochrane, and Bruce G. Lindsay. Implementing set-oriented production rules as an extension to starburst. In Guy M. Lohman, Amílcar Sernadas, and Rafael Camps, editors, *17th International Conference on Very Large Data Bases, September 3-6, 1991, Barcelona, Catalonia, Spain, Proceedings,* pages 275–285. Morgan Kaufmann, 1991.