行政院國家科學委員會補助專題研究計畫成果報告

※※※※※※※※※※※※※※※※※※※※※※※※※※

※ ※

※ 即時異動排程暨系統恆久考量 ※

※ ※

※※※※※※※※※※※※※※※※※※※※※※※※※※

計畫類別：☑個別型計畫 □整合型計畫

計畫編號：NSC90-2213-E-002-080

執行期間：90 年 08 月 01 日至 91 年 07 月 31 日

計畫主持人：郭大維

本成果報告包括以下應繳交之附件：

- □ 赴國外出差或研習心得報告一份
- □ 赴大陸地區出差或研習心得報告一份
- □ 出席國際學術會議心得報告及發表之論文各一份
- □ 國際合作研究計畫國外研究報告書一份

執行單位：國立台灣大學資訊工程所

中 華 民 國 91 年 8 月 31 日

行政院國家科學委員會專題研究計畫成果報告

計畫名稱：即時異動排程暨系統恆久考量

（Real-time Transaction Scheduling and System Durability Consideration）

計畫編號：NSC90-2213-E-002-080

執行期限：計畫自民國 90 年 08 月 01 日起至民國 91 年 07 月 31 日止

主持人：郭大維　　國立台灣大學資訊工程所

計畫參與人員：郭錦福、蔡能聰、吳卓俊、羅習五、侯彥希

國立台灣大學資訊工程所

一、 摘要

中文摘要：

本計畫主要在延伸一個著名的即時並行控制協定 Read/Write Priority Ceiling Protocol (RWPCP)，來解決當考量交易交付時，RWPCP 的不可回復，與優先權反轉的次數不被限制的問題。報告的內容為兩部分：依序是系統發展的緣由與目的。接著介紹 RWPCP 的機制，我們將 RWPCP 延伸為 RWPCP-DC 與 RWPCP-ELP 來解決回復與優先權反轉的相關問題，並套用四種查核點機制來檢視可行性。

Abstract :

This project mainly focuses on extending a well-known real-time concurrency control protocol, the Read/Write Priority Ceiling Protocol (RWPCP), to solve the following problems: RWPCP could be unrecoverable, and the priority inversion problems could be unbounded, when transaction commitment is considered. We also explore different mechanisms for durability considerations.

二、 緣由與目的

Recoverability and transaction durability have been important research topics in non-real-time main-memory database systems in the past decades [2, 3, 5]. A *write-through* procedure must be done to enforce the updating of transaction logs into stable storage [2, 8]. Real-time transaction scheduling, however, on the issues in maintaining transaction durability, checkpointing, and write-through procedures, is often ignored in the design of real-time concurrency control protocols and in their performance evaluation [3, 4, 9].

We first show that schedules generated from RWPCP could be irrecoverable when transaction commitment is considered. We then revise RWPCP by two simple techniques, i.e., *deferred commitment* and *extended locking period* [2, 8], and provide discussions on their impacts on priority inversion and deadlock-freeness.

In this project, we point out different aspects of the durability issues on real-time concurrency control, they could be integrated together under the scheduling framework of RWPCP. That is, RWPCP-DC and RWPCP-ELP with different checkpointing processes/methods.

三、 即時系統動態排程與恆久性

3.1 研究簡介

The purpose of this project is to investigate the well-known RWPCP [10] on the recoverability issues. RWPCP introduces a write priority ceiling WPLi and an absolute priority ceiling APLi for each data object Oi to emulate share and exclusive locks, respectively. The write priority ceiling WPLi of data object Oi equals to the priority of the highest priority of the transactions which may write Oi. The absolute priority ceiling APLi of data object Oi equals to the priority of the highest priority of the transactions which may read or write Oi. When data object Oi is read-locked, the read/write priority ceiling RWPLi of Oi is set to WPLi. When data object Oi is write-locked, the read/write priority ceiling RWPLi of Oi is set to APLi.

The transaction, which has the highest priority among all executing transactions, will be assigned to use the processor. The lock request of a transaction ti may be granted if its priority is higher than the highest read/write priority ceiling RWPLi of all the data objects currently locked by other transactions. Otherwise, the lock request is rejected and the transaction will be blocked until its requested lock is released by the holder. A transaction ti uses its assigned priority for execution and setting locks, unless it is blocking a higher priority transaction. For this case, a priority inheritance technique is adopted.

3.2 延伸 RWPCP 來解決不可回復及優先權反轉問題

A simple way to make a RWPCP schedule recoverable is to defer the commitment of a transaction t until all transactions from which t reads commit. We call this variant of RWPCP as RWPCP with Deferred Commitment (RWPCP-DC). In order to reduce the delay period, the priority inheritance policy may be adopted for those transactions from which the delayed transaction t reads. The deferred commitment mechanism generates recoverable schedules by reordering the commitment times of transactions. However, the problem of RWPCP-DC is that the maximum number of priority inversion for a transaction becomes unbounded because there may exist a sequence of uncommitted transactions, where each transaction reads from another transaction in the sequence.

Another way to make RWPCP recoverable is to extend the locking period of the write locks of a transaction to prevent other transactions from reading dirty data until the commitment of the transaction. We call this variant of RWPCP as RWPCP with Extended Locking Period (RWPCP-ELP). Since RWPCP requires nested locking of data objects, RWPCP-ELP should keep the locking of data objects properly nested.

3.3 套用四種查核點機制

We model checkpointing under real-time scheduling framework using RWPCP. Different checkpointing methods are considered: fuzzy

checkpointing [4], record-based checkpointing [9], page-based checkpointing, and transaction-consistent checkpointing [9]. The four checkpointing methods represent four different granularities in checkpointing, which are in a word, a data object (record), a page, or a segment, where a segment is a collection of pages such that no transaction accesses data belonging to different segments.

A fuzzy checkpointing process proceeds as follows [4, 9]: when a new fuzzy checkpoint begins, the checkpointer writes a begin-checkpoint marker to stable storage. The checkpointer then flushes (dirty) pages from the main memory to stable storage, where locks and other transaction activities are ignored. The updates may be reflected in the backup and the logs in stable storage. When the checkpointer finishes the backup work, an end-checkpoint marker will be written to stable storage. After a system failure, the recovery manager will first read the backup database into the main memory and then apply the logs to the database to restore the database back to a consistent state before the failure.

The idea of record-based checkpointing is similar to the idea of action-consistent checkpointing in [9]. The checkpointer locks and copies one record at a time to the I/O buffer. Before the checkpointer locks the next record, the checkpointer must unlock the currently locked record. The checkpointer backups records in the same memory page consecutively and processes one page at a time. The backup order of records in the same page is assumed to be arbitrary. After a system failure, the recovery manager will first read the backup database into the main memory and then apply the logs from stable storage to the database to bring it back to a consistent state [9].

The procedures of the page-based checkpointing is the same as record-based checkpointing, except that the checkpointer locks and flushes a whole page, instead of one record, at a time. It is assumed that a page is larger than a record. Locking a page instead of a record can reduce the locking overheads but increase the probability of lock conflict with real-time transactions. After a system failure, the recovery manager will first read the backup database into the main memory and then apply the logs to the database to bring the database back to a consistent state.

The transaction-consistent checkpointing algorithm discussed in this project is closely related to the idea of transaction-consistent checkpointing in [9]. During a checkpointing process, the checkpointer backups segments one by one. It locks all pages in a segment in the two-phase locking (2PL) fashion and flushes the (dirty) pages from the main memory to the secondary storage, where a segment is a collection of pages such that no transaction accesses data belonging to different segments. If DMA is available, then DMA is initiated for I/O transfers. Otherwise, disk I/O operations are initiated.

四、 結果與討論

In this project, we are interested in the impacts of different checkpointing methods and write-through procedures on lock-based real-time concurrency control protocols. We choose the well-known Read/Write Priority Ceiling Protocol (RWPCP) to explore the durability and related performance issues. We have shown that schedules generated by the original RWPCP can be irrecoverable, and the maximum number of priority inversions for a real-time transaction can be more than one without proper modifications. The performance of RWPCP is explored when it is revised based on well-known mechanisms such as deferred commitment and extended locking period to maintain the recoverability in transaction execution.

We believe that more research on the recovery mechanisms may be proved very rewarding in building robust real-time data-intensive applications.

五、 參考文獻

[1] R. Abbott, and H. Garcia-Molina (1992) Scheduling real-time transactions: a performance evaluation. ACM Transactions on Database Systems, vol. 17, no. 3, pp. 513-560.

[2] P.A. Bernstein, V. Hadzilacos, and N. Goodman (1987) Concurrency Control and Recovery in Database Systems, Addison-Wesley, Massachusetts, USA.

[3] H. Garcia-Molina and K. Salem (1992) Main Memory Database Systems: An Overview, IEEE Transactions on Knowledge and Data Engineering, vol. 4, no. 6, pp. 509-516.

[4] R.B. Hagmann (1986) A Crash Recovery Scheme for a Memory-Resident Database Systems, IEEE Transactions on Computers, Vol. 35, No. 9, September, pp. 839-843.

[5] T. Haerder and A. Reuter (1983) Principles of Transaction Oriented Database Recovery, ACM Computing Survey, Vol. 15, No. 4, pp. 287-317.

[6] T.-W. Kuo and A.K. Mok (1993) SSP: a Semantics-Based Protocol for Real-Time Data Access, the 14th IEEE Real-Time Systems Symposium, December, New York, pp. 76-86.

[7] T.-W. Kuo, Y.-T. Kao, and L.-C. Shu (1998) A Two-Version Approach for Real-Time Concurrency Control and Recovery, the 3th IEEE International High Assurance Systems Engineering Symposium, Washington DC, November 13 - 14.

[8] E. Navathe (1994) Fundamentals of Database Systems," Second Edition, Addison-Wesley.

[9] K. Salem and H. Garcia-Molina (1989) Checkpointing Memory-Resident Databases, the 5th IEEE International Conference on Data Engineering, pp. 452-462.

[10] L. Sha, R. Rajkumar, S. H. Son, C.H. Chang (1991) A Real-Time Locking Protocol, IEEE Transactions on Computers, vol. 40 , no. 7, pp.793-800.