Transactional Model and Evalution of Vehicle System *

7P−7

Hironari Murai, Makoto Takizawa †
Tokyo Denki University ‡
e-mail{hiro,taki}@takilab.k.dendai.ac.jp

1 Introduction

There are various kinds of systems where something is moving, e.g. motor cars in roads, packets in network, automated guided vehicles in factories, and workflows in offices. A ***vehicle system*** is a model obtained by abstracting these systems, which is composed of *vehicles* moving in the *space*. The space is considered to be composed of units named *space objects*. Each object denotes some parts of paths. The objects are further partitioned into smaller objects. An object denoting a path is represented by a sequence of objects, each of which denotes a part of the path. Each higher-level object denotes more abstract and broader area than the lower-level ones. Takizawa and Hamada [3] discuss a ***broad path decision*** where each vehicle v first finds a ***broader*** path composed of higher-level objects to the destination, and then the path is so detailed as to include lower-level objects as v is approaching to the destination.

Since each object o can admit a limited number of vehicles, some vehicle cannot pass o or it takes longer time if o is congested. There are two approaches to resolving conflict among requests on o from multiple vehicles, i.e. scheduling and dynamic resolution. In the scheduling method, the vehicle movement is decided what time and what objects each vehicle v would pass before v departs. In vehicle systems including large number and various kinds of vehicles in a complex space, it consumes much computation to make up the schedule. In the resolution method, each vehicle v tries to pass o any time v would like to pass o. In this paper, we adopt the resolution scheme where v locks o before passing o. If the lock request is accepted, v can pass o. We discuss three schemes for releasing objects with respect to when vehicles release objects, i.e. *open*, *semi-open*, and *close* ones. In addition, we discuss how to relate the releasing schemes with the types of vehicles.

In section 2, we present a system model. In section 3, we present synchronization schemes by which the vehicles lock and release objects. In section 4, we present the evaluation of the synchronization schemes.

2 System Model

A vehicle system **T** is composed of a collection **V** of vehicles and a space **S** which is a set of paths where the vehicles move. A set of roads is an example of the vehicle space where motor cars move around as the vehicles. S is partitioned into disjoint units named *objects*.

Each primitive path $\langle o\rangle$ can be represented by a path $p = \langle\langle o_1\rangle, \ldots, \langle o_n\rangle\rangle$ where each o_i is a component of o. p is written as $\langle o\rangle^1$. $\langle o\rangle^1$ is named a *component path* or *first expansion* of $\langle o\rangle$. The *i-th expansion* $\langle o\rangle^i$ is defined as $\langle\langle o_1\rangle^{i-1}, \ldots, \langle o_n\rangle^{i-1}\rangle$ for $i \geq 1$, and $\langle o\rangle^0 = \langle o\rangle$.

[Definition] Let p, q, and r be paths from an object s to d. q is a ***broadest*** path of p if $p \prec\!\!\prec q$ and no path r such that $q \prec\!\!\prec r$. q is a ***most detailed*** path of p if $q \prec\!\!\prec p$ and there is no object r such that $r \prec\!\!\prec q$ □.

Each movement of a vehicle v on an object o to be *atomic*, i.e. v can pass either o or not. Hence, we model the atomic movement as a *transaction* on the vehicle space in this paper.

Since the vehicle space is tree-structured, the atomic movement of v on o is realized as a sequence of atomic movements on component objects $o_1, \ldots, o_n$ of o. The movement v on the component o_i is also modeled as a *transaction* named a *subtransaction*. Thus, the movement of v is a *nested* transaction [2].

2.1 Types of vehicles

There are types of vehicles on how they could move in the vehicle space. The vehicles are classified with respect to the following three points.

1. Vehicles are *removable* or not.
2. Vehicles are *stoppable* or not.
3. Vehicles are *retreatable* or not.
4. Vehicles are *separatable/mergetable* or not.

There are two kinds of vehicles, i.e. one can be removed and the others cannot. The second point is concerned with whether vehicles can stop or not. The third point is concerned with whether or not v can back along the path passed by v. For each *unretreatable* vehicle v, as soon as v passes o, v can release o since v never backs. On the other hand, if v is retreatable, v can hold o since v may back the same path which v has passed. One train composed of multiple cars can be separated to multiple subtrains, each of which includes the cars in the train. Each subtrain can take different paths. Such a vehicle is *separatable*. In addition, the subtrains can be merged into one train. Such a train is *mergeable*.

3 Vehicle Transaction

3.1 Locking scheme

In order to make sure that v can pass o, v locks o before arriving at o. If v locks o, v is assured to be able to pass o. After leaving o, v can release o. Here, suppose that $cap(\langle o\rangle)$ represents how many vehicles o can admit at the same time and $hold(\langle o\rangle)$ denotes a number of vehicles which are now on o. If $cap(\langle o\rangle) - hold(\langle o\rangle) \geq 0$, o can accept further lock requests to pass $\langle o\rangle$ from other vehicles.

Suppose that v on o_1 would like to pass $\langle o\rangle$. First, v sends a lock request to o.

[Locking scheme]

(1) If $cap(\langle o\rangle) - hold(\langle o\rangle) \geq 0$, o tries to lock all the ancestors of o for v. If all of the ancestors cannot be locked, v cannot lock o. If locked, o is locked and $hold(\langle o\rangle) := hold(\langle o\rangle) - 1$.

(2) Otherwise, o rejects the lock request from v. That is, v cannot hold o. □

The lock request to o is propagated to the ancestor objects of o. If all the ancestors could be locked, o

*移動体システムのトランザクションモデルと評価
†村井 洋成, 滝沢 誠
‡東京電機大学

can be locked. Unless v can pass o_j after passing o_1, ..., o_{j-1}, i.e. v cannot lock $\langle o_j \rangle$ or cannot lock all the ancestors of o_j while locking $\langle o_j \rangle$, v_j is referred to as *abort*. In the conventional database systems, if a transaction aborts, the transaction disappears after the whole update effect done by the transaction is removed. On the other hand, only the part of the transaction can be aborted, i.e. *partial abortion* [4].

$$\begin{tikzcd}
o_1 \arrow[r] & \cdots \arrow[r] & o_j \arrow[r] \arrow[rrrrrr, bend right, dashed] & \cdots \arrow[r] & o_i \arrow[rrrr, bend left, dashed] \arrow[ll, bend left, dashed] & \cdots & o_{i+1} & \cdots & o_n
\end{tikzcd}$$

Figure 1: Partial abortion

Suppose that v fails to pass some component o_i after passing $o_1, \ldots, o_{i-1}$. If v could find another way to o_n instead of passing $o_i, o_{i+1}, \ldots, o_n$ as show in Figure 2, v can pass o by taking the way found.

Suppose that a vehicle v can be separated into subvehicles $v_1, \ldots, v_n$, i.e. v is separatable. Each subvehicle v_i can be moved independently. Hence, v_1, ..., v_n are modeled to be subtransactions of v which can be executed in parallel. If all the subvehicles arrive at the destination, they are combined into one vehicle v. v commits if all the subtransactions v_1, ..., v_n commit. In addition, even if some v_i aborts, the others are not aborted because the subtransactions are independent.

3.2 Releasing schemes

There are three ways to release objects. Here, suppose that $\langle o \rangle^1 = \langle \langle o_1 \rangle, \ldots, \langle o_n \rangle \rangle$ and v_i denotes a subtransaction of v on o_i.

[Schemes for releasing objects]

(1) If v is a root of the vehicle, i.e. v arrives at the destination, all the locks held by v are released. Otherwise, no objects are released.

(2) $\langle o_1 \rangle$, ..., $\langle o_n \rangle$ held by the subtransactions v_1, ..., v_n of v are released.

(3) $\langle o \rangle$ is released, and all the paths obtained by v_1, ..., v_n are released if they are still obtained. □

The first scheme is a *close* scheme which is a strict *2PL* [1]. The second scheme is a *semi-open* one. It is noted that $\langle o \rangle$ is still held by v while $\langle o_1 \rangle$, ..., $\langle o_n \rangle$ obtained by v_1, ..., v_n of v are released. The third scheme is an *open* one where all the objects obtained by v are released. The open vehicle transactions hold less objects than the other two schemes. On the other hand, the open vehicle v may not back since all the objects passed by v are released.

It depends on the type of the vehicle which scheme is adopted to release objects. Suppose that v is *unretreatable*, i.e. v never backs along the path passed by v. Hence, as soon as v passes objects, v can release the objects, i.e. unretreatable vehicles can adopt the open scheme. On the other hand, A retreatable vehicle v can adopt the semi-open or close scheme since v cannot back if the objects which v has passed are released. If the close scheme is adopted, v can retreat, i.e. v can reverse the same path taken by v. In the semi-open scheme, v can find another path of the component objects of higher-level objects which v still holds. Since the semi-open scheme holds less objects than the close one, the semi-open one implies that more vehicles can be admitted in the space than the close one.

4 Evaluation

We evaluate the close, semi-open, and open schemes in terms of the number of objects held by each vehicle.

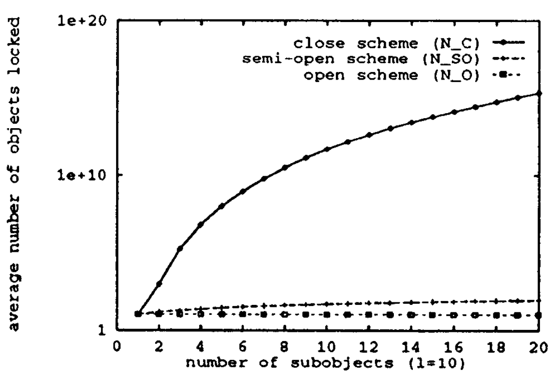

Figure 2: Evaluation of releasing schemes

We assume that a space tree T is height- and breadth-balanced, where the height is $l+1$ and each non-leaf object has k subobjects. Suppose that T denotes one-way path. The lowest-level path includes k^l lowest-level objects. We assume that a vehicle v moves on all the objects from the left-most lowest-level object to the right-most one, i.e. v passes k^l lowest-level objects. Suppose that v is in a lowest-level object o_l in T. A parent of o_l is denoted by o_{l-1}. Thus, o_i is an ancestor of o which is at a level $i (\leq l)$. Here, o_0 denotes the root of T. o_1 is the $a_1 (< k)$-*th* component object of the root, i.e. o_o. o_i is the a_i-*th* component object of o_{i-1} ($i = 1, \ldots, k$). The position of v in T is represented by $\langle a_1, \ldots, a_l \rangle$, where each $a_i \leq k$.

Let N_O, N_{SO}, and N_C be the average numbers of objects held by v in the open, semi-open, and close scheme, respectively. $N_O = l+1$, $N_{SO} = ((l+2)/2) + (kl/2)$, $N_C = ((l+2)/2) + l/2$ (for $k = 1$), $((l+2)/2) + (k/2)(k^l - 1)/(k-1)$ (for $k > 1$). Figure 1 shows the logarithms of N_O, N_{SO}, and N_C for k given $l = 10$. The semi-open transaction holds less objects than the close one. In the semi-open scheme, $O(k)$ objects are locked while $O(k^l)$ objects are locked in the close scheme.

5 Concluding Remarks

In this paper, the vehicle movement is modeled as a nested transaction in the space tree. The vehicles are classified in four points, *removable* or not, *stoppable* or not, *retreatable* or not, and *separatable* or not. We have shown *open*, *semi-open*, and *close* schemes for releasing objects and how each kind of vehicle can adopt a synchronization scheme. We have shown the evaluation of the releasing schemes in terms of the number of objects locked.

Reference

[1] Eswaren, K. P., Gray, J., Lorie, R. A., and Traiger, I. L., "The Notion of Consistency and Predicate Locks in Database Systems," *CACM*, Vol.19, No.11, 1976, pp.624-637.

[2] Lynch, N. and Merritt, M., "Introduction to the Theory of Nested Transactions," *MIT/LCS/TR 367*, 1986.

[3] Takizawa, M., Hamada, S., and Deen, S. M., "Vehicle Transactions," *Proc. of the 4rd Int'l Conf. on Database and Expert Systems Applications (DEXA '93)*, 1993, pp.611-614.

[4] Yasuzawa, S. and Takizawa, M., "Uncompensatable Deadlock in Distributed Object-Oriented Systems," *Proc. of the International Conference on Parallel and Distributed Systems (ICPADS)*, 1992, pp.150-157.