QoS-Based Concurrency Control on Multimedia Objects [*]

5L−01

Naokazu Nemoto, Katsuya Tanaka and Makoto Takizawa [†]
Tokyo Denki University [‡]
Email : {nemoto, katsu, taki}@takilab.k.dendai.ac.jp

1 Introduction

It is critical for applications to obtain enough quality of service (QoS) from multimedia objects. Not only states but also QoS of objects are changed by performing methods on the objects. Each object is required to be consistent in presence of multiple transactions issuing requests to the objects. We discuss new types of equivalent and conflicting relations among methods with respect to QoS. We introduce two types of locking modes, serialization and mutually exclusive modes to synchronize concurrent accesses to objects based on the QoS relations. We also discuss how those lock modes conflict.

2 QoS Based Compatible Relations

2.1 Consistent relations

A class *movie* is composed of two component classes; *advertisement* and *content*. An object m_1 created from the *movie* class is also composed of an *advertisement* object a_1 and a *content* object c_1. Another *movie* object m_2 is a same as m_1 except that the *advertisement* object of m_2 is a_2. An application does not care the difference between m_1 and m_2 since the application is interested in only the content of *movie*. Here, m_2 is considered to be *equivalent* with m_1. A class like *content* in which applications are interested is *mandatory*.

There are the following equivalent relations between a pair of states s_t and s_u of a class c:

- s_t is *state-equivalent* with s_u ($s_t - s_u$) iff $s_t = s_u$.
- s_t is *semantically equivalent* with s_u ($s_t \equiv s_u$) iff $s_t - s_u$ or $c_i(s_t) \equiv c_i(s_u)$ for every mandatory component class c_i of c.
- s_t is *QoS-equivalent* with s_u ($s_t \approx s_u$) iff $s_t - s_u$ or s_t and s_u are obtained by degrading QoS of some state s of c.
- s_t is *semantically QoS-equivalent* with s_u ($s_t \cong s_u$) iff $s_t \approx s_u$ or $c(s_t) \cong c(s_u)$ for every mandatory component class c_i of c.
- s_t is requirement QoS-equivalent (*RoS-equivalent*) with s_u on RoS R ($s_t -_R s_u$) iff $s_t \approx s_u$ and $Q(s_t) \cap Q(s_u) \succeq R$.
- s_t is *semantically RoS-equivalent* with s_u on RoS R ($s_t \equiv_R s_u$) iff $s_t -_R s_u$ or $c_i(s_t) \equiv_R c_i(s_u)$ for every mandatory class c_i of c.

Let *State*, *Sem*, *QoS*, *RoS*, *Sem-QoS*, and *Sem-RoS* be sets of possible state-, semantically, QoS-, RoS-, semantically QoS-, and semantically RoS-equivalent relations of a class c. Let E be a family of *State*, *Sem*, *QoS*, *RoS*, *Sem-QoS*, and *Sem-RoS*. For a pair of sets α and β in E, "$\alpha \rightarrow \beta$" means $\alpha \subseteq \beta$, showing that every pair of operations op_1 and op_2 satisfy the β-equivalency if op_1 and op_2 satisfy the α-equivalency.

[**Theorem**] The Hasse diagram as shown in Figure 1 holds for the α-equivalent relations. □

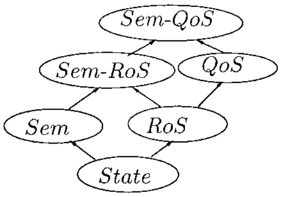

Figure 1: Equivalent relations.

2.2 Compatible relations

A method op_t *conflicts* with another method op_u in an object o iff the result obtained by performing op_t and op_u depends on the computation order. Multimedia objects are characterized by QoS in addition to the states. There are the following compatible relations between a pair of methods op_t and op_u of a class c:

- op_t is *state-compatible* with op_u ($op_t \mid op_u$) iff $op_t \circ op_u - op_u \circ op_t$.
- op_t is *QoS-compatible* with op_u ($op_t \parallel op_u$) iff $op_t \circ op_u \approx op_u \circ op_t$.
- op_t is *RoS-compatible* with op_u on RoS R ($op_t \mid_R op_u$) iff $op_t \circ op_u -_R op_u \circ op_t$.
- op_t is *semantically compatible* with op_u ($op_t \mid\mid\mid op_u$) iff $op_t \circ op_u \equiv op_u \circ op_t$.
- op_t is *semantically QoS-compatible* with op_u ($op_t \wr op_u$) iff $op_t \circ op_u \cong op_u \circ op_t$.
- op_t is *semantically RoS-compatible* with op_u on R ($op_t \mid\mid\mid_R op_u$) iff $op_t \circ op_u \equiv_R op_u \circ op_t$.

Let "*α-compatible* relation ($\diamond_\alpha$)" show some of the compatible relations where $\alpha \in$ {*State*, *Sem* (*semantically*), *QoS*, *RoS*, *Sem-QoS* (*semantically-QoS*), *Sem-RoS* (*semantically-RoS*)}. For example, $\diamond_{QoS}$ shows $\parallel$ and $\diamond_R$ shows $\mid_R$. op_t *α-conflicts* with op_u ($op_t \not\diamond_\alpha op_u$) unless $op_t \diamond_\alpha op_u$. For example, op_t QoS-conflicts with op_u ($op_t \nparallel op_u$) unless $op_t \parallel op_u$. $\diamond_\alpha$ is symmetric but is not transitive.

3 Synchronization

3.1 Traditional locking protocol

Suppose a pair of transactions T_i and T_j issue methods op_t and op_u to an object o, respectively. Here, T_i *precedes* T_j ($T_i \rightarrow T_j$) iff op_t and op_u conflict and op_t is performed on the object o before op_u. A collection of the transactions $T_1, \ldots, T_m$ are *serializable* iff either $T_i \rightarrow T_j$ or $T_j \rightarrow T_i$ for every pair of transactions T_i and T_j, i.e. the transactions are totally ordered in the precedent relation "→". In order to do that, an object o is locked before a method op_t is performed on o. If o is already locked for a method op_u conflicting with op_t, op_t blocks until the lock held by op_u is released. On the other hand, every pair of compatible methods can be performed on the object o in any order. An

[*] QoSに基づくマルチメディアオブジェクトの同期手法
[†] 根本 直一, 田中 勝也, 滝沢 誠
[‡] 東京電機大学

object o is locked in a mode $\mu(op_t)$ before op_t is performed. $\mu(op_t)$ *conflicts* with $\mu(op_u)$ if op_t conflicts with op_u. Otherwise the modes are *compatible.* If o is locked in a mode conflicting with $\mu(op_t)$, op_t blocks. Otherwise, op_t is performed. It is well known that a collection of transactions are *serializable* if every transaction is two-phase locked.

3.2 QoS based lock modes

The following orthogonal types of lock modes for a method op_t are introduced with respect to the α-compatibility and α-mutually exclusive relations :

1. *α-serialization* lock mode $\sigma_\alpha(op_t)$, and
2. *α-mutually exclusive* lock mode $\mu_\alpha(op_t)$.

The serialization locks are used to serialize a collection of conflicting methods issued by different transactions. That is, the lock can be used to decide which method to be started before the other. If an object is locked in an α-serialization mode conflicting with $\sigma(op_t)$, op_t blocks. The mutually exclusive locks are used to make methods mutually exclusively performed on an object. Let τ_1 and τ_2 be lock modes. τ_1 is compatible with τ_2 ($\tau_1 \diamond \tau_2$) iff τ_1 does not conflict with τ_2. Furthermore, $\mu_R(\mathit{reduce})$ is not compatible with $\mu_R(\mathit{mediascale})$. Every type of conflicting relation is assumed to be symmetric but not transitive.

Suppose that an object o is locked for a method op_t and another method op_u is issued to the object o. If $\sigma_\alpha(op_u)$ conflicts with $\sigma_\alpha(op_t)$, op_t blocks until op_u terminates. Suppose an object x supports a pair of methods op_1 and op_2 on x and another object y supports op_3 and op_4 on y. A transaction T_1 issues op_1 to x and op_3 to y. Another transaction T_2 issues op_2 to x and op_4 to y. First, suppose op_1 is α-compatible with op_2 ($op_1 \diamond_\alpha op_2$) and $op_3 \diamond_\alpha op_4$. Here, $\sigma_\alpha(op_1)$ is compatible with $\sigma_\alpha(op_2)$ ($\sigma_\alpha(op_1) \diamond \sigma_\alpha(op_2)$) and $\sigma_\alpha(op_3) \diamond \sigma_\alpha(op_4)$. op_1 and op_2 can be performed on the object x and op_3 and op_4 on y in any order. For example, op_1 is performed after op_2 on x and op_4 is performed after op_3 on y.

Next, suppose $\mu_\alpha(op_1)$ conflicts with $\mu_\alpha(op_2)$ but $\mu_\alpha(op_3)$ is compatible with $\mu_\alpha(op_4)$. op_2 can be started after op_1 completes. op_1 and op_2 cannot be concurrently performed. However, op_3 and op_4 can be concurrently performed on the object y because $\mu_\alpha(op_3)$ is compatible with $\mu_\alpha(op_4)$.

[Theorem] Suppose a method op_t strongly α-conflicts with op_u. A mutually exclusive mode $\mu_\alpha(op_t)$ conflicts with $\mu_\alpha(op_u)$ if a serialization mode $\sigma_\alpha(op_t)$ conflicts with $\sigma_\alpha(op_u)$. □

If a method op_t α-conflicts with another method op_u, op_t and op_u cannot be concurrently performed.

If a transaction T issues a method op_t in the α-conflicting relation to an object o, o is locked according to the following protocol.

[Locking protocol]

1. The transaction T first issues a serialization lock request $\sigma_\alpha(op_t)$ to the object o.
2. If o is not locked in any mode conflicting with $\sigma_\alpha(op_t)$, o is locked in $\sigma_\alpha(op_t)$ and then the lock mode is tried to be converted in a mode $\mu_\alpha(op_t)$.
3. If the lock mode is converted, op_t is ready to be performed on o.
4. Otherwise, op_t blocks. □

3.3 Relation among lock modes

Figure 1 shows how compatibility. Here, *State*, *QoS*, *RoS*, *Sem*, *Sem-QoS*, and *Sem-RoS* indicate sets of possible *State*-, *QoS*-, *RoS*-, semantically, and semantically *QoS*- and *RoS*-conflicting relations of a class c, respectively. Let C be a family of the sets $\{\mathit{State}, \mathit{QoS}, \mathit{RoS}, \mathit{Sem}, \mathit{Sem\text{-}QoS}, \mathit{Sem\text{-}RoS}\}$. The Hasse diagram as shown in Figure 1 holds for the α-conflicting relations in C. Let α_1 and α_2 be two types of conflicting relations. α_1 *dominates* α_2 ($\alpha_1 \succ \alpha_2$) iff $\alpha_1 \supseteq \alpha_2$ in Figure 1. α_1 and α_2 are *uncomparable* ($\alpha_1 \parallel \alpha_2$) if neither $\alpha_1 \succ \alpha_2$ nor $\alpha_2 \succ \alpha_1$. In Figure 1, *Sem* $\parallel$ *RoS*. Let $C_\alpha(\tau)$ be a set of lock modes which are compatible with a lock mode τ with respect to an α-compatible relation ($\diamond_\alpha$). The following property holds on the dominant relation "$\prec$" :

[Definition] A lock mode τ_1 on α_1 is *stronger* than another mode τ_2 on α_2 iff $C_{\alpha_1}(\tau_1) \subseteq C_{\alpha_2}(\tau_2)$. □

"$\tau_1 \succ \tau_2$" means that τ_1 conflicts with more number of lock modes than τ_2. The lock mode τ_1 is more restricted than τ_2. For example, *write* $\succ$ *read.*

Let α_1 and α_2 be types of *RoS*-conflicting relations on RoS R_1 and R_2, respectively. We discuss $\alpha_1 \succ \alpha_2$ or $\alpha_1 \prec \alpha_1$. Here, "$R_1 \succ R_2$" (R_1 dominates R_2) means that R_1 shows a higher level of QoS than R_2 as presented before. It is straightforward for the following theorem to hold from the definitions.

Suppose that R_1 and R_2 show monochromatic and colored movies. Let τ_1 and τ_2 be serialization lock modes of a method *grayscale* on RoS R_1 and R_2, respectively, i.e. $\tau_1 = \sigma_{R_1}(\mathit{grayscale})$ and $\tau_2 = \sigma_{R_2}(\mathit{grayscale})$. Here, $R_2 \succ R_1$. *grayscale* R_1-conflicts with *add* but is R_2-compatible with *add.* τ_1 is stronger than τ_2 ($\tau_1 \succ \tau_2$) since $C_{R_1}(\tau_1) \supset C_{R_2}(\tau_2)$.

4 Concluding Remarks

We discussed novel types of relations among methods on the basis of QoS and the state of an object, i.e. state-, semantically, QoS-, RoS-, and semantically QoS- and RoS-conflicting relations of methods in the object-based multimedia system. We presented the locking protocol to realize these new types of conflicting relations, where new lock modes, serialization and mutually exclusive modes are introduced. By using the serialization and mutually exclusive locks, we can increase the performance of the system.

In the locking protocol, the lock mode is converted from serialization ones σ to mutually exclusive ones μ is stronger the σ, transactions may be deadlocked. We need unsympathetic modes discussed in the paper [?].

Publications

1 Nemoto N., Tanaka K., and Takizawa M., "Quality-Based Synchronization Methods of Multimedia Objects," to appear in *Information Sciences an International Journal*, 2001.

2 Nemoto N., Tanaka K., and Takizawa M. : "Quality-Based Approach to Locking Multimedia Objects," to appear in *Journal of Internet Technology, special issue on "Internet Multimedia Information Systems"*, Vol. 2, No. 4, 309–316, 2001.

3 Nemoto, N., Tanaka, K., and Takizawa, M. : Quality-Based Concurrency Control for Multimedia Objects. *The 7th International Conf. on Distributed Multimedia Systems*(DMS'2001), 218–225, 2001.

4 Nemoto, N., Tanaka, K., and Takizawa, M. : Quality-Based Approach to Locking Multimedia Objects, *Proc. of the ICDCS-2001 Workshops*, 351–356, 2001.