M-012

複製計算モデルに基づくオブジェクト共有手法における 複製オブジェクトの楽観的な一貫性管理手法の提案

An Optimistic Management Method of Replicated Objects Consistency for Object Sharing Based on Replicated Computing

植田 亘 † Wataru Ueda
高田 秀志 ‡ Hideyuki Takada

1. はじめに

分散システムにおいて，複数の端末間でオブジェクト共有を可能とする手法の一つに，複製計算モデルに基づく，複製型オブジェクト共有手法がある [1]．これは，任意のオブジェクトの複製を各端末がそれぞれ保持し，その複製に対するメソッド呼び出しを各端末上で複製することで，オブジェクト共有を可能とする．この手法は，端末間で共有される複製オブジェクトが，それぞれの端末上に存在するため耐故障性に優れているが，一方で，各端上の複製オブジェクトの状態の一貫性を厳密に保証することが困難となる．そこで，本稿では，複製型オブジェクト共有手法の特性を活かした，複製オブジェクトの楽観的な一貫性管理手法を提案する．本手法では，あるプロセスで発生した複製オブジェクトへのメソッド呼び出しを，まずそのプロセス上で処理した後，他プロセス上で複製する．そのとき，メソッド呼び出しの伝播メッセージをベクタークロックを利用して順序付けることで，一貫性破綻の可能性を検知可能とする．これにより，もし，一貫性に破綻が生じた場合には複製オブジェクトの状態を修復することで，複製オブジェクトの一貫性を保証することが可能となる．

2. P2P 型複製オブジェクト環境

著者らは，複製型オブジェクト共有ための “複製オブジェクト環境”を提供する協調システム基盤 CUBE を開発している [2]．この環境では，図 1 のように，それぞれの端末がオブジェクトの複製を持ち，これらの振る舞いが同期されることによって，端末間で論理的に 1 つのオブジェクトを共有することが可能となる．この各端末が保持するオブジェクトの複製を “複製オブジェクト”と呼ぶ．ある端末上で，複製オブジェクトに対するメソッド呼び出しが発生した場合，それが他端末に伝播され，各端末が保持する複製オブジェクトの振る舞いが同期される．

複製オブジェクト環境では，各端末がそれぞれ P2P で接続され，同一の状態の複製オブジェクトを保持している．そのため，ある端末に障害が発生した場合でも，その端末が単一故障点にならないという性質を持っている．また，各端末で発生した複製オブジェクトへのメソッド呼び出しは，まずローカルで実行され，その後，他端末に伝播される．これによって，メソッド呼び出しの発生した端末では，メソッド呼び出し伝播のための通信オーバーヘッドに関わらず，迅速にメソッドが実行される．このように，複製オブジェクト環境は耐故障性と応答性の面に優れる特徴を持つ．しかし，一方で，各端末に分散した複製オブジェクトの一貫性を保持することが要求される．

3. 楽観的な複製オブジェクトの一貫性管理

複製オブジェクト環境において，厳密な一貫性保証を実現することは，耐故障性や応答性といった複製オブジェクト環境の特徴を制限することに繋がる．そこで本節では，複製オブジェクト環境の特徴を活かした一貫性保証を実現する楽観的な複製オブジェクトの一貫性管理手法について述べる．

† 立命館大学大学院 理工学研究科
‡ 立命館大学 情報理工学部

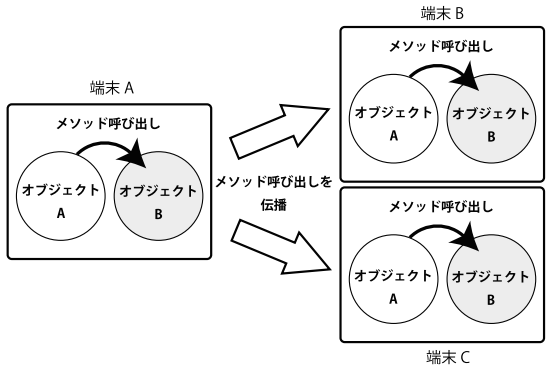

図 1　メッセージパッシングの複製

3.1 複製オブジェクト環境における複製オブジェクトの一貫性

複製オブジェクト環境においては，各々の端末は自身が保持している複製オブジェクトに任意のタイミングでアクセスすることができる．そのため，同期されている複製オブジェクトに対して複数の端末で同時にメソッド呼び出しが発生すると，図 2 のように，メソッド実行の順序が端末ごとに異なる場合がある．このとき，各端末の保持する複製オブジェクトの状態は同一ではない可能性があり，複製オブジェクトの状態の一貫性が損なわれている危険があると言える．これを防ぐためには，実行されるメソッド呼び出しの順序をすべての端末で同一にしなければならない．

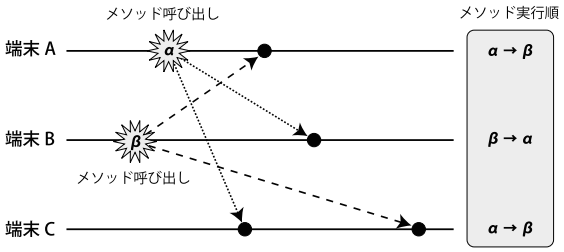

図 2　メソッド呼び出しの同時発生

3.2 楽観的な一貫性管理アプローチ

各端末上のプロセスで発生するメソッド呼び出しの順序を同一にするための方法として，一般的に以下の 2 つのアプローチが考えられる．

- 悲観的アプローチ
  排他制御などを用い，前もってメソッド実行順が厳密に各プロセスで同一になるように調整する．
- 楽観的アプローチ
  各プロセスのメソッド実行順が不一致である場合のみ，メソッド呼び出しの順序を調整する．

ここで，悲観的なアプローチを取った場合，各プロセスが連携し，複製オブジェクトへのアクセスの排他制御を行うことが必要となる．しかし，複製オブジェクトへのアクセスのたびにプロセス間の連携が密に発生すると，複製オブジェクト環境の特徴である耐故障性，および応答性を制限してしまう可能性がある．そのため，本稿では，楽観的アプローチでの一貫性管理手法を提案する．

楽観的アプローチを実現するための要件として，

- メソッド呼び出しの順序付け
- 一貫性破綻の検知
- 複製オブジェクトの状態修復

の 3 つが必要である．

3.3 メソッド呼び出しの順序付け

一般的に，分散システムにおいては，各プロセス間で発生したメッセージパッシングを順序付けるための仕組みとして論理クロックの考え方が用いられる [3]．論理クロックの中でも，ベクタークロックを利用した強制因果的通信はメッセージ間の因果性を表すメッセージ配信方式として利用されている [4]. 本手法では，この強制因果通信の考え方を用い，メッセージにタイムスタンプを付与することによって，順序付けを行う (図 3). 各プロセスはそれぞれ，ベクタークロック VC_i を保持する．ここで，$VC_i[i]$ はプロセス P_i 上で発生したメソッド呼び出しの数を表すものである．各プロセスのクロック VC_i は以下のステップで単調増加していく．

1. ローカルでメソッド呼び出しが起こった場合，$VC_i[i] \leftarrow VC_i[i]+1$ とする．
2. メソッド呼び出しの他ノードへの通知メッセージ m にタイムスタンプ $tm(m) = VC_i$ を設定する．
3. 他ノードからメッセージ m を受信した場合，各 k について，$VC_i[k] \leftarrow max\{VC_i, ts(m)[k]\}$ に調整する．

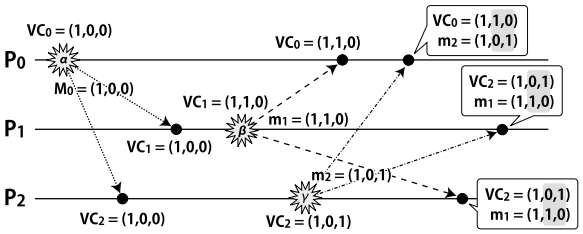

図 3　メソッド呼び出しの順序付け

3.4 一貫性破綻の検知

前節で述べたタイムスタンプを利用すると，あるプロセス間において，メソッド呼び出しのが異なる順序で実行されている場合を検知することができる．あるプロセス P_i が他のプロセス P_j からメソッド呼び出し通知メッセージ m を受信した場合を考える．このとき，P_i と P_j において，メソッド呼び出しが同順であるためには，以下の 2 つの条件を満たしている必要がある．

1. $ts(m)[j] = VC_i[j]+1$
2. $ts(m)[k] = VC_i[k]$ すべての $k \neq j$ について

1 つ目の条件は，P_i の受信したメッセージが P_j から次に来ると期待していたメッセージであることを表している．2 つ目の条件は，P_i の実行したメソッドを P_j もすべて実行していることを表している．

これらの条件が満たされない場合，P_i と P_j のメソッド呼び出しが同順で実行されていない．さらに，条件 2 を満たさない P_k の間でメソッド呼び出しの順序が前後していることが分かる．

例えば，図 3 において，P_1 と P_2 では，メソッド β と γ の実行順序が異なる．メソッド β への呼び出しが発生したとき，$VC_1 = (1,1,0)$ でありメッセージ m_1 は $ts(m_1) = (1,1,0)$ を付与されて送信される．しかし，P_2 では，m_1 を受信したさい，メソッド γ が実行されており，$VC_2 = (1,0,1)$ となっている．このとき，$ts(m_1) = (1,1,0)$ と $VC_2 = (1,0,1)$ を比べる．すると $k = 1,2$ に対して，$ts(m)[k] = VC_i[k]$ が満たされておらず，P_1 と P_2 の間で，メソッドの実行順序が前後していることが分かる．また，同様のことが P_1 でも検出される．このようにして，一貫性破綻の可能性を検知することができる．

3.5 複製オブジェクトの状態修復

一貫性破綻の検知がなされた後，複製オブジェクトの状態を修復するための手続きについて述べる．本手法では，あらかじめ各プロセスに優先度が設定されていると仮定する．他プロセスからのメッセージ受信時に一貫性破綻の検知がなされたプロセスは，自身が最も優先度の高いプロセスでない場合にはブロックされる．この時，最も優先度の高いプロセスは，一貫性破綻を検知した時点で，他の端末にオブジェクトの状態情報を送信する．ブロックされたプロセスは，最も優先度の高いプロセスから正しい複製オブジェクトの状態情報を受信し，修復を行う．

図 3 において，P_0，P_1，P_2 の順に高い優先度が設定されているとする．図の例では，P_1，P_2 がブロックされる．P_0 は m_2 受信時に，一貫性の破綻を検知すると，一貫性の破綻に関与していると判定されるプロセス P_1，P_2 に，自身の複製オブジェクトの状態情報を送信する．P_1，P_2 は，それぞれ P_0 から複製オブジェクトの状態情報を受け取ると，その情報をもとに，自身の複製オブジェクトの状態を修復し，ブロックを解除する．

あるプロセスにおいて，一貫性に破綻が生じた場合，優先度の高いプロセスの状態へ集束していくため，すべてのプロセスで複製オブジェクトの一貫性が保証される．

4. おわりに

本稿では，複製型のオブジェクト共有を可能とする複製オブジェクト環境において，複製オブジェクトの一貫性を管理するための手法について述べた．本手法では，ベクタークロックを利用したメッセージの順序付けによって，一貫性破綻の原因となるプロセスを特定し，状態修復手続きを行う．これによって，耐故障性や応答性といった複製オブジェクト環境の特徴を活かした楽観的な一貫性管理が可能となる．今後はノードの優先度決定アルゴリズムや，複製オブジェクトの状態情報をどのように扱うかといった問題を考察し，本手法の実装を進めていく．

参考文献

[1] David P. Reed, “Designing Croquet’s TeaTime- A Real-time, Temporal Environment for Active Object Cooperation,” Invited Talk, OOPSLA,2005.

[2] Shogo Noguchi, Hideyuki Takada, “CUBE: A Synchronous Collaborative Applications Platform Based on Replicated Computation,” Proceedings of the Fifth International Conference on Collaboration Technologies, pp.19-24, CollabTech2009, 2009.

[3] M. Raynal, M. Signhal, “Logical Time: A Way to Capture Causality in Distributed Systems,” IRISA, Technical Report, 1995.

[4] A.S. Tanenbaum, M. van Steen, “Distributed Systems: Principles and Paradigms,” Chapter 6, Prentice Hall, 2002.