M-054

端末間の処理能力を考慮した複製オブジェクトのふるまい同期手法
A Synchronization Method for Replicated Object Behavior Considering Processing Performance of Terminals

山本 佑樹† 植田 亘‡ 野口 尚吾‡ 高田 秀志†
Yuki Yamamoto Wataru Ueda Shogo Noguchi Hideyuki Takada

1. はじめに

複数の端末でユーザがオブジェクトを共有する仕組みの一つとして, 各ノード上にオブジェクトの複製を配置し, それらのオブジェクトの間でふるまいを同期させる手法が考えられる. 我々が開発を行っている Java による協調作業支援システム開発基盤 "CUBE"[1] では, このような手法を "オブジェクトミラーリング" と呼んでいる. オブジェクトミラーリングでは, あるノードであるオブジェクトのメソッド呼び出しが発生すると, それが他ノードへ伝播され, 他ノードでも同じメソッド呼び出しが実行される. この手法を処理能力に差のある端末間で実現する場合, 処理能力が高い端末で多くのメソッド呼び出しが発生すると, 処理能力が低い端末ではすべてのメソッド呼び出しを処理することができず, 処理の遅延が発生しうる. また, あるノードで発生したメソッド呼び出しを他ノードへ伝播させるとき, ネットワークの帯域に差があると, すべてのメソッド呼び出しを伝播できない可能性もある.

ここで, ノード間でオブジェクトのふるまいを完全に同期させるのではなく, あるノードで発生した一連のメソッド呼び出しのうち, アプリケーションにより許容される範囲で必要なもののみを選択し, 伝播するようにすれば, 上記の問題を解決できると考えられる. 例えば, マウスでオブジェクトをドラッグする操作をノード間で同期させる場合, 処理能力が高い端末へはマウスの軌跡をすべて伝播するが, 処理能力が低い端末に対しては途中の軌跡を省略し, 最終的なオブジェクトの位置を確定するメソッド呼び出しのみを伝播するようにすることで, 処理の遅延を防ぐことができる.

本稿では, ノードで発生したメソッド呼び出しの中から, アプリケーション開発者が Java の機能の一つであるアノテーションを用いて指定したものを省略することにより, 処理能力が異なる端末間でも効率的なオブジェクトのふるまい同期を可能とする手法について述べる.

2. 複製オブジェクト分散環境

本節では, 複製されたオブジェクトが各ノード上に配置されている環境において, それらのオブジェクトのふるまいの同期手法とノード間の通信手法について述べる.

2.1 複製オブジェクトのふるまい同期

協調作業支援システム開発基盤 CUBE は, 複製計算モデルにもとづき, 各ノード上にオブジェクトの複製を配置し, それらのオブジェクトの間でふるまいを同期させるオブジェクトミラーリングという機能を提供する. この機能では, あるノードであるオブジェクトのメソッドが呼び出されると, そのメソッド呼び出しの情報を格納したメッセージが他ノードへ伝播される. そして, メッセージを受信した他ノードでも, 同じオブジェクトの同じメソッドが呼び出される.

2.2 複製オブジェクトのふるまいの伝播

メッセージを伝播する手法として, さまざまな通信モデルが考えられる. 1 対多通信を実現する場合, ローカルネットワークであればブロードバンドやマルチキャストを利用できる. 一方で, グローバルネットワークであれば, Firewall や NAT が混在するため, ノード間で直接通信を行うことができない場合がある. 一般に Firewall や NAT を越える方法として, 中継サーバを利用して通信を行うものがある.

サーバを介して通信を実現するために, 各ノードはメッセージをサーバに送信し, それを受信したサーバはそのメッセージをキューに格納する. 一方で, 各ノードはサーバに一定間隔でリクエストを行い, それを受信したサーバがキューに格納しているメッセージをそのノードにレスポンスとして返す.

3. 端末間の処理能力を考慮したメソッド呼び出しの伝播

オブジェクトミラーリングを処理能力に差のある端末間やネットワークの帯域に差のある端末間で実現するとき, すべてのメソッド呼び出しを処理できない場合や, すべてのメソッド呼び出しを伝播できない場合がありうる.

このような問題を解決するために, ノード間のオブジェクトのふるまいを完全に同期させるのではなく, アプリケーションにより許容される範囲で, ふるまいの同期を省略する. 本節ではメソッド呼び出しの中で, どのようなものが省略可能かについて述べる. また, そのようなメソッド呼び出しを判別し, 伝播しないようにする手法について述べる.

3.1 上書き可能なメソッド呼び出し

メソッドの性質によっては, あるメソッドが連続して呼び出されていたとしても, 最後のメソッド呼び出しのみ

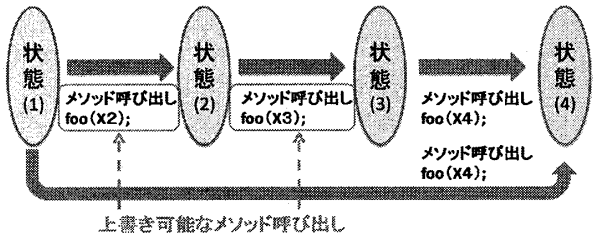

図 1: 上書き可能なメソッド呼び出しの例

†立命館大学 情報理工学部
‡立命館大学大学院 理工学研究科

を実行するだけで, すべてのメソッド呼び出しを実行した場合とオブジェクトの状態を同一にできる場合がある.

図 1 は, あるオブジェクトのメソッド foo が呼び出されたことにより, 状態が変化する様子を表している. このオブジェクトが状態 (1) のとき, メソッドが実行されると, その結果, このオブジェクトが状態 (2) に変化する. このようにメソッドが呼び出されるたびにこのオブジェクトの状態が変化し, 最終的に状態 (4) に変化するとする. ここで, このオブジェクトの状態が最初の状態 (1) のとき, もし最後のメソッド呼び出しのみを実行するだけで, 最後の状態 (4) に変化可能ならば, 途中のメソッド呼び出しを省略しても, 最後の状態に変化させることが可能と考えられる.

このように, 最後のメソッド呼び出しで途中のメソッド呼び出しを上書きすることができるため, このようなメソッド呼び出しを “上書き可能なメソッド呼び出し”と呼ぶ. また, この上書きは, 同じノードで行われた同じオブジェクトに対する同じメソッドの呼び出し同士で行うことができる.

3.2 上書き可能なメソッド呼び出しの判別

本手法では, 上書き可能なメソッド呼び出しを判別可能にするために, 開発者がそのメソッドに Java のアノテーション機能を使用して注釈 (@OverWritable) を付ける機能を提供する.

あるノードからメソッド呼び出しが伝播されるとき, それを通知するメッセージにそのメソッドが上書き可能かどうかを示す情報を付加する. メッセージはサーバに送られ, 他ノードへ伝播される前に, 上書き可能かどうかが判断される. もし, 上書き可能なメソッド呼び出しが連続していれば, 最後のメソッド呼び出しのみが残される.

3.3 上書き可能なメソッド呼び出しの削除

図 2 で示すように, サーバがメソッド呼び出しの削除を行う処理は, インプットキュー, アウトプットキュー, バッファおよびフィルタを用いて行われる.

まず, サーバは受信したメッセージをインプットキューへ格納する. また, バッファが空になったとき, インプットキューに格納されたメッセージはバッファへ送られる. さらに, アウトプットキューが空になったとき, バッファに格納されたメッセージはアウトプットキューへ送られる. そして, サーバはリクエストを受信したとき, アウトプットキューに格納しているメッセージをレスポンスとして返す.

上書き可能なメソッドが削除される仕組みについて, 図 3 の例を用いて説明する. バッファからメッセージを取り出すとき, メソッド毎にバッファ内の上書き可能なメソッド呼び出しの中で最後に格納されたものを検索する. メソッド A の場合はメソッド呼び出し A3 であり, メソッド B の場合はメソッド呼び出し B3 である. これらのメソッド呼び出しがフィルタを通過する. また, メソッド C は上書き可能なメソッド呼び出しではないので, すべてのメソッド呼び出しがフィルタを通過する. 結果として, バッファから取り出されたメソッド呼び出しの中で A3, C1, C2 と B3 がフィルタを通過し, 順番にアウトプットキューへ格納される.

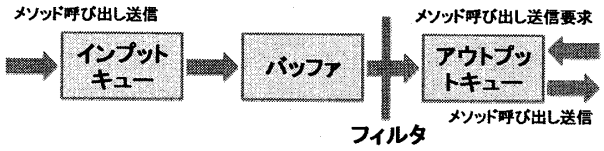

図 2: システム構成

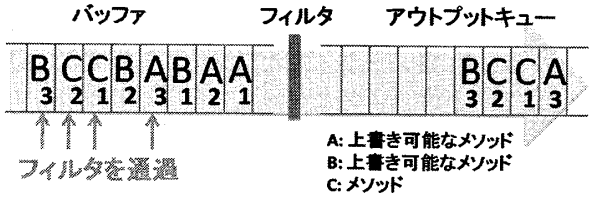

図 3: 上書き可能なメソッド呼び出しの削除の例

4. 適用例

本手法の適用例として, 図 4 のようなスライドパズルアプリケーションを構築した. ピースの移動の軌跡は, ピースを移動させるメソッドを連続して呼び出すことで表示される. もしこの軌跡を省略し, 一度にピースを隣のマスへ移動させたとしても, ピースの最終的な位置に矛盾は発生しなので, ピースを移動させるメソッドが上書き可能メソッドとして定義されている.

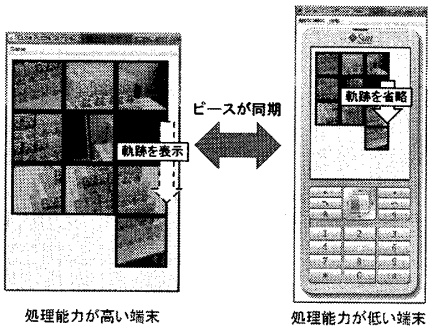

図 4: スライドパズルアプリケーション

5. おわりに

本稿では, 端末間の処理能力を考慮して, 効率的な複製オブジェクトのふるまい同期を実現する手法について述べた.

今後は, クライアントサーバモデル以外の通信モデルにも本手法を適用する方法について考察していく. また, 適用可能なアプリケーションを拡大するために, メソッド呼び出しをすべて省略するのではなく, 省略する度合いを変化させることを可能にする方法について検討する.

参考文献

[1] Shogo Noguchi, Hideyuki Takada, “CUBE: A Synchronous Collaborative Applications Platform Based on Replicated Computation”, CollabTech2009, 2009.