適応型ストーレジにおける複製管理トークンの移動方式

中村 元紀　　井上 知洋　　久保田 稔

日本電信電話株式会社 ＮＴＴ未来ねっと研究所

1 はじめに

あらゆる場所にコンピュータ (ノード) が遍在し，それらがネットワークを介して接続されることにより，ユーザに対して様々なサービスを提供する，ユビキタスサービスの実現が期待されている．ユビキタスサービスを提供するノードには，PDA や携帯電話のようにユーザが持ち歩くものが多く，また今後は自律的に移動するロボットのようなノードの普及も考えられる．このように移動する可能性が高いノード間のネットワークとしては，アドホックネットワークのようにネットワークトポロジが動的に変化するネットワーク環境が考えられる．筆者らはこのような環境で安定的に複数のノードで共有するデータを利用可能とするための適応型オンラインストーレジ CAOSS (Client-based Adaptive On-line Storage System) を提案している [1].

CAOSS ではデータの複製を適応的にネットワーク内に分散配置することにより，ネットワークトポロジが変化してもデータの読み出しが可能となる確率を高めている．また，分散配置された複製の中の一つに管理権限 (管理トークン) を与え，その複製が責任を持ってデータの更新を行うことにより，分散された複製データが他の複製を無視して更新されることが無いようにしている．一つの複製のみに管理トークンを持たせることにより，複製の管理手順を簡潔にし，更新処理遅延を削減している．

一方，前述のようなネットワークトポロジが動的に変化する環境では，このような管理トークンを一つのノードに固定的に割り当てるのは望ましくない．本稿ではこのようなネットワーク環境において，データの利用形態やノードの状況を考慮して管理トークンを複製間で移動する方式について述べる．

2 データの利用形態

本稿では以下のようなデータの利用形態を考える (図 1).

- ユーザは任意の端末をノードとして利用し，アドホックネットワークを構成する．この場合，ノード間のリンクが動的に変更されたり，頻繁に電源を落とすなどノード自体が不安定な場合がある．

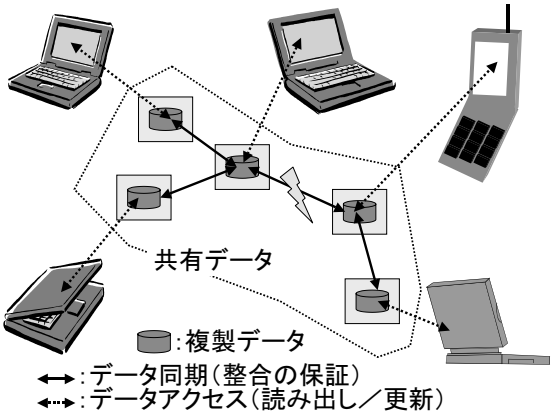

図 1: 利用例

- 複製データの利用アプリケーション例として，オンラインゲームの共有データを対象とする．特に本稿ではネットワーク上でパズルを解くことを競い合うようなゲームを考える．すなわち，ネットワーク上に共有されたパズルのデータがあり，解けると思ったユーザはそのパズルデータを書き換えて解答に近付ける．このような枠組を協調作業に適用することも可能である．

- データアクセスの種類は読み出しと更新 (新規生成や削除も含む) のいずれかとし，ローカルな複製データにアクセスする．

- データの更新時に複製データ間の整合を保証するレベルは，アプリケーションの要求条件によって異なる．本稿の例では，ゲーム当事者間では複製データ間の整合が非常に重要であることとする．ゲームの観戦者が読み出す複製データに対しては必ずしも厳密な整合は必要ではない.

- データアクセスするユーザは動的に変化する．本稿の例では，一人の参加者がデータ更新を連続して行う傾向にあり，そのような参加者が時間とともにランダムに入れ替わる．

3 CAOSS

本節では CAOSS の概要を示す.

CAOSS を構成するノード上には CAOSS マネージャ (以下 **Cmgr**) が存在し，それらが協調してデータの複製

A Method to Move Replicas Management Token in an Adaptive Storage System
Motonori NAKAMURA, Tomohiro INOUE and Minoru KUBOTA
NTT Network Innovation Laboratories, NTT Corporation
3-9-11 Midori-cho, Musashino-city, Tokyo, 180-8585, JAPAN
motonori@ma.onlab.ntt.co.jp

管理を行う．データアクセス及び複製管理は以下のように行う (図 2)．なお，CAOSS は複製間のデータの整合に関して様々なポリシを扱うことができるが，以下では説明の簡単のため強い一貫性を満たして整合をとる場合について述べる [2].

- クライアント (client) が Cmgr にデータの生成を要求すると，Cmgr はそのノードにオリジナルのデータであるコアデータ (core data) を生成する (図 2(a))．コアデータを管理する Cmgr を (そのデータに対する) **コア Cmgr** と呼ぶ．コアデータはそのデータの複製に関する管理トークンの役割を果たす.

- コアデータの無いノードでクライアントからのデータの読み出しを受けた Cmgr は，そのデータに対するコア Cmgr からデータを読み出し，クライアントに結果を返すと共にローカルに複製データ (replicated data) を生成する (図 2(b))．複製データを管理する Cmgr を (そのデータに対する) **複製 Cmgr** と呼ぶ．コア Cmgr はそのデータに対する全ての複製 Cmgr のアドレスを記憶する．更に各複製データには有効期間を設定する.

- データの更新要求はそのデータのコア Cmgr まで転送される．コア Cmgr はそのデータに対する全ての Cmgr に更新の開始を通知する．それを受けとった複製 Cmgr は複製データをロックし，コア Cmgr に応答を返す．全ての複製 Cmgr から応答を受けとったコア Cmgr は，コアデータに更新を実行し，クライアントに結果を返すと共に，再び全ての複製 Cmgr に更新の実行を要求する．更新の実行を受けとった複製 Cmgr は複製データに更新を実行してロックを解除する (図 2(c))．なお更新の開始を受けとった複製 Cmgr が応答を送信して一定時間内に更新の実行要求を受けとらなかったら複製データを破棄する.

- 複製 Cmgr はデータの有効期間が切れる直前にコア Cmgr に対して有効期間の延長要求を送信する．コア Cmgr から有効期間の延長応答を受けとらずに有効期間が切れたら，複製データを破棄する.

4 コア移動方式

本節では CAOSS において，2節で述べたようなデータの利用形態を持つアプリケーションに対して，データアクセスの遅延を削減し，データの可用性を高めるための，管理トークンの移動方式を提案する.

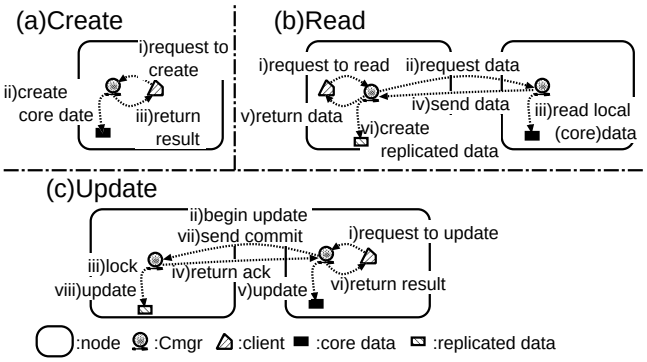

図 2: CAOSS の動作概要

4.1 要求条件

CAOSS では，管理トークンを入れ替えることによりコア Cmgr を移動することが可能である．本稿で対象とするコア移動方式は以下の要求条件を満たすべきである.

1. 最終的にコア Cmgr がデータ更新要求を受け付けるため，できるだけ更新要求の発生するノードにコア Cmgr を配置する.
2. ユーザがアプリケーションの利用終了を宣言した場合，データ更新する可能性は低くなるので，そのノードにコアデータが存在しないようにする.
3. リンクが不安定なノード間でコア移動をする場合，コア移動のためのメッセージが紛失する可能性がある．その場合，メッセージの再送などで対応する必要があるが，コアが重複して存在する場合を避ける必要があり，簡単には解決しない．よって，不安定なリンクを経由したコア移動は積極的に行わない.
4. コアの移動が頻繁に起こると本来のデータアクセス処理に悪影響を及ぼす可能性があるため，必要以上のコア移動は避ける． 特に，特定ノード間で頻繁にコアを移動し合うのは避ける.
5. 不安定なノードにコアが存在すると，データの可用性が低下する可能性があるため，不安定なノードにはコアを移動しない.

4.2 準備

移動の実行トリガを判断するために，各 Cmgr は定期的に以下の情報を収集しておくものとする.

St(Stability): 自ノードの安定度．例えば各 Cmgr が存在するノードの起動時間や，有線ネットワーク上の特定ノードとの通信成功確率のこと.

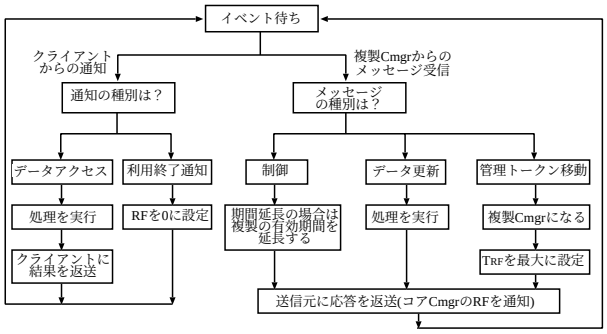

図 3: コア Cmgr の動作フローチャート

CS(Communication Status): 複製 Cmgr とコア Cmgr との通信状況．例えば通信遅延や通信失敗確率．

RF(Request Frequency): Cmgr が一定時間内にクライアントから受ける要求の頻度．ここでは特に更新要求のみを考える．

CS と RF はデータごとに把握する必要があるため，情報の収集に多くの処理が必要になるが，実際には CAOSS ではいくつかのデータをまとめたデポという単位で管理する [2] ため，必要な処理量を抑えることが可能である．本稿では簡単のため，データごとに独立に管理するものとして説明している．

St の値がある閾値より小さい場合，そのノードの Cmgr はコア Cmgr にはなれないものとする．この閾値を T_{St} とする．具体的には，各 Cmgr が安定度を調査した際にその値が T_{St} よりも小さければ， RF を 0 に設定する．

4.3 コア Cmgr の動作

コア Cmgr はクライアントからの通知，及び複製 Cmgr からのメッセージを受信して，それらに応じた処理を行う．コア Cmgr の動作フローチャートを図 3に示す．

コア Cmgr が管理トークンの移動要求を受けた場合は，移動が可能かどうかを判断する必要があるが， Cmgr 間の認証方法などは今後の課題であり，基本的に更新処理中など移動が不可能な場合を除いて移動要求には応じることとする．

なお，コア Cmgr が複製 Cmgr へ何らかの応答メッセージを送る際には，自身 (コア Cmgr) の RF をピギーバックして通知するものとする．

4.4 複製 Cmgr の動作

各複製 Cmgr は以下の値を保持するものとする．

- コア Cmgr の RF(RF_C)．これは何らかの応答メッセージにピギーバックして送られてくる．

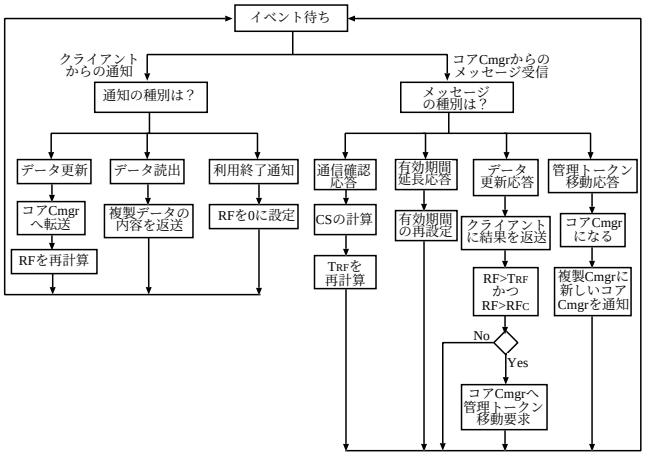

図 4: 複製 Cmgr の動作フローチャート

- コア移動を要求するかどうか判断するための閾値 T_{RF}．これは CS を使って計算する．具体的には，前回計算した T_{RF} と，今回調査した CS に一定値を掛けた値を比較し，前回の T_{RF} の方が大きければそれらの中間値を，小さければ今回計算した値を，新しい T_{RF} とすれば良い．この場合， CS が安定して小さい場合のみ， T_{RF} も小さくなる．

複製 Cmgr はクライアントからの通知，及びコア Cmgr からのメッセージを受信して，それらに応じた処理を行う．複製 Cmgr の動作フローチャートを図 4に示す．

本稿では強い一貫性を満たした複製データのみを対象とする (3節参照) ため，クライアントからのデータ読み出しの要求に対しては，ローカルな複製データをクライアントに返送すれば良い．一方データ更新要求はコア Cmgr に転送してから， 3節で述べたように更新を実行する．

4.5 コア移動の例

図 5にコア移動時の例を示し， Cmgr が管理する情報の変化の様子を示す． RF としては過去の 5 分間に受信した要求の数， CS としては通信遅延 (ミリ秒単位)， ST としてはノードの起動時間 (秒単位)， T_{ST} を 300 とする．

時刻 t_0 では A がコア Cmgr であり， B が複製 Cmgr である．その後時刻 t_1 の時点までに B で更新要求が多く発生し， A の RF が 5， B の RF が 15 となったとする．t_2 で A の RF_C が通知されると， B では RF が RF_C 及び T_{RF} よりも大きく， ST も T_{ST} より大きいため，コアの移動を要求する． A は移動に応じると共に，自身の T_{RF} を大きな値に設定し，しばらくはコア移動を要求しないようにする (t_3)．コア移動成功後，全ての複製 Cmgr に新しいコア Cmgr のアドレスと RF_C が通知される (t_4)．

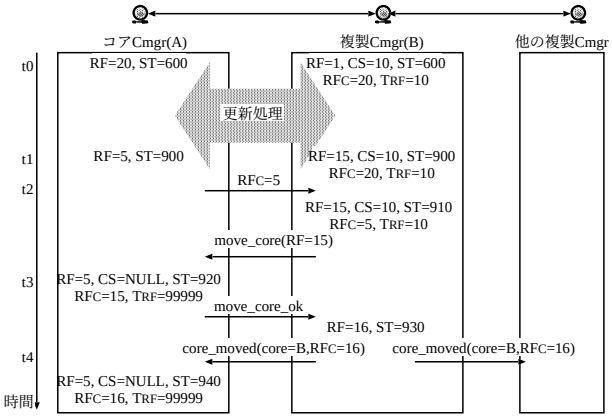

図 5: コア移動時の各管理情報の変化の様子

5 効果

本節では 4節で述べたコア移動方式が 4.1節で述べた要求条件を満たしていることを確認し，提案方式の効果について述べる．

- 提案方式では，一定期間内の更新頻度 (RF) が高いノードにコアを移動する．一方 2節で述べた通り，想定アプリケーションでは一人のユーザが連続してデータ更新を続けるため，一定期間を十分短くとれば，更新要求が発生するノードにコアデータが存在する可能性が高い (要求条件 1).

- クライアントからの利用終了通知に基づき，記録している利用頻度 (RF) を 0 にする．その結果，そのノードにコアデータがあっても別の利用頻度が高いノードにコアが移動することとなる (要求条件 2).

- 複製 Cmgr はコア Cmgr との通信状況に応じてコア移動を要求する際の閾値 (T_{RF}) を増減する．通信状況が悪い場合は閾値を高く設定するため，データ更新頻度が非常に高くならないとコアを移動しない．データ更新頻度が非常に高ければ，このようなリンクを介してコアを移動する効果はある (要求条件 3).

- コア Cmgr は移動要求に応じてコアを移動した場合，T_{RF} を最大値に設定する．T_{RF} は CS の調査時に再計算し，短時間に小さくなることはないので，一度管理トークンを手放した Cmgr が短時間でコア移動を要求することはない (要求条件 4).

- 複製 Cmgr は自身のノードの安定度が低い場合は，更新要求の頻度が高くても TF を 0 にするのでコア移動を要求しない．また，コア Cmgr は自身のノードの安定度が低くなったら，自身の要求頻度 (RF) を 0 とするため，結果的に他の要求頻度が高い複製 Cmgr へコアを移動することとなる (要求条件 5).

6 関連研究

モバイル分散 DB において，ネットワークから離脱する端末にデータ更新の権限を移動する方式が提案されている [3]．これは，通常データ更新をする際には複数の複製のうちある決められた集合 (レフェリ) の許可が必要となるが，離脱端末に対してはユーザがその端末上の複製のみをレフェリとして定義することで，ネットワーク離脱時のデータ可用性を高めているものである．しかし，ユーザの指示による移動をサポートしているのみである．

アドホックネットワークでデータの整合などの調停機能を持つ端末を，ネットワークトポロジの中心に配置する方式が提案されている [4]．調停端末がネットワークの中心に存在すれば，参加者への要求メッセージ及び応答の通信遅延を最小とすることができる．しかし，ネットワークの最大ホップ数が小さく，要求が連続する場合，要求元から調停端末への要求送信と応答の遅延の影響の方が大きくなくなる可能性がある．本稿での提案方式でも複製の中でネットワークトポロジ的に中心に存在する Cmgr にコアを移動することによる遅延削減効果も考えられるため，今後定量的な評価を行っていく予定である．

7 まとめ

本稿では適応型オンラインストーレジ CAOSS における，複製データの管理トークンであるコアデータの移動方式について述べた．提案方式では，データの利用頻度，コア Cmgr と複製 Cmgr の通信状況，及びノードの安定度を考慮してコア移動の要求を行うかどうかを決定するため，あるユーザがデータ更新を連続して要求するようなアプリケーションに対して，データの更新遅延を削減しつつ，データの可用性を高めている．また，ユーザ指示によるコアの移動もサポートしている．今後は提案方式の定量的な評価を進める予定である．

参考文献

[1] 中村 元紀, 井上 知洋, 久保田 稔, ”環境適応型オンラインストーレジにおける複製管理方式の評価”, 信学技報 IN2002-108, pp.31–36, Nov. 2002.

[2] 井上 知洋, 中村 元紀, 久保田 稔, ”動的なネットワーク環境に適した適応型オンラインストーレジシステムの提案”, マルチメディア通信と分散処理研究会 110-10, pp.55–60, Nov. 2002.

[3] Q.R. Wang and J.F. Pâris, ”Managing Replicated Data Using Referees”, Proc. Workshop Mobility and Replication, ECOOP ’95, Aug. 1995.

[4] 鈴木 貴也, 石原 進, 水野 忠則, ”アドホックネットワークにおけるトポロジ依存機能の動的再配置”, 情処論, Vol.43, No.12, pp.3959–3969”, Dec. 2002.