モバイルエージェントの同時集中移動時における通信量の削減方式

東野 正幸*，高橋 健一，川村 尚生，菅原 一孔
(鳥取大学大学院 工学研究科 情報エレクトロニクス専攻)

A Method for Reducing Data Traffic on Concentrated Agent Migrations

Masayuki Higashino*, Kenichi Takahashi, Takao Kawamura, Kazunori Sugahara
(Graduate School of Engineering, Tottori University)

1. はじめに

モバイルエージェントとはノード間を移動可能な自律的なソフトウェアである．モバイルエージェント技術は分散アプリケーションの設計パラダイムとして通信の効率化や複雑な設計の簡素化など幅広い分野で利用されている．

エージェントはノード間を移動する際に処理対象のデータだけでなく処理に利用するプログラムコードやプログラムの実行時状態の転送を必要とする．このためエージェントの移動はデータ転送だけを目的とした通信プロトコルよりも通信量を増加させる．エージェントの移動に伴う通信量を削減するためにプログラムコードをノードへキャッシュする移動方式が提案されている[(1)(2)]．しかし，複数のエージェントが異なるノードから１つのノードへ短時間に集中して移動（以下，同時集中移動と呼ぶ）する場合に，キャッシュミスが同時に発生する事で同じプログラムコードが重複して転送され，通信量を十分に削減できない問題がある．

そこで本稿では，同時集中移動する各エージェントが同じプログラムコードを持っている場合に，各プログラムコードの転送元ノードを１つに限定することで，同じプログラムコードの重複した転送を抑制する方式を提案する．提案方式では各プログラムコードの転送元ノードを１つに決定する問題を組合せ最適化問題の一種である一般化割当問題として解く．これにより，各移動元ノードによって異なるプログラムコードの転送可否の条件や通信速度に応じたプログラムコードの転送量の配分を考慮した，エージェントの同時集中移動時における通信量の削減が可能となる．

2. 提案方式

〈2･1〉 エージェントのモデル　一般にエージェントの移動はエージェントを構成するデータを移動元ノードから移動先ノードへ転送することで実現されている．そこで，エージェントを構成するデータ構造を定義する．エージェントは，実行時状態領域，アプリケーション領域，及びプログラムコード領域から構成される．実行時状態領域はエージェント固有のコールスタックやプログラムカウンタといった動作状態を表現するデータを保持する．アプリケーション領域はエージェント固有の設定や処理対象といったデータを保持する．プログラムコード領域はエージェントの動作が記述された複数のプログラムコードを保持する．

エージェントを構成するデータを移動元ノードから移動先ノードへ転送するまでの一連の手続きをまとめて移動セッションと呼ぶ．同時集中移動が発生する時，各移動セッションは時間的な重なりを持って並行処理される．以上は既存方式[(1)(2)]のモデルを包含している．

〈2･2〉 プログラムコードの転送元ノードの限定　提案方式では，複数の移動セッションが時間的に重なり，同じプログラムコードを複数の移動元ノードから転送可能な場合に，各プログラムコードの転送元のノードを１つに限定する．これを次式で表現する．

$$\sum_{i=1}^{m} x_{ij} = 1,\ x_{ij} \in \{0,1\},\ \forall i \in I,\ \forall j \in J \quad \cdots\cdots (1)$$

ここで，$I = \{1, ..., m\}$ は時間的に重なった m 個の移動セッションの移動元ノード集合，$J = \{1, ..., n\}$ は時間的に重なった移動セッション全体で転送する必要がある n 個の重複なしプログラムコード集合，x_{ij} は移動元ノード i からプログラムコード j を転送する場合に 1，そうでない場合に 0 の値をとる $0-1$ 変数である．

〈2･3〉 各移動セッションの通信速度の差異　一般にコンピュータネットワークではノード間の通信速度に差が存在する．これを考慮しなければ，通信速度の遅い移動セッションでのデータサイズの大きなプログラムコードの転送が時間的なボトルネックとなる．そこで，各移動セッションについて，プログラムコードの転送に利用する通信量の上限を設定して通信速度と比例させることで，通信時間を均等化する．また，各移動セッションで転送するプログラムコードの総データサイズは，この上限を超えないようにする．これを次式で表現する．

$$s_i = \frac{\sum \{a_1, ..., a_n\}}{\sum \{b_1, ..., b_m\}} b_i,\ s_i \geq \sum_{j=1}^{n} a_j x_{ij} \cdots\cdots\cdots (2)$$

ここで，b_i は移動元ノード i と移動先ノードとの通信速度，s_i は移動元ノード i から移動先ノードへのプログラムコードの転送に利用可能な通信量の上限，a_j はプログラムコード j のデータサイズである．

〈2･4〉 一般化割当問題に対する適用　エージェントは自身が保持しているプログラムコードを追加・変更・削除することで自身の動作を変化させる．このため，同時集中移動する各エージェントが保持しているプログラムコード集合の包含関係が完全一致するとは限らない．したがって，プログラムコード毎に転送可能な移動セッションと不可能な移動セッションが存在する．そこで，移動元ノード i が，プログラムコード j を転送可能な場合に 0，そうでない場合に ∞ となるコスト $c_{ij} \in \{0, \infty\}$ を定義し，各移動セッションのコストの総和 $\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij}$ を最小化する x_{ij} を求める．ただし，式 1 と式 2 を制約条件とする．このような問題は一般化割当問題として解く事ができる．

〈2･5〉 制約条件の緩和　一般にプログラムコードのデータサイズは任意であり，データサイズの大きなプログラムコードが通信速度の遅い移動セッションからのみしか転送できない場合，移動セッションの通信量上限 s_i を超過してしまい制約条件の式 2 を満たせない場合がある．そこでラグランジュ緩和法を用いて制約条件である式 2 を緩和する．各移動元ノード i との移動セッションにおいて，転送するプログラムコードのデータサイズの総和 $\sum_{j=1}^{n} a_j x_{ij}$ と通信量上限 s_i との差が 0 に近づくようにする．以上により次式のラグランジュ緩和問題を得る．

$$\begin{aligned} \text{min.} \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{i=1}^{m}\left|\sum_{j=1}^{n} a_j x_{ij} - s_i\right| \\ \text{s. t.} \quad & \sum_{i=1}^{m} x_{ij} = 1,\ \forall i \in I,\ \forall j \in J, \\ & x_{ij} \in \{0,1\},\ c_{ij} \in \{0, \infty\} \end{aligned} \quad \cdots (3)$$

提案方式では，移動セッションの時間的な重なりが発生する毎に式 3 を用いて x_{ij} を求める事で各プログラムコードの転送元ノードを決定する．

〈2･6〉 アルゴリズム　同時集中移動では，複数の移動セッションの開始時刻が短時間に集中する事から，式 3 は短時間で処理される必要がある．しかし，一般化割当問題は NP 困難である事が知られており，最適解を得る列挙法では $O(m^n)$ の計算量を要する．そこで提案方式では，近似解を $O(mn)$ の計算量で得る欲張り法を用いる．

3. 評　価

提案方式の有効性を評価するために，同時集中移動数の増加に伴う通信量の変化について，既存方式と従来方式との比較実験を行う．既存方式には，キャッシュ無し方式，キャッシュ方式 (1)，及びキャンセル方式 (2) を用いる．キャッシュ無し方式はエージェントの移動時にプログラムコードをキャッシュしない．キャッシュ方式はエージェントの移動時にプログラムコードをキャッシュする．また，異なるエージェント間でキャッシュを共有できる．ただし，同時集中移動時にキャッシュミスが同時に発生することで同じプログラムコードが重複して転送される．キャンセル方式はキャッシュ方式にプログラムコード転送のキャンセル機能を追加した方式である．キャッシュミスが同時に発生する事でプログラムコードの転送命令が複数の移動元ノードへ発行された場合，移動先ノードは，いずれか 1 つの移動元ノードからプログラムコードを取得した時点で，他の移動元ノードにキャンセル命令を発行する．しかし，キャンセル命令が間に合わない場合はプログラムコードが重複して転送される．

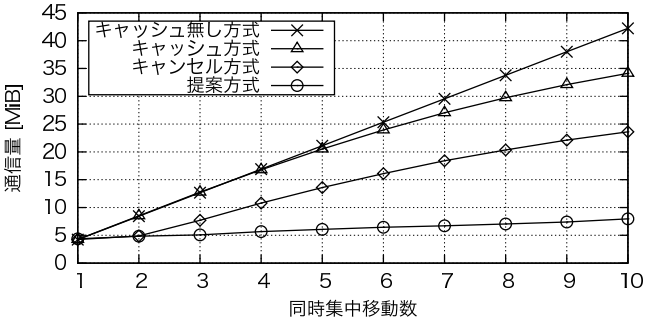

図 1.　同時集中移動数に対する通信量の変化

比較実験では，エージェントが持つ実行時状態領域を 72 KiB，アプリケーション領域を 0 KiB，プログラムコードの数を 4096 個，各プログラムコードのサイズを 1 KiB，各エージェントの移動開始時間の差を 70 ms，各ノード間の通信速度を 100 MBit/s とし，複製したエージェントの同時集中移動数を 1 から 20 に変化させた場合の通信量の変化を算出した．

実験結果を図 1 に示す．キャッシュ無し方式では，キャッシュ機能が無いため，同時集中移動数に比例して通信量が増加している．キャッシュ方式では，各移動セッションでキャッシュミスが同時に発生することでプログラムコードが重複転送され，同時集中移動数が 10 の場合においてキャッシュ無し方式に対して通信量を約 19%しか削減できていない．キャンセル方式では，キャンセル機能により重複転送を抑制できているが約 44%の削減にとどまっている．一方，提案方式では 81%と既存方式よりも大幅に削減できている．

4. おわりに

複数のモバイルエージェントが異なるノードから 1 つのノードへ短時間に集中して同時に移動する場合における通信量を削減するエージェントの移動機構を提案した．提案方式では，同時に移動するエージェントが同じプログラムコードを持っている場合に，各プログラムコードの転送元ノードをいずれか 1 つに決定して転送することで，同じプログラムコードの重複した転送を抑制する．同時集中移動数の増加に伴う通信量の変化について既存方式と提案方式との比較実験を行ったところ，同時集中移動数が 10 の場合に，既存方式が約 44%の通信量を削減するのに対して提案方式は 81%の通信量を削減できた．

文　献

(1) Braun, P. and Rossak, W.: *Mobile Agents: Basic Concepts, Mobility Models, and the Tracy Toolkit*, Morgan Kaufmann Publishers Inc. (2005).

(2) 東野正幸, 高橋健一, 川村尚生, 菅原一孔: キャッシュによるエージェントの移動効率化, 電子情報通信学会論文誌, Vol. J96-D, No. 7, pp. 1576–1584 (2013).