L-018 マルチエージェントに基づく分散型 e-Learning システムにおけるバックアップと障害復帰

Backup and Recovery Scheme for Multi-Agent-based e-Learning System

木下 俊吾 † 本村 真一 † 川村 尚生 ‡ 菅原 一孔 ‡
Shungo KINOSHITA Shinichi MOTOMURA Takao KAWAMURA Kazunori SUGAHARA

1. はじめに

近年，e-Learning システムと称されるコンピュータを用いた学習システムが広く一般的に普及している．我々はこの中でも，教師を必要とせず学習者が好きな時に学習できる「教師無し非同期 Web-based traning（以下 WBT）」に注目し研究を行っている．

従来の教師無し非同期 WBT はクライアント/サーバモデルで構築されており，サーバへの負荷や機能集中などにより一般的に拡張性や頑健性に乏しい．

我々は分散型 e-Leraning システムを提案し実装してきた [1]．提案システムは我々の開発した prolog 基盤のモバイルマルチエージェントフレームワークである Maglog[2] を用いている．以下にこのシステムの持つ 2 つの特徴を挙げる．1 つ目は，P2P 構造を基盤としておりシステムに参加する全てのコンピュータ（以下ノード）はクライアントとサーバの両者の役割を担っている点である．新規参加ノードは既参加ノードより幾らかの学習コンテンツを受け取りシステムの一部となる．そして他のノードより学習コンテンツの要求があった場合は要求先にコンテンツの提供を行う．2 つ目は，本システムにおける各学習コンテンツは単なるデータファイルではなく，機能を持ったエージェントで実装されている点である．つまりこのエージェントが問題・解説・採点機能などを提供することで，コンテンツだけでなく従来サーバが提供していた機能の分散も実現している．

本システムは問題と機能の分散によるシステムの負荷分散だけでなく，ノード障害が発生した際にも残りのノードによりシステムのサービスを継続できるという頑健性を実現している．しかし障害が発生したノードが管理していた学習コンテンツは一旦失うと学習することができなくなる．

本研究において，この問題の解決を図るためエージェントのバックアップをノードへ分散する事を考える．我々はこのバックアップ・障害復帰の方法として，ノード単位とカテゴリ単位の 2 つの方法を提案する．どちらの方法も 1 ノードの障害からの復帰が保障され，以下に示す条件下においてはそれ以上のノードの同時障害から復帰する事が可能となる．

2. 提案 e-Learning システムの概要

提案システムでは全ての問題が “英語/文法”，“数学/統計” という様に階層構造を持つカテゴリに分類される．

提案システムではシステム起動時において，初期ノードがシステムの全てのカテゴリを有している．他のノードが新規参加する際は，初期ノードより幾らかのカテゴリを受け取る．また離脱する際は，システムに参加しているノードのいずれかにカテゴリを返還する．このように各ノードの参加と離脱によってカテゴリはシステムの全てのノードに分散される．

提案システムでカテゴリの分散を実現するため，我々は P2P ネットワークを構築した．この P2P ネットワークは $(key, value)$ の組を格納する仮想座標空間を持つ．各 key の値をハッシュ関数に与えることで得られるポイントをそれぞれ座標空間に配置する．提案システムではカテゴリ名を key とし配置している．そして各ノードはこの座標空間の一部を管理しており，その管理領域に含まれるカテゴリを管理している．但し実際の問題のセットはカテゴリ単位で管理されている．

我々の P2P ネットワークは図 1 で示されるような 2 次元直交座標空間で構成される．図 1 はノード C がシステム新規参加した際の様子を示している．ノード C が参加する前は，ノード A，B がそれぞれ座標空間を半分ずつ管理していた．つまりノード A は “数学/幾何学”，“数学/統計”，“歴史/ローマ”，ノード B は “英語/文法”，“英語/リーディング”，“歴史/日本” のカテゴリを管理していた．ノード C がシステムに参加するとランダムに選ばれたノード，ここではノード B がノード C へ座標空間の半分を与える．よってノード C は与えられた座標空間に含まれる “歴史/日本” のカテゴリをノード B より受け取り管理する．

これまで提案システムにおける学習コンテンツの分散手法について述べてきた．しかし WBT という面において，これまでサーバが提供してきたような採点や解説機能を分散する必要がある．我々はこの機能の分散をモバイルエージェント技術により実現した．つまり提案システムにおいて，各学習コンテンツは単なるデータではなく採点や解説機能を有したエージェントとして実装されている．これにより学習コンテンツと WBT としての機能の分散を実現している．

3. 参加・離脱手順

アルゴリズムの説明に先立ち以下を定義をする．

定義 1 提案 P2P ネットワークにおいて，2 つのノードが管理する分散ハッシュテーブル上の領域の X もしくは Y 軸に重なる部分があればこれらのノードは互いに隣接ノードである．ノード n の隣接ノードの集合は $N(n)$ で表す．図 1 を例として示すと，$N(A) = nodeB, nodeC$，$N(B) = nodeA, nodeC$，$N(C) = nodeA, nodeB$ となる．

定義 2 ノード n の領域情報を $N(n)$ へ伝播する手順を $update(n)$ で表す．このノード n の領域情報はノード n と $N(n)$ の領域座標から構成される．

†鳥取大学大学院工学研究科知能情報工学専攻
‡鳥取大学工学部知能情報工学科

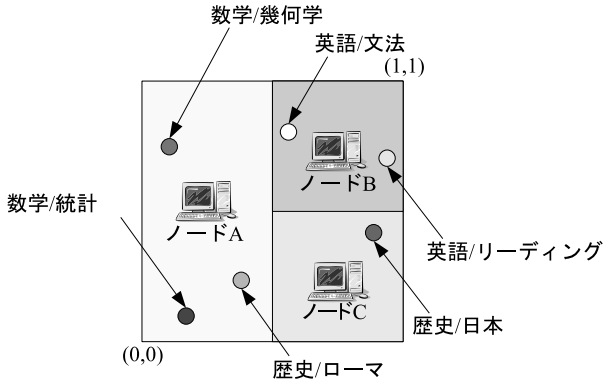

図 1: P2P ネットワーク

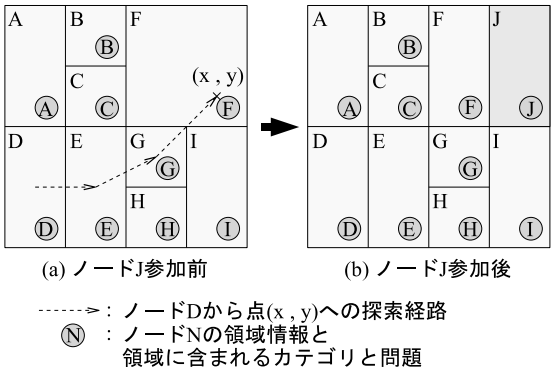

図 2: ノード J システム参加の様子

3.1 参加手順

例としてノード n のシステム参加手順を以下に示す.

1. ノード n は分散ハッシュテーブル上の座標空間よりランダムにポイント P を選択し，そのポイント P に対して参加要求を送信する．この要求は P2P ネットワークの既参加ノードを経由して送信される．この時各ノードは管理領域にポイント P を含むノード n_p に到達するまで要求を伝播する．但し，新規参加ノードは一番初めに問い合わせるノードの IP を最低 1 つは知っているものとする.

2. 要求を受けたノード n_p は管理領域を半分に分割し，ポイント P を含む領域をノード n に割り当てる．この時領域に含まれるカテゴリとその問題集合も同時にノード n に転送される.

3. ノード n は領域情報の獲得により $N(n)$ の IP アドレスを得る．同時にノード n_p は $N(n_p)$ から既に隣接ノードでないものを排除する.

4. ノード n と n_p は $update(n)$ と $update(n_p)$ をそれぞれ実行し，変化した領域情報を隣接ノードへ伝える.

図 2 にシステムへノード J が参加する例を示す．まずノード J はランダムポイント (x, y) を選択する．次にノード J は IP アドレスを知っていると仮定されるノード D に対し参加要求を行う．図 2(a) に示されるように，この参加要求は隣接ノードを通し (x, y) を領域に含むノード F まで伝播される．2 倍されたカテゴリを同一ノードが管理しなければ最後にノード F は自身の領域を半分に分割し，図 2(b) で示されるように (x, y) を含む領域をノード J に割り当てる.

3.2 離脱手順

例としてノード n のシステム離脱手順を以下に示す.

1. $N(n)$ 中にノード n の管理領域の面積に等しいノードが存在する場合はステップ 4 へ，しない場合はステップ 2 へ進む．以下このような領域の等しい 2 つのノードをノードペアと呼ぶ.

2. ノード n は $N(n)$ 中で一番面積の小さいノードへ検索要求を伝播する．この伝播プロセスはノードペアを発見するまで繰り返される．仮に発見されたノードペアをノード p，q とする.

3. ノード n と p は互いに領域を交換し，それぞれ $update(n)$，$update(p)$ を実行する.

4. ノード n と q の領域は一つに統合されノード q に割り当てられる．この時ノード n の領域に含まれていたカテゴリとその問題集合はノード q に転送される．ノード q は $update(q)$ を行う.

図 3 はノード B がシステムから離脱する様子を示している．この例では，ノード C と D のみが統合可能であるので，図 3(b) のようにノード B と D が領域の交換を行う．ここでノード B，C，D はそれぞれ上述のノード n，q，p に相当する．最後にノード B と C の領域は 1 つに統合され，図 3(c) のようにノード C に割り当てられる.

4. ノード単位のバックアップと障害復帰

4.1 ノードのバックアップ

この手法では，各ノードの隣接ノードの一つがバックアップを管理する．ノード n がノード m のバックアップを管理する場合，ノード n をバックアップノード，ノード m をオリジナルノードと呼ぶ．オリジナルノードはバックアップノードを知っており，その逆も同様である．あるノードのバックアップは，そのノードの管理領域の領域情報とカテゴリとその問題集合から成り立つ．システムの負荷分散のために，エージェント数が最小の隣接ノードがバックアップノードとして選択される．このエージェント数は常に公開されている.

ここで 3.1 で説明した参加手順をバックアップのために変更する必要がある．つまり参加ノードがバックアップ作成する手順を参加手順のステップ 4 の後に加える必要がある．しかしそれだけでは不十分であることを図 4 の例を用いて説明する．ここではノード単位のバックアップ手法を用いたシステムにおいてノード J が参加す

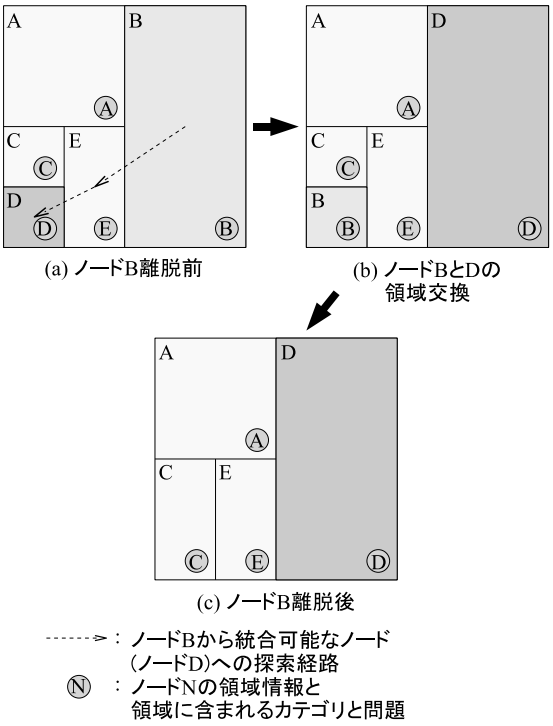

図 3: ノード D システム離脱の様子

る時の様子を示している．ノード J はノード F より領域を獲得し，結果としてノード F の領域は変化している．そのためノード F のバックアップは無効となる．そこでノード F はノード C に古いバックアップを破棄するよう要求し，更に新しいバックアップを管理するよう要求する．図 4 のノード C 上にある黒丸に白字の記号は更新されたバックアップを示している．

またバックアップは必ず隣接ノードが管理するため，領域の変化によりバックアップノードとオリジナルノードが隣接しなくなった場合は，バックアップノードはバックアップの破棄をオリジナルノードはバックアップの再生成を行う．図 4 のノード G 上の黒丸に白字の記号は再生成されたノード I のバックアップを示している．

同様に 3.2 で説明した離脱手順においても変更が必要となるが，領域変化時にバックアップ更新とバックアップ破棄が加わるだけなので，ここでの説明は割愛する．

4.2 ノード障害復帰

各ノード n_i は定期的に $update(n_i)$ を実行する．よってこの $update(n_i)$ メッセージが隣接ノードから一定時間受け取れない場合，各ノードは障害を検知する．バックアップノード n がオリジナルノード m の障害を検知した場合，以下の手順にしたがって障害復帰を行う．

1. ノード n はノード m のバックアップから一時代理ノードを生成する．
2. 一時代理ノードは離脱作業を行う．従ってノード m の領域は通常通り隣接ノードの 1 つに受け渡される．

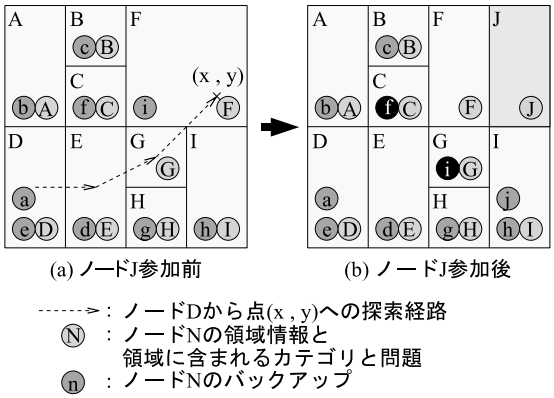

図 4: ノード単位のバックアップ手法を用いたシステムにおけるノード J 参加の様子

逆にオリジナルノードがバックアップノードの障害を検知した場合は，単純に自身のバックアップを再作成するだけでよい．

またこのアルゴリズムを用いた障害復帰は，バックアップノードとオリジナルノードの両方に障害が起きない条件下で可能となる．条件を満たしていれば複数のノードの障害からの障害復帰も可能である．

5. カテゴリ単位のバックアップと障害復帰

5.1 カテゴリのバックアップ

この手法は，全てのカテゴリが 2 倍され 2 次元直交座標空間に分散される．2 倍されたカテゴリを同一ノードが管理しない前提条件下において 2 倍されたカテゴリの一方がノード障害により失われた場合，残りもう一方がそのカテゴリをもう一度 2 倍にすることができる．それは図 5 のように 2 次元直交座標空間において 2 倍されたカテゴリの各ペアが $[0.5, 0.5]$ 点で点対称を成し，$x = 0.5$ もしくは $y = 0.5$ のライン上に配置されない場合に保証されている．

このバックアップと障害復帰手法においては今までの参加・離脱手法を変更する必要がないことに注意する．つまりこれらの手法は単にバックアップカテゴリをオリジナルカテゴリと同様に扱うことで実現している．

5.2 ノード障害復帰

この手法における障害復帰方法は以下の点を除けば，ノード単位の手法の 1 つとして同様に扱うことができる．

1. あるノードを障害から回復する責任のあるノードは，障害ノードの隣接ノードの 1 つである．
2. 失われたカテゴリの組の全てのカテゴリは一時代理ノード上に 2 倍される必要がある．

またこのアルゴリズムを用いた障害復帰も，前章で述べたアルゴリズムと同様に，バックアップノードとオリジナルノードの両方に障害が起きない条件下において，複数ノードの障害からの障害復帰が可能となる．

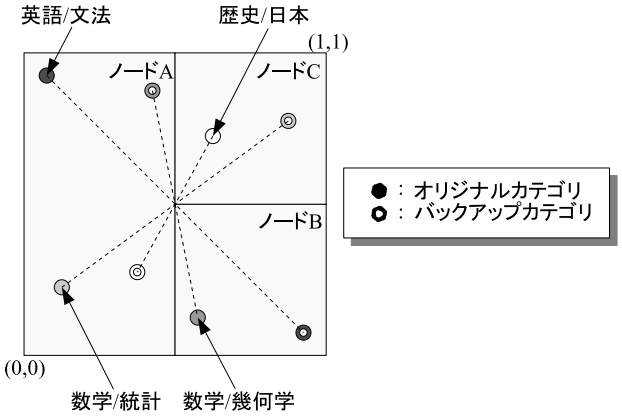

図 5: $[0.5, 0.5]$ 点で点対称配置されたカテゴリの各組

6. 実験

この章では実験結果を示す．実験環境として，Pentium4 3.0GHz の CPU と 1GB のメモリを搭載した PC30 台を 1000Base-T の LAN で接続した．全ての PC の OS は GNU/Linux（kernel version : 2.6.0）を用いた．

我々は上記の実験環境において表 1 に示される条件でバックアップを含む各ノードのエージェント数 (ノード間を移動しないエージェントは除く) を測定した．両バックアップ手法において，参加ノード数が増加すれば各ノードのエージェント平均数が減少している．ここで低い平均値は提案システムの高い負荷分散を示している．図 6 はノード単位のバックアップ手法においての結果を示しているが，カテゴリ単位のバックアップ手法においてもほぼ同様の結果が得られた．

表 1: 実験条件

参加ノード数	1-30
カテゴリ数	50
1 カテゴリ中の問題数	3

次に我々は，バックアップと障害復帰手法の有無でエージェント転送量（ノード間のパケット転送量 $[byte]$）がどのように違うのか測定した．図 7 はバックアップの有無によるエージェント転送量の比を示している．参加ノード数毎の転送量の比のばらつきは領域の分割状態やカテゴリの配置により左右されるが，実験結果はどちらのバックアップ手法においても，一定以上のばらつきはなく，参加ノード数の増加によってバックアップと障害復帰による負荷は増加しないことを示している．

7. おわりに

従来のクライアント/サーバモデルによる非同期 WBT システムは拡張性や頑健性の面で問題があった．提案システムはモバイルエージェントと P2P 技術による分散手法を用いこの問題を解決した．

今回我々は 2 つのバックアップ手法の開発を行い，提案システムにおいて，ノードに障害が起こった際にバックアップを用いて障害復帰を行うことができた．どちらの手法もバックアップノードとオリジナルノードの両方に障害が起きなければ，複数のノードの障害から復帰することができる．また分散ハッシュテーブルによるコンテンツの分散や P2P ネットワークを構築するシステムであれば，同様にこの機能を用いて障害復帰を行うことができると考えられる．

実験が示すように 2 つのバックアップ手法は，拡張性の面で分散型 e-Learning システムの利点を失うことなく実現できた．今後の課題は 2 つのバックアップ分散手法の特徴を明確にする事である．

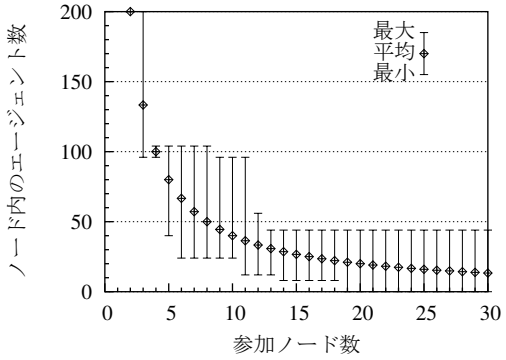

図 6: ノード単位のバックアップと障害復帰手法における各ノードのエージェント数

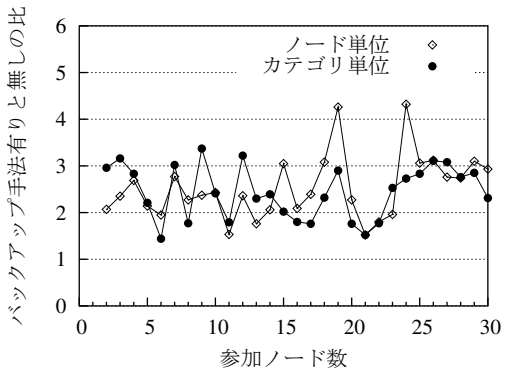

図 7: エージェント転送量の比

参考文献

[1] 川村尚生, 菅原一孔: モバイルエージェントに基づく P2P 型 e-Learning システム, 情報処理学会論文誌, Vol. 46, No. 1, pp. 222–225 (2005).

[2] Motomura, S., Kawamura, T. and Sugahara, K.: Logic-Based Mobile Agent Framework with a Concept of “Field”, *IPSJ Journal*, Vol. 47, No. 4, pp. 1230–1238 (2006).