3F3-7

高速マルチエージェントシステムによるリアルタイムデータ集計処理

High-Performance Multi-Agent System on Real-Time Data Aggregation

村田 悠也 *1 Yuya MURATA　山本 学 *1 Gaku YAMAMOTO　寺野 隆雄 *1 Takao TERANO

*1東京工業大学大学院総合理工学研究科知能システム科学専攻
Depertment of Computational Intelligence and Systems Science Interdiscplinary, Graduate School of Science and Engineering ,Tokyo Institute of Technology

Recently, Real-time Data Aggregation (RDA) of stream data is important with the advent of Machine to Machine (M2M) technology. RDA is a technology for aggregating the massive amounts of data generated in a short period of time. Therefore, RDA system performance is a major challenge. In this paper, we describe how to solve the multi-agent challenges of RDA. An agent based RDA solves the problem with such functions as pipeline processing, parallel processing, high performance data read/update of each of the agents.

1. はじめに

スマートフォンの普及とともに，スマートフォンに備えられた加速度センサー，GPS ロガー等のセンサー機器を利用したサービスが増えてきた．これらのサービスは，Machine to Machine(M2M) と呼ばれる機器間の通信に人が介在しないシステムとなっている．例えば，マップアプリケーションでは，スマートフォンで取得した位置情報をマップ上へリアルタイムに反映する．そのため，マップアプリケーションの利用ユーザー数や付随するサービス数により秒間に処理されるデータは膨大となる．

M2M では，短期間に大量データが恒常的に流れてくることを想定し，システムを開発しなければならない．近年，ストリームデータを高速に処理するストリーム処理技術やサーバー台数の増加によりスケールに性能を向上させるスケールアウト技術 (分散処理技術) の研究が注目されている [Arasu 06][Amini 06][Dean 08]．これらの技術は，発生したデータのリアルタイム処理や蓄積されたデータの高速処理に適した技術となっている．しかしながら，インタラクティブな M2M システムではリアルタイム性と蓄積データの利用の 2 つが必要とされる場合がある．

リアルタイム処理とデータ集計処理を組み合わせた処理手法としてリアルタイムデータ集計処理 (Real-time Data Aggregation, RDA) がある [山本 11]．RDA は，恒常的に発生するデータをリアルタイムに集計することで，リアルタイム処理とデータ集計処理を両立する．しかしながら，その特徴から処理性能が大きな課題となっている．そのため，RDA の実現では，ストリーム処理技術と分散処理技術を適切に連携させる技術が必要となる．そのような技術として，エージェントプログラミングモデル (Agent Programming Model, APM) に基づくインメモリデータストアが存在する [山本 08]．このデータストアは，データをインメモリで保持する機能に加え，エージェント間のメッセージング機能によりパイプライン処理・並行処理・並列処理を実現するデータ処理システムとなっている．すでに，この技術を利用して高速な RDA の実現可能性が報告されている [山本 2011]．しかしながら，APM に基づくインメモリデータストアを用いて， RDA をどのように設計するか，性能要件はどのように満たすかといった点については十分に議論されていない．

本稿では，APM に基づくインメモリデータストアを利用し，RDA マルチエージェントシステム (RDA Multi-Agent System, RDA MAS) を実現する．また，RDA MAS をランキングアプリケーションに適用し性能評価を行う．

連絡先: 村田悠也，東京工業大学大学院総合理工学研究科知能システム科学専攻，神奈川県横浜市緑区長津田町 4529，045-924-5815，murata@trn.dis.titech.ac.jp

2. マルチエージェントシステム

本稿で設計する MAS は，[山本 08] により提案されたエージェントを複数用い，協調的にデータ処理を行うシステムとなっている．図 1 は，APM によるインメモリデータストアを複数配置した場合の構成である．各エージェントは，メッセージ処理用のメッセージハンドラと排他的に管理されたデータレコードを持ち，アプリケーションロジックから送られたメッセージにより非同期に処理が行われる．

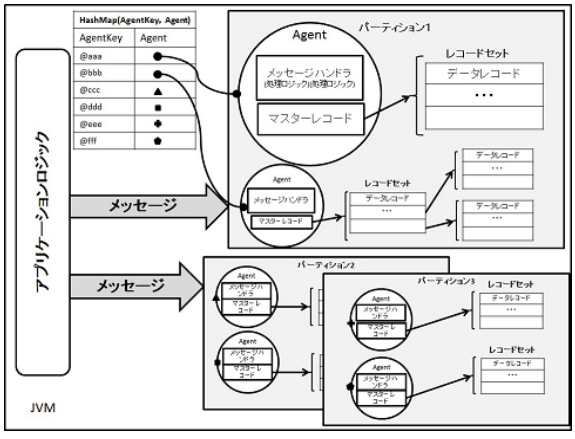

図 1: マルチエージェントシステムの構成

2.1 エージェントプログラミングモデル

2.1.1 処理モデル

エージェントは，排他的に管理されたデータを保持する．エージェントが管理するデータは，メッセージと対応するメッセージハンドラによって処理される．エージェントが，メッセージを受け取るとメッセージハンドラにより適切な処理ロジックが呼び出され，自身が持つデータを参照し更新処理する．各エージェントは唯一の識別子 (キー) を持つ．アプリケーションは，識別子によってエージェントの生成や削除，メッセージ送信を行う．エージェントが持つメッセージハンドラは，メッセージと 1 対 1 対応となっており複数のメッセージが同時に処理されることはない．

2.1.2 データモデル

エージェントが持つデータは，レコードとして管理される．レコードは，リレーショナルデータベースと同様に主キーとデータからなり，列を指定することで，データを取り出すことができる．エージェントは，少なくとも 1 つのマスターレコードと呼ばれるレコードを持ち，列の値にレコードセットと呼ばれる複数のレコードを設定することが可能である．エージェントはレコードセットに対してレコードリストの取得やレコードの追加・削除を行うことができる．エージェントのデータレコードはマスターレコードをルートとしたツリーとなる．

3. RDA MAS

RDA MAS とは，2 章に述べた MAS により RDA を実現したものである．エージェントの非同期性を利用した高速処理技術とインメモリ技術により RDA を可能とする．

3.1 RDA アプリケーション

ここでは，RDA MAS を適用する RDA アプリケーションであるジョギング距離のランキングシステムについて説明する．このアプリケーションは，スマートフォンの位置情報を利用して，ユーザーのジョギング距離をランキングする．ランキングは，ユーザーの年齢や性別，住所等のユーザー属性に基づき複数のランキングが用意されており，集計結果はユーザーに合わせて提示する．また，ユーザーの問い合わせに対しては，常に最新の結果を提示する．

RDA のランキングアプリケーションへの適用は，次の点から難しい問題である．この問題は，ランキングアルゴリズムの特徴に起因する．

- データの順序が集計結果となる．

この特徴から，データ件数が処理速度に依存，1 つのデータ更新が他のデータに影響，問い合わせ処理の複雑化といった問題が発生する．

3.2 RDA MAS の設計

RDA MAS の設計は，システムの機能を設計するエージェント設計とシステムのパフォーマンスを設計する (要件を満たす) エージェント分散配置の 2 段階の設計となる．

3.2.1 エージェント設計

エージェント設計は，図 2 の 4 ステップの設計からなり，エージェントの雛形となるエージェントタイプと負荷分析を目的としたプロトタイプを導出することを目的とする [村田 12].

1. 要件設計
   クライアントから要求されたシステム要件をシナリオとして書きだす．シナリオには，具体的なデータ，処理が含まれている必要がある．
2. システム設計
   シナリオから抽出された処理，データからプロセスフローとデータレコードを設計する．その後，シナリオから設計した成果物を関連付ける．
3. エージェントタイプ設計
   関連付けた成果物をまとめエージェントタイプとする．処理やデータの特徴から名前付けを行う．
4. プロトタイプ設計
   エージェントタイプに具体的なデータを挿入し，システムのプロトタイプを設計する．エージェントの通信数，データレコード数から負荷が集中するところを発見する．

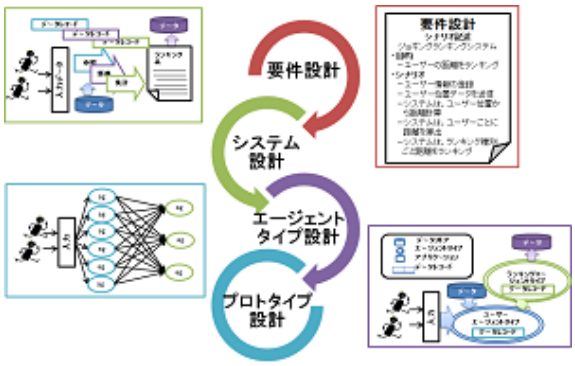

図 2: RDA MAS の 4 ステップ設計フロー

3.2.2 エージェント分散配置

エージェント分散配置では，負荷が集中するエージェントの負荷分散や分散環境上でのエージェントの配置方法を設計する．エージェント分割とエージェント配置によりシステム全体のパフォーマンスを向上させる．

- エージェント分割
  RDA MAS では，データと通信が集中する集約エージェントが存在する．そのため，集約エージェントは，システム上のボトルネックとなる可能性がある．エージェント分割では，集約エージェントのデータを分割し集中する通信を分散する (メッセージ分散)．1) レンジ分割，2) リスト分割，3) ハッシュ分割の 3 つの分割が存在し，アプリケーション特性に合わせて適用する (表 1).

- エージェント配置
  分散環境上へエージェントを配置する際，エージェントのリソース使用量やエージェント間で発生する通信を考慮して配置しなければならない．エージェント配置では，3 つの配置方法によりパフォーマンスのチューニングを行う．1) 通信コストを削減しコロケーションを行うエージェントクローニング，2) リソースをタイプ別に割り当てるエージェントタイプ，3) 通信頻度でエージェントをまとめるエージェントグループ．これらの配置法は，アプリケーションの特性に合わせて選択することで高い効果を得ることができる (表 2).

4. 性能評価

RDA MAS では，RDA を実現するためにエージェント分散配置がアプリケーションに依存した設計となってしまう．その

表 1: エージェント分割ごとの特徴及び適用時の注意

パーティショニング	特徴	適用時の注意
レンジ分割	分割条件が検索条件と一致しているとき検索性能が上がる.	データに偏りが存在する場合，負荷分散の効果が薄い.
リスト分割	レンジ分割よりカーディナリティ(選択性) の低いものに対して適用される.	リスト化できないものに対しては適用が不可能.
ハッシュ分割	データの固有識別子を用いることで均等な分割が可能.	条件検索を行う場合，ハッシュ計算に工夫が必要.

表 2: エージェント配置ごとの特徴及び適用時の注意

配置パターン	特徴 (上：データ集計　下：問い合わせ)	適用時の注意 (上：データ集計　下：問い合わせ)
エージェントクローニング	通信相手が，同一環境上に存在するため通信コストが発生しない.	データ集計時，複製されたエージェントと同期をとる必要がある.
	クローニングによりタスクとデータが分散されるため，問い合わせ負荷が小さい.	エージェントが分散環境上に複製されるため，すべての環境に問い合わせを行う必要がある.
エージェントグループ	通信頻度でグループ化しているため通信コストが小さい.	配置するにあたり通信頻度を測定する必要がある.
	通信頻度から問い合わせ頻度を予測することができる.	通信頻度の高いエージェントグループが存在する環境は，負荷が集中する.
エージェントタイプ	エージェントタイプごとに，独立にリソースを割り当てることができるため，異なるエージェント間でリソース競合が発生しない.	異なるエージェントタイプ間で通信が行われる時，通信コストが最大となる.
	問い合わせ処理をエージェントタイプで分散することが可能.	集約エージェントが，同じ環境に存在するため負荷が集中する.

ため，表 1，2 に基づきアプリケーションの特性に合わせた配置，分割を適用する必要がある．以下の実験では，RDA MAS を適用したジョギング距離のランキングシステムに分割，配置を行った場合の性能評価である．実験環境は，表 3 にあるサーバーを 3 台，100Mbps の LAN により接続した．1 台をテストデータを発生させるマシン，2 台をエージェントサーバーに設定した．

表 3: 実験環境

CPU	AMD-FX8100 8Core 2.8GHz
Memory	12GB
OS	CentOS 5.8 64bit
Java	JRE 1.7.0 IBM Linux 64bit
JVM HeapSize	Xmx = 2048M

4.1　実験 1 エージェント分割

実験 1 では，データの順序関係を崩さないレンジ分割と均等分割を可能とするハッシュ分割を分割数を変えて，問い合わせ性能を比較した．

結果，図 3 となった．図から，エージェント分割数の増加に伴い，レンジ分割の処理性能が向上した．これは，分割によりエージェントが保持するデータ数が削減されたこととデータ順序が保存されることで問い合わせするエージェント数が常に一定となり，全体でのデータ参照回数が減少したことが性能向上につながった．一方，ハッシュ分割は，1 エージェントが保持するデータ数は削減されるもの問い合わせ回数が増加するため，全体でのデータ参照回数は分割数によらずほぼ一定である．そのため，分割数の増加が性能向上に影響しなかった．

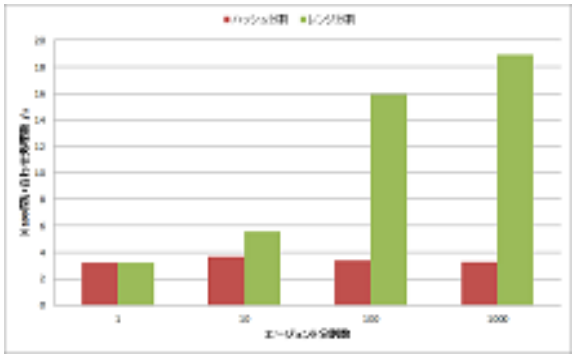

図 3: 実験 1 異なるエージェン分割を適用した際の問い合わせ性能の比較

4.2　実験 2 エージェント配置

実験 2 では，エージェントクローニングとエージェントタイプの 2 つの配置方法をエージェント数を変えて配置し，データ集計処理の処理性能を比較した．

結果，図 4 となった．図から，エージェントタイプによる配置が処理性能が高い結果となった．エージェントタイプによる配置では，1 エージェントあたり 0.1ms の通信コストが発生

している．しかし，処理速度の速いユーザーエージェントとリソースの占有期間の長いランクエージェントが独立の環境に存在するため，異エージェントタイプ間でのリソース競合は発生しない．よって，エージェントタイプのほうが処理性能が高い結果となった．

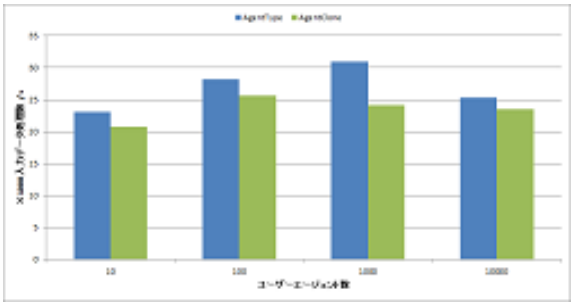

図 4: 実験 2 異なるエージェント配置を適用した際のデータ処理性能の比較

5. 考察

エージェント分割では，レンジ分割を適用することで分割数に対してスケールに性能を上げることができた．しかしながら，本実験で用いたテストデータはユーザー ID を基準とした一様乱数により生成されている．そのため，レンジ分割を適用後，通信やデータの偏りは発生しない．データの偏りを考慮した場合，レンジ分割を適用する際，エージェントごとにレンジ幅を変更するか，エージェントの分割数を増やしレンジ範囲を狭める必要がある．前者では，データ順序を崩さないためにレンジ幅の変更後，隣接するレンジを持つエージェント間で同期を取らなければならない．後者では，通信が発生しないエージェントが生成される可能性がある．一方，ハッシュ分割では均等分割を行うことができるため，分割時にはエージェントが保持するデータのみを考慮すれば良い．しかしながら，分割数の増加による処理性能の向上が難しい．処理性能の向上を図るには，ハッシュ計算時にデータの問い合わせやデータ構造を考える必要がある．

エージェント配置では，エージェントタイプによる配置が処理性能の高い結果であった．しかしながら，この実験結果はエージェントサーバーが 2 台の環境下での実験であることの影響が大きい．エージェントタイプによる配置では，エージェント 1 体あたりの通信コストが 0.1ms とエージェントの処理コスト 0.5 μ s と比較して非常に大きい．もし，ユーザー数の増加に対してサーバーを増加させた場合，ユーザー数に比例して通信コストは増大する．一方，エージェントクローニングでは，通信コストが発生しないのでユーザーの増加における影響は小さい．また，再配置を考慮した場合エージェントタイプでは，エージェントタイプごとの影響を考慮し配置数やエージェントタイプごとのサーバー割当数を考慮しなければならない．しかし，エージェントクローニングでは，ユーザーエージェントの分散のみを考慮して配置を行えばよい．

6. 結論

本稿では，スマートフォンサービスによくある位置情報を利用したアプリケーション基盤に RDA MAS を適用した．このアプリケーションでは，ユーザーの走行距離の集計とランキング集計の 2 段階の集計を行うシステムとなっている．そのため，次の 2 点の処理性能が課題となる．1) 内部のデータ集計処理の性能，2) 外部からの問い合わせ処理の性能の 2 点である．本稿では，処理性能の課題をエージェント分散配置手法であるエージェント分割，エージェント配置の 2 つのパフォーマンスチューニング手法により解決を図った．

エージェント分割では，データベースで利用されるパーティショニングと呼ばれる手法を実装したシステムに適用し，負荷が集中する集約エージェントの分割数を変えて実験を行った．実験の結果，データの順序関係を崩さずに分割することができるレンジ分割のほうが処理性能が高い結果となった．

エージェント配置では，3 つの配置パターンを適用し，ユーザーエージェント数を変えて実験を行った結果，リソースの割り当てをタイプによって分けられるエージェントタイプ配置のほうが性能の高い結果となった．

それぞれの実験結果から，エージェント分割，エージェント配置をアプリケーションの特性に合わせて選択することでパフォーマンスを向上させることが可能であることを確認した．また，今回のアプリケーション開発では，GPS データの更新期間である 1 秒以内の集計を目標として開発を行った．結果，実験 2 結果から RDA MAS にアプリケーション特性に合わせたエージェント分散配置を適用することで，全ユーザーのリアルタイム処理を可能とするシステムを実現した．

今後の展望として，データの偏り，通信の偏りに対処可能なエージェント分散配置の提案を挙げる．考察にもあるように，本稿で行った実験はテストデータの増分がほぼ一定である．そのため，特定のエージェントに通信が集中するといったことはない．実際には，データには偏りが存在し，その偏りによりレンジ分割の適用は難しくなることが予想される．

参考文献

[Amini 06] Amini, L., Jain, N., Sehgal, A., Silber, J., and Verscheure, O.: Adaptive control of extreme-scale stream processing systems, *In ICDCS 2006*, pp. 71–79 (2006)

[Arasu 06] Arasu, A., Babu, S., and Widom, J.: The CQL continuous query language: semantic foundations and query execution, *The VLDB Journal*, Vol. 15, pp. 121–142 (2006)

[Dean 08] Dean, J. and Ghemawat, S.: MapReduce : Simplified Data Processing on Large Clusters, *Communications of the ACM*, Vol. 51, No. 1, pp. 1–13 (2008)

[山本 08] 山本 学 : エージェントプログラミングモデルの高速トランザクション処理システムに対する効果の考察, 合同エージェントワークショップ＆シンポジウム JAWS(Joint Agent Workshop and Symposium)2008 (2008)

[山本 11] 山本 学 : 高速マルチエージェントシステムによるリアルタイムデータ集計処理の考察, 合同エージェントワークショップ＆シンポジウム JAWS(Joint Agent Workshop and Symposium)2011 (2011)

[村田 12] 村田 悠也 : リアルタイムデータ集計を行う高速マルチエージェントシステムにおけるエージェント分散配置の性能評価, 合同エージェントワークショップ＆シンポジウム JAWS(Joint Agent Workshop and Symposium)2012 (2012)