大規模ソーシャル・ネットワークシミュレーション基盤の設計と実装

岡田 瑞穂† 鈴村 豊太郎‡

† 東京工業大学大学院 情報理工学研究科, JST CREST 〒152-8550 東京都目黒区大岡山 2-12-1
‡ IBM Research, University College Dublin, JST CREST
E-mail: † okada.m.af@m.titech.ac.jp, ‡ toyo@jp.ibm.com

あらまし 近年，Twitter などの SNS における情報拡散に関する研究が多く行われている．既存の研究では，過去の現象を分析してモデル化する一方で，それらのモデルを用いた情報拡散のシミュレーション実験に関しては，小規模な実験しか行われていない．そこで本研究では，Twitter から収集したデータを用いて数億ユーザ，数百億エッジから成る実ネットワークを再現し，ネットワーク上を流れる情報の広がり方を再現するための大規模情報拡散シミュレーション基盤をスーパーコンピュータ TSUBAME 上に構築した．さらに，リツイートによって情報が拡散される際に，同じ情報が流れることを利用してメッセージの圧縮を行い，通信量の削減を行った．

キーワード Twitter，シミュレーション，大規模グラフ，ソーシャル・ネットワーク

1. はじめに

近年，Twitter や Facebook などのソーシャル・ネットワークの普及により，ユーザ同士の情報の共有が盛んに行われている．Twitter では，他のユーザをフォローすることにより，そのユーザが投稿したツイートを自分のタイムラインに受け取ることができる．また，他のユーザのツイートをリツイートすることにより，自分をフォローしているユーザ（フォロワー）のタイムラインにそのツイートが表示される．リツイートされたツイートがさらにリツイートされることによって，ツイートの情報は巨大なネットワーク上に爆発的に拡散される．2013 年には，Twitter の月間アクティブユーザ数は 2 億人にものぼり，全世界のユーザが 1 秒あたりに投稿するツイートの数は 5000 を超えることが発表されており[1]，ネットワークを流れる情報の量はますます増えている．

Twitter を構成する巨大なネットワークにおいて，どのような情報がどのような経路で拡散されていくのかという問題は，多くの研究者が取り組んでいる分野である．この分野では，過去の情報拡散に関してデータを収集・分析し，ユーザの行動や情報の広がり方をモデル化する．しかし，実際に Twitter のサービスを利用してモデルの検証を行うことは現実的に不可能である．そこで，Twitter のネットワーク構造を再現したグラフを計算機上に構築し，擬似的にユーザの振る舞いを制御するシミュレーション実験を行うことが有効になってくる．しかし，Twitter から全てのユーザのフォロワー情報やプロファイル情報を取得するのは困難であり，部分グラフを用いた小規模なシミュレーション実験しか行われていない．また，仮に完全な Twitter グラフを収集できたとしても．既存の分散グラフ処理系による計算モデルは情報拡散シミュレーションに対して最適化されておらず，時間経過に従って拡散されていく大量のツイートが計算機間の通信におけるボトルネックになる．

そこで本研究では，Twitter の実データネットワークを搭載した大規模ソーシャル・ネットワークシミュレーション基盤の構築と情報拡散シミュレーションのためのメッセージ通信の最適化手法の 2 点を提案する．

シミュレーション基盤の構築では，Twitter から 3 ヶ月かけて収集したデータ[2]を用いて数億ユーザ，数百億エッジから成る実データネットワークを再現し，エージェントベースシミュレーション基盤 XAXIS[3]上に実装した．ユーザにはタイムラインとフォロワーリストのデータに加えてツイート，リツイートの機能を持たせた．加えて，プロファイルデータを用いて実際のツイート頻度をデータとして持たせ，シミュレーションシナリオに合わせた頻度でツイートさせることを可能とした．

さらに，ツイートがリツイートによって拡散される際に計算機間の膨大なメッセージ通信が性能のボトルネックとなる問題について，同じツイートに対する複数ユーザのリツイートをひとつのメッセージにまとめてから通信することにより，通信量を削減する手法を提案した．

評価実験はスーパーコンピュータ TSUBAME 2.5 を用いて行い，クラスタ数とシミュレーションステップの設定が特定の条件下では現実世界の時間よりも早い実行時間でシミュレーションを行えることを示した．また，メッセージ通信の提案手法では，シミュレーションステップの経過に伴ってツイートが拡散していくほど通信量削減の効果が大きくなることを確認した．

以降の章では，第 2 章で情報拡散とグラフ処理系に関する関連研究について述べる．第 3 章では本研究で使用したエージェントベースシミュレーション基盤 XAXIS について述べたあと，第 4 章で提案手法について述べる．第 5 章で評価実験を行い，第 6 章でまとめと今後の課題について述べる．

2. 関連研究

2.1. 情報拡散に関する研究

岡田ら[4]は，東日本大震災時に広まった数例のデマに対し，拡散過程の解析，デマ拡散のモデル化を行い，そのモデルを検証した．[4]では伝染病拡散モデルのひとつである SIR モデルを改良し，デマに感染した状態や訂正情報を受け取った状態などの感染状態をユーザに持たせ，数理モデルに従ってユーザの状態変化を観察するシミュレーションを行った．[4]の実験で使用されたデータは 5 万ユーザの小規模なネットワークであり，全世界規模の実験は行われていない．もし全世界規模のネットワークを効率的に処理できる処理基盤があれば，ネットワーク上でデマが発生する場所などの条件を変えてシミュレーションし，モデルの詳細な検証や新しいモデルの考察が可能になる．

また，アメリカ大統領選挙時のツイート[10]や，インターネット・ミーム（meme）の広がり[11]，地震時のツイートの拡散の様子[12]など，Twitter で実際に起きた情報拡散に関するツイートデータを集め，時系列に沿った拡散量を中心に分析した研究があるが，シミュレーション実験に関しては行われていない．収集したツイートデータを実際のネットワーク上に配置して大規模なシミュレーション実験を行えるようになると，それらのツイートがネットワーク上をどのように流れたかを知ることができ，対象のイベントのさらなる理解が進むようになる．

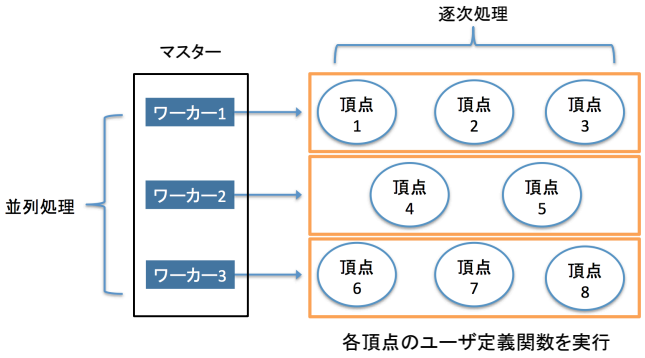

図 1 バルク同期並列の概要

2.2. グラフ処理系

2.2.1. Pregel とバルク同期並列

情報拡散モデルなどのアプリケーションに関する研究も多い一方で，大規模ネットワークを処理するための大規模グラフ処理系の手法も多く提案されている．Pregel[7]は汎用的な分散グラフ処理モデルであり，これに従った多くの実装が存在する[8][9]．Pregel はバルク同期並列（bulk-synchronous parallel）という計算モデルに則っている．バルク同期並列では，図 1 のようにただひとつ存在するマスターと呼ばれるオブジェクトが複数のワーカーを並列に実行し，ワーカーはグラフの各頂点のユーザ定義関数を実行することで分散処理を実現している．ユーザ定義関数内では他の頂点にメッセージを送ることができる．頂点の処理とメッセージ交換はスーパーステップと呼ばれる繰り返し単位ごとに行うことができ，頂点の内部状態を更新していくことで計算を行う．この計算モデルにおいて最も時間を要するのは，頂点間のメッセージ交換である．一般に，ワーカーは計算機ごとに配置されるので，ワーカー間をまたぐメッセージの数はほぼそのまま計算機間の通信量となる．我々が実験で用いた XAXIS もバルク同期並列に則る．

2.2.2. Pregel における通信量削減

グラフアルゴリズムの中には，PageRank や BFS など，すべての隣接頂点に同じメッセージを送る手法を使うものが多くある．PageRank では現在の頂点の値を，BFS では頂点の ID をメッセージとして送信する．また，我々が行う情報拡散シミュレーションでは，ユーザが投稿したツイートがフォロワーエッジに沿って隣接頂点に拡散していく．このような場合，従来の Pregel 処理系では，メッセージを行き先頂点ごとにコピーして送信していたが，同じメッセージが同じワーカーに複数届く場合，重複しているメッセージは減らせるはずである．

この問題に対処するため，Pregel のオープンソース実装である GPS[8]では，閾値 n 以上のエッジ数を持つ頂点に関して，隣接頂点リストを頂点ごとではなく送り先のワーカーにもたせ，メッセージを全ワーカーに通信したあとで，各ワーカーで送られてきたメッセージの配送先となる頂点を決定してからコピーすることで，メッセージ量を削減する手法を提案した．この隣接リストの分割により，頂点とワーカー間はメッセージをひとつだけ送信すればよい．しかし，[8]では閾値 n の最適値が不明で，グラフごとに異なるためユーザの入力によるものであり汎用性に欠ける．また，情報拡散シミュレーションでは隣接リストは各頂点が持ち，ユーザ定義関数内で参照できたほうが良い．

3. エージェントベースシミュレーション基盤 XAXIS

我々は，大規模ソーシャル・ネットワークシミュレ

ーション基盤の実装について，エージェントベースシミュレーション基盤 XAXIS（X10-based Agents eXecutive Infrastructure for Simulation）[3]を使用した．

3.1. エージェントベースシミュレーション

エージェントベースシミュレーションとは，多数の個体（エージェント）が自立的に振る舞った際に，それらの相互作用をシミュレーションするシステムである．本稿においては，エージェントは Twitter のユーザに相当する．

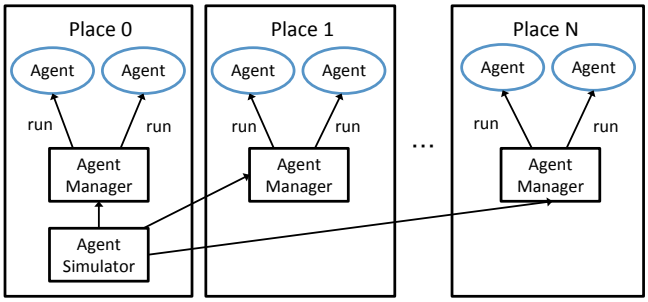

図 2 XAXIX の概要

3.2. XAXIS

XAXIS は，並列分散プログラミング言語 X10[6]で書かれている．X10 は Partitioned Global Address Space（PGAS）モデルを採用したプログラミング言語で，単一アドレス空間を採用していると同時に，計算機を”プレース”と呼ばれる区間に分割する．プログラムで使われるデータは 1 つのプレース内からしかアクセスできないものの他に，全てのプレースから参照できるデータを扱うこともできる．ひとつの計算機に複数のプレースを作成することも可能である．一般に，計算機の数を増やすことで大規模なデータの処理にも対応できる．

XAXIX の概要を図 2 に示した．XAXIS では Agent を各プレースに割り当てて処理を分担し，ステップ単位で計算を行う．各 Agent は AgentManager がスレッド並列で実行し，AgentManager は AgentSimulator がノード並列で実行する．ここで，AgentSimulator と AgentManager は，バルク同期並列におけるマスターとワーカーの役割を果たす．XAXIS には Java のインターフェースが用意されており，開発者は Java で処理を記述することでシミュレーションを実行することができる．各ステップの流れは以下の通りであり，開発者は run メソッドと receiveMessage メソッドによって Agent の処理を記述する．run メソッドの中では，sendMessage という内部メソッドを呼び出すことで指定した Agent にメッセージを送ることができる．

1. 各エージェントの動作: run()
2. メッセージ交換
3. メッセージ受信時の処理: receiveMessage()

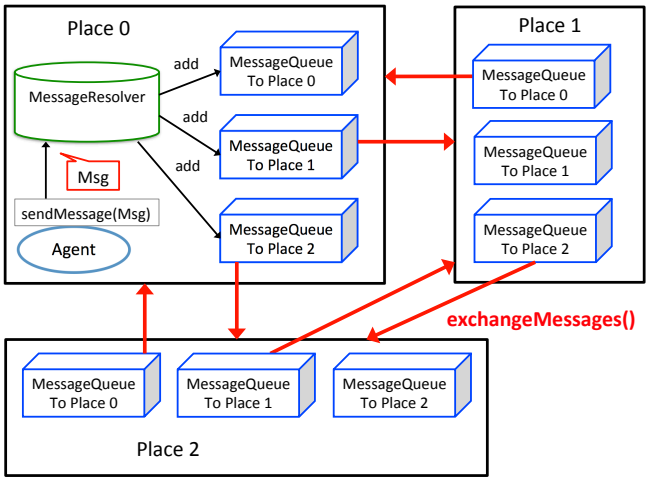

図 3 メッセージ交換の概要

3.3. XAXIS におけるメッセージ交換

XAXIS におけるメッセージ交換の概要を図 3 に示した．Agent が sendMessage メソッドを呼び出すと，メッセージのオブジェクトが作成されて MessageResolver に渡される．MessageResolver はメッセージの宛先 Agent が配置されているプレースを解決し，MessageQueue に貯めていく．MessageQueue はそれぞれのプレースに向けたメッセージを蓄えるキューである．各ステップに一度，AgentSimulator による exchangeMessage メソッドによって，全プレース間で MessageQueue を交換しあう．AgentManager が MessageQueue を受け取り，各 Agent の receiveMessage メソッドによってメッセージ受信時の処理を行う．

我々が行う情報拡散シミュレーションでは，ユーザがエージェント，ツイートがメッセージに対応し，ユーザが投稿したツイートはフォロワーエッジに沿って隣接ユーザに送信される．また，リツイートが繰り返されることによって，ツイートはグラフ上に爆発的に拡散されていく．

Pregel や XAXIS では，ある頂点から同一内容のメッセージが送信されることを想定していないため，図 4 のように一つのメッセージに一つの送信者 ID と一つの受信者 ID を持たせて送信するユニキャストというメッセージ送信手法が採用されている．図 4 では，最初のステップにおいてユーザ 0 がツイートを発し，ユーザ 1 とユーザ 2 が受け取ったツイートを次のステップでリツイートして，ユーザ 3～6 がリツイートを受け取る．この場合，ワーカー0 と 1 の間，ワーカー1 と 2 の間のメッセージ量はそれぞれ 2，4 となり，ステップの経過に伴いメッセージ量が増えることがわかる．ユニキャストは汎用的な手法であるが，同じ内容のメッセージが同じワーカーに複数届く場合，重複したメッセージは減らせるはずである．

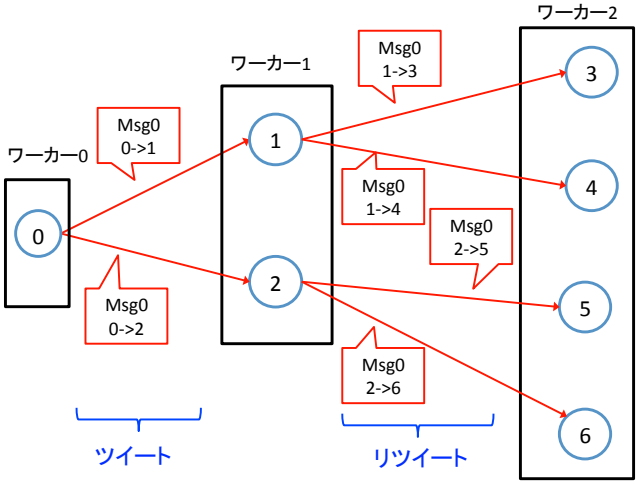

図 4 ユニキャスト

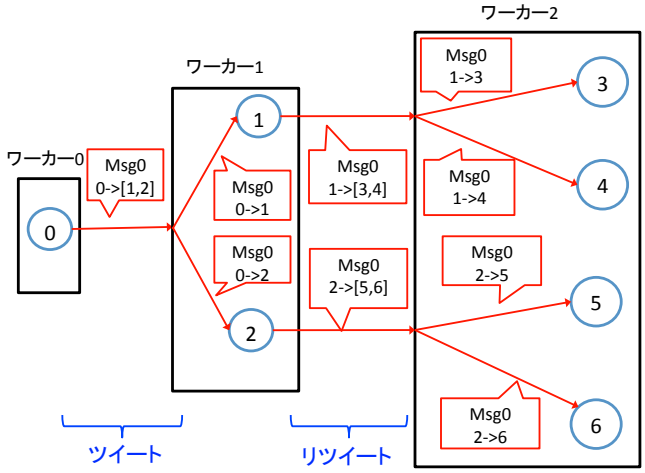

図 5 マルチキャスト

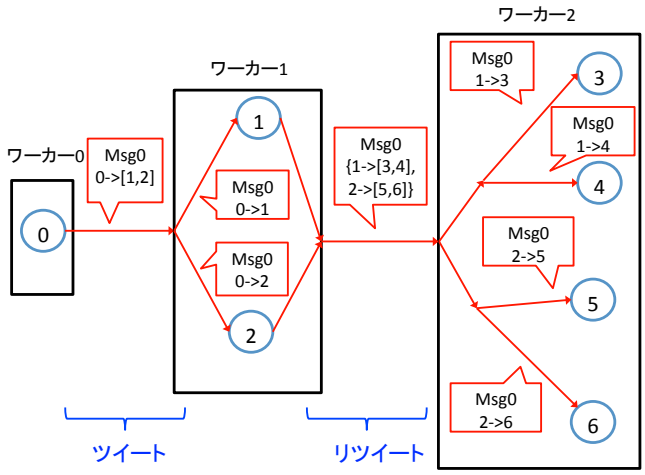

図 6 マルチリツイート

4. 提案手法

我々は，情報拡散シミュレーションにおいて，同じ内容のメッセージがワーカー間の通信に複数発生する場合のメッセージ削減手法を提案する．また，提案手法を用いて，Twitter における全世界規模の情報拡散シミュレーションを行うための実装を XAXIS に施した．

4.1. マルチキャスト

我々はまず，図 5 のように同一内容のメッセージに複数の受信者 ID を持たせて通信するマルチキャストを提案する．これにより，図 4 のユニキャストにおいてユーザ 0 からワーカー1 に向けて 2 つ送信されていたメッセージは，1 つに削減される．同様に，ユーザ 1 とユーザ 2 は，ワーカー2 に向けてそれぞれひとつずつのメッセージを送れば良い．しかし，ユーザ 1 とユーザ 2 が送信するメッセージは，共にユーザ 0 が送信したメッセージと内容が同じであるため，これらをさらに圧縮することができる．

4.2. マルチリツイート

そこで我々は，図 6 のように，メッセージに複数の送信者 ID と複数の受信者 ID を持たせて送信するマルチリツイートを提案する．マルチリツイートでは，同一プレース内で同じ ID を持つマルチキャストのメッセージが複数送信された場合に，それらを圧縮してから送信することにより，通信量をさらに削減する．図 5 と図 6 を比較すると，ワーカー1 からワーカー2 へのメッセージ量が 1 つ減ったことがわかる．

以上の通信手法のまとめを表 1 に示す．

表 1 メッセージ通信手法のまとめ

	送信者	受信者
ユニキャスト	1 人	1 人
マルチキャスト	1 人	複数
マルチリツイート	複数	複数

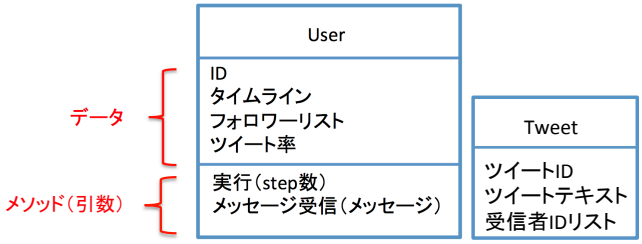

図 7 各クラスの概要

4.3. 実装

我々は XAXIS 上に，図 7 のようにエージェントとして User クラスを，メッセージとして Tweet クラスを実装した．User にはユーザ ID，タイムライン，フォロワーリスト，ツイート率のデータを持たせた．ここでタイムラインとは受信したツイートを格納するリストであり，フォロワーリストとはフォロワーの ID のリストである．また，ツイート率とは，そのユーザが各ステップでツイートする確率である．さらに，User のメソッドとして各ステップで必ず実行される処理と，メッセージを受信したときの処理を定義できるようにした．Tweet クラスでは，データとしてツイート ID，ツイートテキスト，受信者 ID のリストを実装した．

表 2　1 ステップあたりの性能

ステップ幅	1 秒	1 分	10 分	30 分	1 時間
実行時間(秒)	50.039	109.532	226	428.118	707.186
ツイート数	3,740	223,791	2,199,180	6,131,861	10,815,024
メッセージ数	323,087	19,323,827	189,728,271	520,258,547	888,385,682
通信時間(秒)	1.496	14.263	140.349	306.768	458.41

5. 評価実験

我々は，Twitter に存在する全世界のユーザデータを用いて 3 種類のシミュレーション実験を行い，システムの性能を明らかにする．

表 3 使用したデータの規模

収集期間	2012 年 7 月~2012 年 10 月
ユーザ数	435,846,642
フォロワーエッジ数	26,839,944,508

5.1. 使用したデータ

我々は，先行研究[2]により世界規模に収集したユーザデータを用いて，実際の Twitter ネットワークを XAXIS 上に再現した．データは Twitter Rest API を用いて 2012 年 7 月から 2012 年 10 月まで 3 ヶ月間クロールし，フォロワー・フレンド情報，ユーザプロファイル情報を収集した．[2]では 4.69 億ユーザが集まったが，本研究ではユーザ情報を非公開としているユーザはデータ不足のため除き，最終的に表 3 に示した規模のデータを用いた．収集したデータの詳細な分析については[2]が詳しい．

5.2. 実験環境

実験は，東京工業大学のスーパーコンピュータ TSUBAME 2.5 の計算機を複数台用いた．環境は全ノード共通で，CPU Intel Xeon（2.93GH 6 コア 2 ソケット），54GB Memory，QDR InfiniBand x2 (80Gbps) ネットワーク，ソフトウェア構成は IBM Java1.6.0 と X10 2.2.2.1 を用いる．

5.3. 基本的なシミュレーションシナリオ

この実験では，全てのユーザに現実世界と同じ頻度でツイートさせるシミュレーション実験を行い，その際のシステムの性能を調べる．このシナリオではリツイートを用いないためマルチリツイートを適用できないので，メッセージは全てマルチキャストである．この実験ではステップ幅と呼ぶパラメータを変えて実験を行う．ステップ幅は 1 ステップにおける現実世界の時間に対応し，ユーザはステップ幅に対応したツイート頻度にもとづいてツイートを流す．例えば，ステップ幅を大きくするほどネットワークを流れるツイートの数が多くなる．ステップ幅の設定としては，1 秒，1 分，10 分，30 分，1 時間の 5 つのパターンを用意した．なお，ここでは X10 のプレース数は 128 とし，1 プレースにつき約 330 万ユーザを割り当てた．

5.3.1. 実験結果

実験結果を表 2 に示した．1 ステップあたりの実行時間は，1 秒，1 分のシミュレーションでは現実世界よりも遅い性能となったが，ステップ幅が 10 分以上では現実世界よりも早い処理を行えている．

また，ステップ幅を大きくするとツイート数とメッセージ数が増え，それに伴い通信時間も増えていることがわかる．このメッセージ数はマルチキャストであるが，ツイート数とメッセージ数を見てみると，ツイート数よりもかなり多い数のメッセージが流れている．これは Twitter の平均フォロワー数と比較すると非常に大きい数であるが，ステップ幅を 1 秒としたときにツイートしたユーザの平均フォロワー数は 850 人にも登り，頻繁にツイートを投稿するユーザは多くのフォロワーを持っていることがわかる．

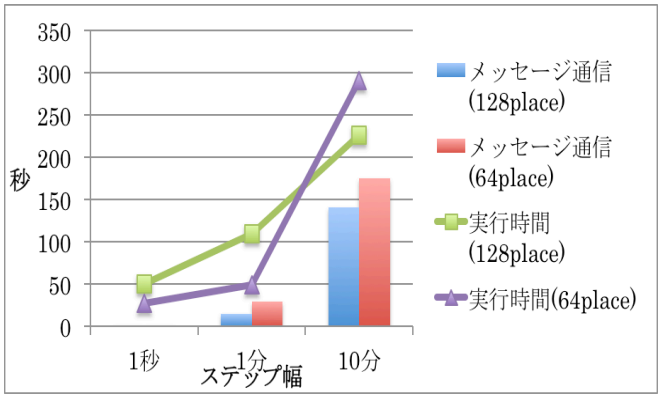

図 8　プレース数ごとの性能比較

5.4. プレース数を変えた実験

この実験では，前の実験で 128 としたプレース数を 64 にして性能を比較する．一般にプレース数を減らすと，エージェント実行の並列数とワーカー間のメッセージ数が減るというデメリットがあるが，通信経路が減ることによって同期待ちのオーバーヘッドが減るというメリットがある．

5.4.1. 実験結果

実験結果を図 8 に示した．1 ステップあたりの実行時間は，1 秒や 1 分での短いステップ幅では 64 プレースの方が早く処理を終えている．しかし，メッセージの通信時間では 128 プレースの方が常に早い結果となっており，並列通信の効果が出ている．この結果から，メッセージ数が少ない場合はプレース数を増やしたことによる並列通信の効果が薄く，通信時間に差が出ないことがわかる．逆にメッセージ数が多い場合，通信

量が大きなボトルネックとなるので，プレース数を増やして通信の並列数を増やすことがパフォーマンスの改善に繋がる．

5.4.2. マルチリツイートの効果測定

この実験では，4.2 節で述べたメッセージ通信量削減手法であるマルチリツイートについて，マルチキャストとの比較を行う．マルチリツイートはリツイートによる情報拡散シナリオを設けた際に効果を発揮する手法であるので，今回は各ステップにおいて以下のシナリオを用意した．

ステップ 0. フォロワー数が 1000 人のユーザ 27 人がツイートする

ステップ 1. 前のステップで受け取ったツイートを 1%の確率でリツイートする

ステップ 2. 前のステップで受け取ったツイートを 0.5%の確率でリツイートする

ステップ 3. 前のステップで受け取ったツイートを 0.25%の確率でリツイートする

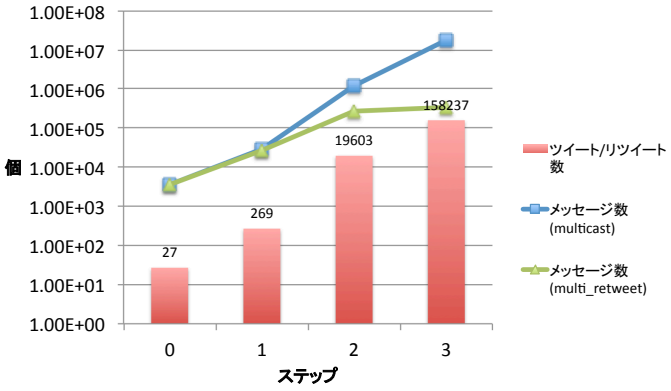

図 9 メッセージ数の比較

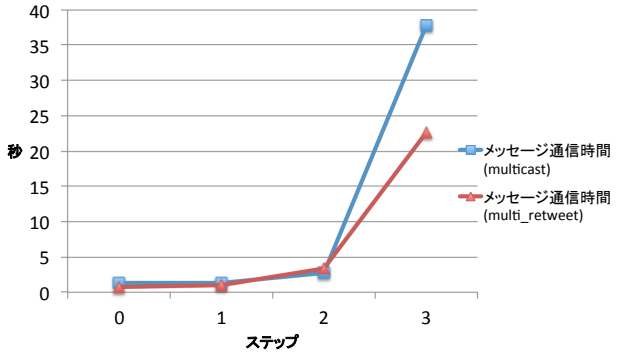

図 10 通信時間の比較

5.4.3. 実験結果

図 9 にステップごとのツイート数，メッセージ数の対数グラフを載せる．ステップが増すごとにリツイートするユーザ数が増えていき，それに伴ってマルチキャストのメッセージ数が増えるが，マルチリツイートではメッセージの増え方が緩やかになり，メッセージ削減の効果が大きくなることが確認できる．しかし，図 10 の通信時間の比較を見ると，ステップ 2 までではメッセージ数が少ないためか，通信時間に差がなく，ステップ 3 で 15 秒程度の通信時間削減効果が見られる結果となった．

6. まとめと今後の課題

本研究では，Twitter から収集した全ユーザのデータを用いて実際のネットワークを再現し，ソーシャル・ネットワークにおける情報拡散をシミュレーションするための大規模処理基盤をエージェントベースシミュレーション基盤 XAXIS 上に構築した．

実験では，ステップ幅とプレース数が特定の条件下で，現実世界よりも早い時間でシミュレーションが実行できることを示した．また，リツイートによってツイートが拡散される際に，同じツイートに対する複数ユーザのリツイートが計算機間の通信に多く発生することを利用し，これらをひとつのメッセージにまとめてから通信することにより，通信量を削減する手法を提案した．マルチキャストと比較し，ステップ数が増えるにつれて増大するメッセージ数を削減することに成功した．

今後の課題としては，マルチリツイートが既存のグラフ処理系に応用可能か検討し，性能を比較する．また，今回実装したシミュレーション基盤を用いて，実際に情報拡散モデルの精度検証を行いたい．さらに，様々なモデルやシナリオを設定したときのパフォーマンスやボトルネックを測り，基盤としてのさらなる汎用性を求める必要がある．

謝辞

本研究の一部は JST CREST「ポストペタスケールシステムにおける超大規模グラフ最適化基盤」の援助による．

また，本研究を進めるにあたり、ご指導を頂いた東京工業大学徳田雄洋教授と実験用のデータを収集・提供して下さった東京工業大学上野晃司氏に感謝いたします。

参 考 文 献

[1] http://www.sec.gov/Archives/edgar/data/1418091/000119312513390321/d564001ds1.htm

[2] Masaru Watanabe and Toyotaro Suzumura, “How Social Network is Evolving ? A Preliminary Study on Billion-scale Twitter Network”, The 2nd International Workshop on Large Scale Network Analysis (LSNA 2013)

[3] Toyotaro Suzumura and Hiroki Kanezashi, “Highly Scalable X10-based Agent Simulation Platform and its Application to Large-scale Traffic Simulation”, 2012 IEEE/ACM 16th International Symposium on Distributed Simulation and Real Time

Applications

[4] 岡田佳之，榊剛史，鳥海不二夫，篠田孝祐，風間一洋，野田五十樹，沼尾正行，栗原聡，“拡張 SIR モデルによる Twitter でのデマ拡散過程の解析”，2013 年度人工知能学会全国大会

[5] Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo, ”Earthquake Shakes Twitter Users: Realtime Event Detection by Social Sensors”, Proc. 18th International World Wide Web Conference (WWW2010)

[6] Saraswat, Vijay A., Sarkar, Vivek, and von Praun Christoph , “X10: concurrent programming for modern architectures”, In Proceedings of the 12th ACM SIGPLAN symposium on Principles and practice of parallel programming (PPoPP '07)

[7] G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski, "Pregel: a system for large-scale graph processing”, in Proceedings of the 2010 international conference on Management of data, ser. SIGMOD ’10

[8] Salihoglu, Semih and Widom, Jennifer, “GPS: A Graph Processing System”, Scientific and Statistical Database Management 2013.

[9] Bao Nguyen and Toyotaro Suzumura, "Towards Highly Scalable Pregel-based Graph Processing Platform with X10", The 2nd International Workshop on Large Scale Network Analysis (LSNA 2013) In conjunction with WWW 2013

[10] Lin et al, ”#Bigbirds Never Die: Understanding Social Dynamics of Emergent Hashtags”, ICWSM 2013

[11] Backhage et al, ”Mathematical Models of Fads Explain the Temporal Dynamics of Internet Memes”, ICWSM 2013

[12] Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo, ”Earthquake shakes Twitter users: real-time event detection by social sensors”, In Proceedings of the 19th international conference on World wide web (WWW '10)