修士学位論文

通信遅延を考慮した計算資源共有システムに関する研究

東北大学大学大学院工学研究科
電気・通信工学専攻
博士前期課程 2 年
中村 貴

目 次

図 目 次

表 目 次

第1章

序論

1.1 本研究の背景

近年, マルチコア・マルチスレッドによる並列分散処理によって計算処理ユニットの性能向上が試みられており, マルチコアによる演算処理はパーソナルコンピュータから科学計算用 PC クラスタまで幅広く取り入れられている. 第 1 章では単一処理ユニットによる処理性能の限界から並列分散コンピューティングによる上記限界に対する解決について述べ, その後並列分散コンピューティングの具体的な事例に触れた後, 本研究が目的とする計算資源共有システムについて述べる.

1.1.1 単一処理ユニットの限界

演算処理ユニットの性能の指標として 1 秒間に実行可能な命令数が用いられることが多いが, 単一コアからなる処理ユニットにおいては実行可能命令数はクロック周波数に左右されることが多い. およそ 1970 年代から処理ユニット (CPU) の開発は本格的に開始され, 1971 年に発表された世界初の商用マイクロプロセッサである Intel 4004 は 108kHz のクロック周波数で動作した [1]. その後開発が進み 2004 年 Intel Pentium4 においては 1.30GHz までクロック周波数は増加し, クロック周波数のみで比較すれば約 30 年でおよそ 1 万倍の性能向上が行われたことになる.

しかし 1999 年 Intel 社の MRL(Microprocessor Reserch Labs) に所属していた Fred Pollack により「プロセッサの性能がダイサイズの平方根に比例する」という経験則 (ポラックの法則) [2] が発表され, トランジスタ数を増やし消費電力を増大させても性能の向上が思わしくないことが示唆された. プロセッサ全体の発熱量には限界があるため, CPU コアの性能は熱の問題によって頭打ち

になることが予見されたのである．事実, 1990 年代初頭にはラップトップ PC の発熱や PC の冷却ファンの騒音が問題として認識されはじめた．一方クロック周波数の向上が続いた際に, 1 クロックあたりの時間における電気信号の伝搬可能な距離の短さも問題になってくることが想定され, 前述の発熱の問題とともに単一コアからなる処理ユニットの性能向上は限界に達した．

1.1.2 並列分散処理

単一コアによる処理ユニットの性能向上が限界に達すると, 複数のコアを組合せ並列的に動作させることによる並列処理ユニットの開発が活発化した．複数コアの利用自体はスーパーコンピュータなどでは早期から行われていたものではあるが, 近年においては一般ユーザ向け PC 用 CPU におけるマルチコア化やネットワーク上の処理ユニットを用いて分散処理を行う処理ユニットの構築など, 演算処理ユニットの性能向上のための並列化も多様な方法で行われている．

スーパーコンピュータの構築

一般的な処理ユニットの性能をはるかに凌駕する処理性能を有する計算処理ユニットをスーパーコンピュータと呼ぶが, スーパーコンピュータは膨大な計算処理を目的として構築され, そのため大規模なハードウェア・ソフトウェアを備える．気象予測や分子動力学, 金融工学のように大規模な数値解析に基づくシミュレーションなどに主に利用される．スーパーコンピュータの開発は 1980 年代から行われていた．1980 年代には高性能計算に特化したベクトルプロセッサが利用されていたが, 近年では汎用プロセッサの価格性能比の向上に合わせスーパーコンピュータを構成するプロセッサは汎用プロセッサが主流となっている．

世界で最も高速なコンピュータシステムを上位 500 位まで定期的にランクづけし評価するプロジェクトである TOP500 [3] が 1993 年に発足し, スーパーコンピュータの性能の評価を行っている．TOP500 によると 1993 年 6 月に首位にランクづけされたのは CM-5 [4] と呼ばれる計算機で, 実行性能値は 0.0597TFLOPS という結果が残されている．1993 年に発表された Intel 社の Pentium がおよそ 60MHz のクロック周波数であることから, 仮に 1 クロックで 1 回の浮動小数点計算ができると仮定するなら CM-5 は当時のシングルコア演算処理ユニットのおよそ 1000 倍の処理性能を持っていたことになる．TOP500 の 2010 年 11 月の評価では天河一号が首位にランクづけされており 563TFLOPS もの処理性能を実現している．

スーパーコンピュータによる並列処理環境の構築の特徴は, 一ヶ所に計算資源が集まっていることや莫大な予算をかけて構築されることなどがあげられる．

グリッドコンピューティング

スーパーコンピュータが一ヶ所に計算資源を集めて構築されるものである一方, インターネットなどの広域ネットワーク上から計算資源を結びつけ, ひとつの計算処理ユニットとする手法が研究・開発されている. このような大規模計算環境の構築手法をグリッドコンピューティング [6] と呼ぶ. グリッドコンピューティングが近年注目される背景として, コンピュータネットワークの伝達速度や安定性の向上があげられる. また, ネットワークに接続される計算資源の高性能化や各資源の遊休時間の存在も理由のひとつであると考えられる. グリッドコンピューティングにおけるシステムを構成するネットワーク上の資源としては, 一般家庭に存在する PC から研究拠点に存在するスーパーコンピュータまで様々な想定がされている.

「グリッド」は電力網 (Power Grid) に由来される名称であり, プラグをコンセントに指せばどこでも電力の供給が受けられるように, ネットワークに接続すれば計算環境などによらずどこでもだれでも計算資源の提供を受けることが可能な環境の構築を目指して開発が行われている.

ボランティアコンピューティング

ボランティアコンピューティングはグリッドコンピューティングの一種に分類されるものではあるが, グリッドコンピューティングにおいてより限定されたネットワーク上の計算資源利用法であり, システムにおいて資源の提供者と利用者が厳密に区分されるものである. ボランティアの名称が示すとおり, 基本的に計算資源の保有者はプロジェクトの発案者に無償で計算資源の提供を行う. 科学計算のために研究機関が広く計算資源の提供を求めており, 既に実現されているプロジェクトも少なくはない.

ボランティアコンピューティングにおいて有名なプロジェクトとして SETI@home [5] があげられる. SETI@home は地球外生命体の証拠を検出するといった科学的作業を行うことや, ボランティアコンピューティングの実現性と実用性を証明するという目的をもって始められたプロジェクトである. 前者についてはいまだ成果をあげることができていないものの, 後者については一般的に完全に成功したと見なされている. なぜなら, 2009 年 11 月 4 日に 769TFLOPS という処理性能を実現しており, 2009 年 11 月に TOP500 が首位にランクづけしたスーパーコンピュータの処理性能 (ピーク時 1750TFLOPS) と比較すると計算資源の提供が無償であることを考慮すれば遜色ない結果であると考えられるからである.

1.2 本研究の目的

前述したように, グリッドコンピューティングのひとつの分野であるボランティアコンピューティングはシステムとして既に実現し利用が行われている. しかしグリッドコンピューティングの

設計思想として存在する, だれでも・どこでも計算資源を利用可能とするシステムの構築は未だ研究段階である.

本研究では, ネットワーク上に存在する計算資源をシステムへの参加者なら誰でも利用可能とする管理システムの構築を目指す. 目標とするシステムにおいてはボランティアコンピューティングにおいて資源の利用者と提供者間でのみ通信が行われていたのとは異なり, 参加ノード間での通信が活発に行われることが想定される (図 1.1). したがって, 目標とするシステムにおいて計算資源の共有を有効に行うため, ノード間の通信時間が短くなるようにシステムの利用における通信を最適化する必要がある. そこで本研究では, 特にノード間の通信遅延に着目してシステムの構築を行う.

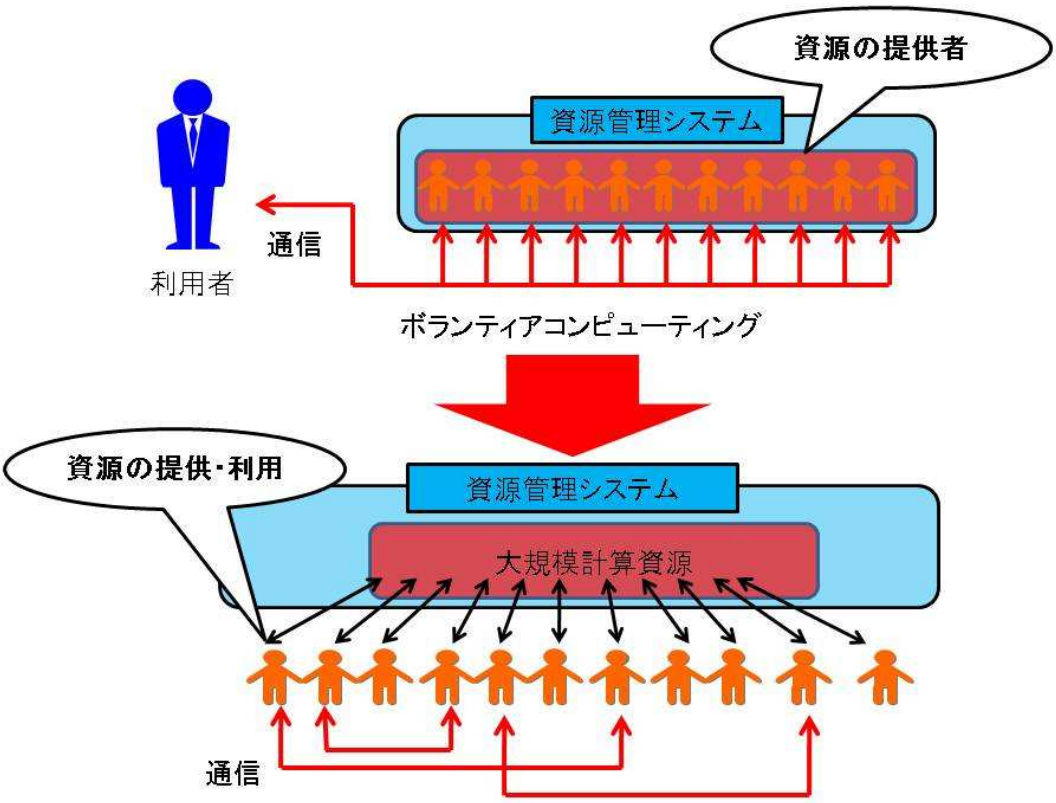


図 1.1: 目標システム概念図

さらにボランティアコンピューティングと異なる点として, 資源の提供・利用を対等な立場で行う本システムでは計算資源の管理者が存在しないことが挙げられる. また一般家庭の PC を参加ノードとして考慮していることから, システムへの参加・離脱が頻繁に起こることが想定される. したがって, 特定の管理サーバーを用いて計算資源の管理を行うことは困難であると考えられ, 管

理システムの実現は P2P 通信を用いたオーバーレイネットワークによって分散管理する必要があると考えられる.

以上の 2 点,

- 通信遅延を考慮した参加ノード管理システムの構築
- 上記管理システムを P2P 通信によるオーバーレイネットワークによって分散管理すること

を本研究の目的とする.

1.3 本論文の構成

第 1 章 序論

本研究の背景と目的について述べた.

第 2 章 オーバーレイネットワーク

管理システムが情報を分散管理するための手法としてオーバーレイネットワークに付いて述べ, その事例について述べる.

第 3 章 座標系による通信遅延推定

通信遅延を効率的に管理する手法として座標系による通信遅延推定を行う手法について述べ, 既存手法の事例について述べる.

第 4 章 階層型座標系による通信遅延管理

座標系を階層型座標系へ拡張することを提案する. また, 階層型座標系の座標情報を分散管理するためのオーバーレイネットワークの構築について述べる.

第 5 章 評価実験

提案手法の評価実験により既存手法との比較を行い, その結果及び考察を述べる.

第 6 章 結論

本研究の成果と今後の課題について述べる.

第2章

オーバーレイネットワーク

本章ではデータの分散管理を行うために必要であるオーバーレイネットワークに関する基礎的な事柄について述べる.

2.1 オーバーレイネットワーク

計算端末・ルータ・物理回線からなる現実に存在するネットワークをアンダーレイネットワークと呼ぶ. これに対し, アプリケーションレベルで仮想的なリンクを持つことで形成されるネットワークをオーバーレイネットワークと呼ぶ (図 2.1). オーバーレイネットワーク上では下位ネットワークを意識せずに通信を行うことが可能である.

2.1.1 オーバーレイネットワークの分類

オーバーレイネットワークは構造化オーバーレイネットワークと非構造化オーバーレイネットワークの 2 つに大別される. 仮想リンクを形成する際に, 何かしらの数学的規則を持ってリンクの形成を行うものを構造化オーバーレイ, 特に規則を持たずにリンクの形成を行うものを非構造化オーバーレイと呼ぶ.

したがって構造化オーバーレイは直接リンクを持つ端末を自由に選択することができず, 各参加ノードがあたえられる一意の ID に基づく規則によって厳格にリンクが決定される. 構造化オーバーレイネットワークはこのリンク生成規則により, ネットワークに接続される端末をほぼ確実に発見可能であり, また非常に効率的に探索を行うことが可能である. 構造化オーバーレイネット

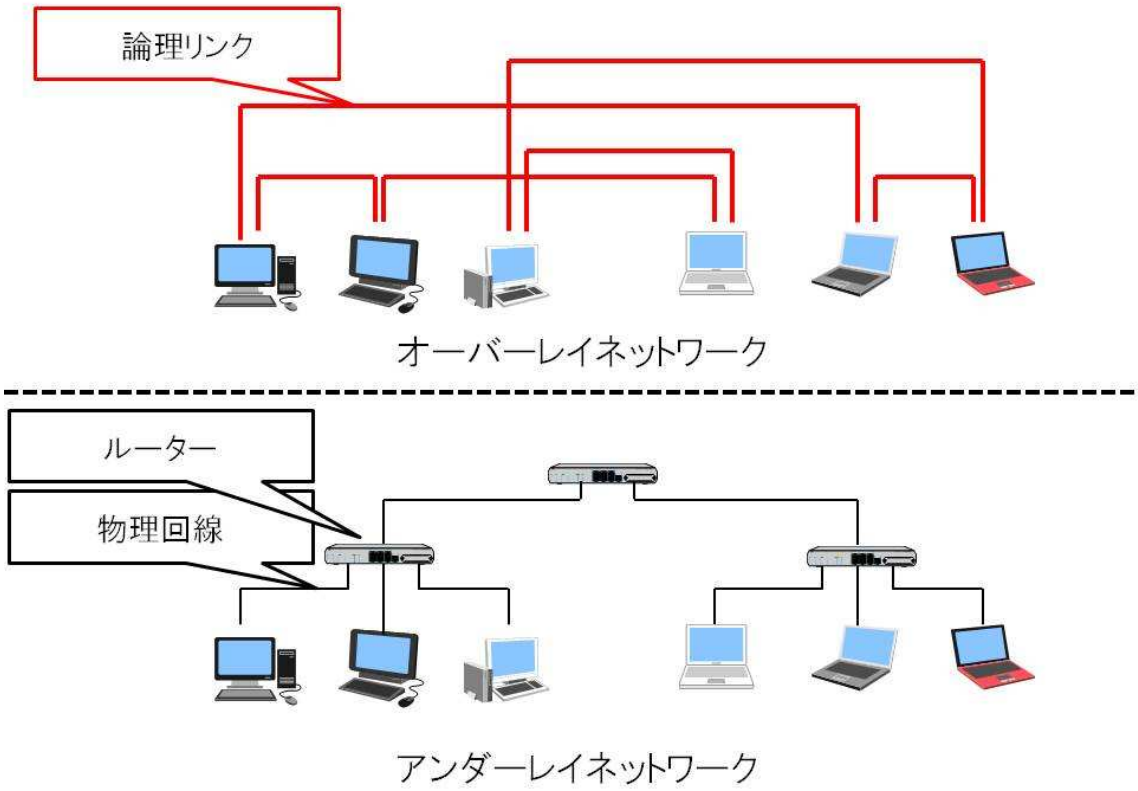


図 2.1: オーバーレイネットワーク

ワークの例としては, DHT(Distributed Hash Table) があげられる．DHT については詳細を後述する．

一方非構造化オーバーレイネットワークは構造化オーバーレイネットワークが考案される以前から存在していたものであり, その対比のために非構造化オーバーレイネットワークと呼ばれる．リンクの形成に規則が設けられていないため, 自由に選択して他の端末とリンクを形成することが可能である．非構造化オーバーレイネットワークにおけるメッセージの伝搬や端末の探索はフラッディングによって行われる．これはリンクを持つ全ての隣接端末にメッセージの送信を行う方式のため, メッセージに生存時間を定める必要があり, 確実な探索を保証することができない．

構造化オーバーレイネットワークと非構造化オーバーレイネットワークを比較した場合, 一般的に探索効率は構造化オーバーレイネットワークの方が優れているとされる．しかしこれは完全一致検索に対する効率であり, 部分一致検索や範囲検索などの柔軟な検索については非構造化オーバーレイネットワークの方が実現が用意である．非構造化オーバーレイネットワークの例としては, Winny [7] などのファイル共有ソフトが構築するネットワークがあげられる．

表 2.1: ハッシュ表の一例

key	value
red	apple
pink	peach
orange	orange
green	melon
yellow	banana

2.1.2 Distributed Hash Table

ハッシュ表は key となるオブジェクトから, 高速に value のオブジェクトを検索することを可能とする. 例として表のようなハッシュ表が存在する場合, key として red を入力すれば apple, yellow を入力すれば banana が得られる.

ハッシュ表によるデータの管理・検索は管理するデータの数が増加してもほぼ検索効率が低下しない. DHT(分散ハッシュ表) はハッシュ表を複数の計算機に分散して管理する手法であり, 構造化オーバーレイによって実現される. 著明な DHT のアルゴリズムとしては, Chord [8] や Kademlia [9] があげられる.

DHT に参加する端末の ID は各端末に一意のものである必要がある. そのため, IP アドレスなどの各端末に固有のものをハッシュ関数によりハッシュしたものを ID として利用する. ハッシュ関数としては 160 ビットの SHA-1 [10] がよく利用される. 分散管理を行うオブジェクト (key, value のペア) の ID についても, 端末の ID と同一の ID 空間を使うために key と同じハッシュ関数を用いてハッシュしたものを利用する. 一つの端末で複数のオブジェクトの管理を行うことが可能であるので, オブジェクトの ID については同一のものが存在してもよい.

DHT に要求される機能としては, (key, value) を適切な端末に保持させるための put, key に対応する端末から value を得るための get, オブジェクトの削除があげられる.

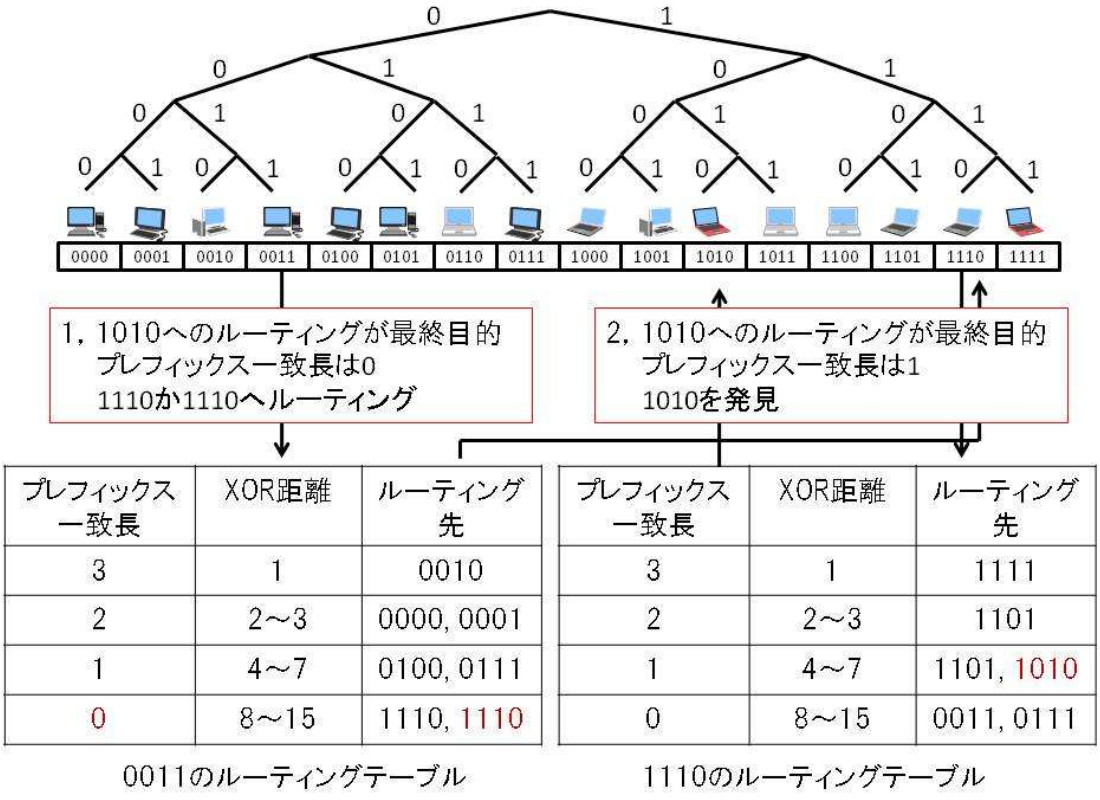
2.1.3 Kademlia

Kademlia は現在もっとも実用的に用いられているアルゴリズムのひとつである. 発表された際にプログラミングの内容にまで踏み込み, また実用性を重視したシンプルなアルゴリズムであるため広く受け入れられた. BitTorent などのシステムでも採用されている. また Kademlia をベースとした DHT の実装も数多い. Kademlia は 2 分木の構造の ID 空間に各端末を配置するものである. オブジェクトをどの端末が管理するかは, 双方の ID のプレフィックス一致長によって決定される. プレフィックス一致長とは 2 つの ID の上位何ビットが一致しているかを示す値である. プレ

フィックス一致長が最長のノードがオブジェクトの管理を行う．この方式は XOR(排他的論理和) を ID 空間の距離関数とすることで簡単に実現が可能である．

各端末のルーティングテーブルもプレフィックス一致長によって決定される．このルーティングテーブル, および Kademlia の概要を図 2.2 に示す．各プレフィックス一致長ごとに保持するルーティングの先のまとまりを K-Bucket と呼ぶ．k は一定の値であり, ID のビット数ごとに k 個のルーティング先が保持されるので, 各端末は最大で (k*ビット数) のルーティング先を保持する．ID 空間のビット数が多くなると空間が疎になるため, プレフィックス一致長の大きな k-Bucket は埋まらなくなり, ルーティングテーブルのサイズは全ノード数を N とすると $O(logN)$ に収まる．

Kademlia の特徴はその対称性であり, ノード A のルーティングテーブルにノード B が入っている場合にはノード B のルーティングテーブルにも通常ノード A が入っている．これによりルーティングテーブルの更新はメッセージの受信によって行うことが可能であり, ルーティングテーブルの更新に特別な通信が必要とされない．また Kademlia は k-Bucket の複数ノードに同時にメッセージを送信してルーティングを行うパラレル・ルックアップという特徴的なルーティングを行う．



プレフィックス一致長	XOR距離	ルーティング先
3	1	0010
2	2～3	0000, 0001
1	4～7	0100, 0111
0	8～15	1110, 1110

0011のルーティングテーブル

プレフィックス一致長	XOR距離	ルーティング先
3	1	1111
2	2～3	1101
1	4～7	1101, 1010
0	8～15	0011, 0111

1110のルーティングテーブル

図 2.2: kademlia 概念図及びルーティング例

2.2　第 2 章まとめ

本章ではオーバーレイネットワークについて概要を述べ, 情報を分散管理する方法について言及し具体例として Kademlia について述べた.

第3章

座標系による通信遅延管理

本章では計算資源の共有を行うための資源管理システムについて, システムに求められる要件を述べた後, 参加ノード間の通信遅延を管理する手法について述べ, 既存手法であるネットワーク座標系 Vivaldi [11] について説明をする.

3.1 計算資源管理システムに求められる要件

本システムにおいて参加者が並列分散計算を行う際には管理システムに利用可能ノードの検索を行うことが考えられるが, その検索に対して管理システムは以下の性能を持つ必要がある.

- 検索の確実性
- 計算性能による検索
- 通信遅延による検索

以下, それぞれの条件が必要とされる理由と利用例を述べる.

3.1.1 検索の確実性

参加者が並列計算を行うことを目的にシステムに参加していることから, 当然管理システムは他の参加者の中から検索者の希望に沿った, 他の利用可能な計算資源を通知することが必要とされる. したがって管理システムはシステムへの参加ノードの全てを検索可能であるべきである. また

参加者は自身が他の計算資源を利用するため, 遊休時間を他の参加者へ提供するものであると考えられる. よってシステムは参加者の中から最適な計算資源を通知するべきである.

計算資源の利用が確実に行えれば問題が無い例としてはボランティアコンピューティングで行われていた宇宙から飛来する電波の解析などが考えられる. データ量が膨大ではあるものの, 並列計算が可能であり, かつ結果が帰ってくれば参加者の要求を満たすことが可能である. 一般的な事例を考えれば, 指定された対象物が撮影された画像を元に, 多数の動画からその対象物が映っている動画を検索する場合などがあげられる.

3.1.2　計算性能による検索

研究機関がシステムへ参加することも可能であり, PC クラスタなどの巨大な計算資源のように特徴的な計算資源が利用可能になることも想定される. この場合にはシステムへの計算資源の提供に対する見返りや, 有料での提供となると思われるが, それでも PC クラスタによる信頼度の高い大規模計算に対する需要は存在すると考えられる. 製造業における大規模計算環境によるシミュレーションはこれからさらに需要が増えると予想さており, 様々な衝突や落下における強度, 建物の耐震性能の評価がなされると予測される. また金融分野におけるリスクマネージメントなどがこれからシミュレーションにより活発に行われると考えられる. さらには CG による映像作品の製作など, 信頼度が高い大規模計算資源は多岐に渡り利用が求められると考えられる. しかし企業が PC クラスタの構築を行い保有することは大きな負担になり, 一部をのぞいて多くの企業は PC クラスタを保有することは難しいと考えられる. したがってシステムにおいてこのような特徴的な計算資源を有する参加ノードを検索可能であることが要件としてあげる.

3.1.3　通信遅延による検索

計算機による処理を行うえで処理時間が重要になる事も多い. 本システムにおいてはノード間の通信時間が処理時間に含まれ無視できない大きさであると考えられるため, ノード間の通信時間の管理は必須である. 特に通信時間が重要であると考えられるのがリアルタイムに処理が必要な場合である. 一般家庭において大規模計算環境が必要とされる事例として, 部屋の気流を考慮した空調のシミュレーションや防犯・育児における監視カメラの画像解析, 物理モデルを完全再現するゲームなどがあげられる. これらの処理を行うためには随時計算資源に情報をインプットし解析などを行う必要があり, 処理による遅延が大きくなると性能に影響がでるため処理時間は短い方が好ましい. したがって通信遅延が小さいノードを随時取得できる検索が管理システムに求められる.

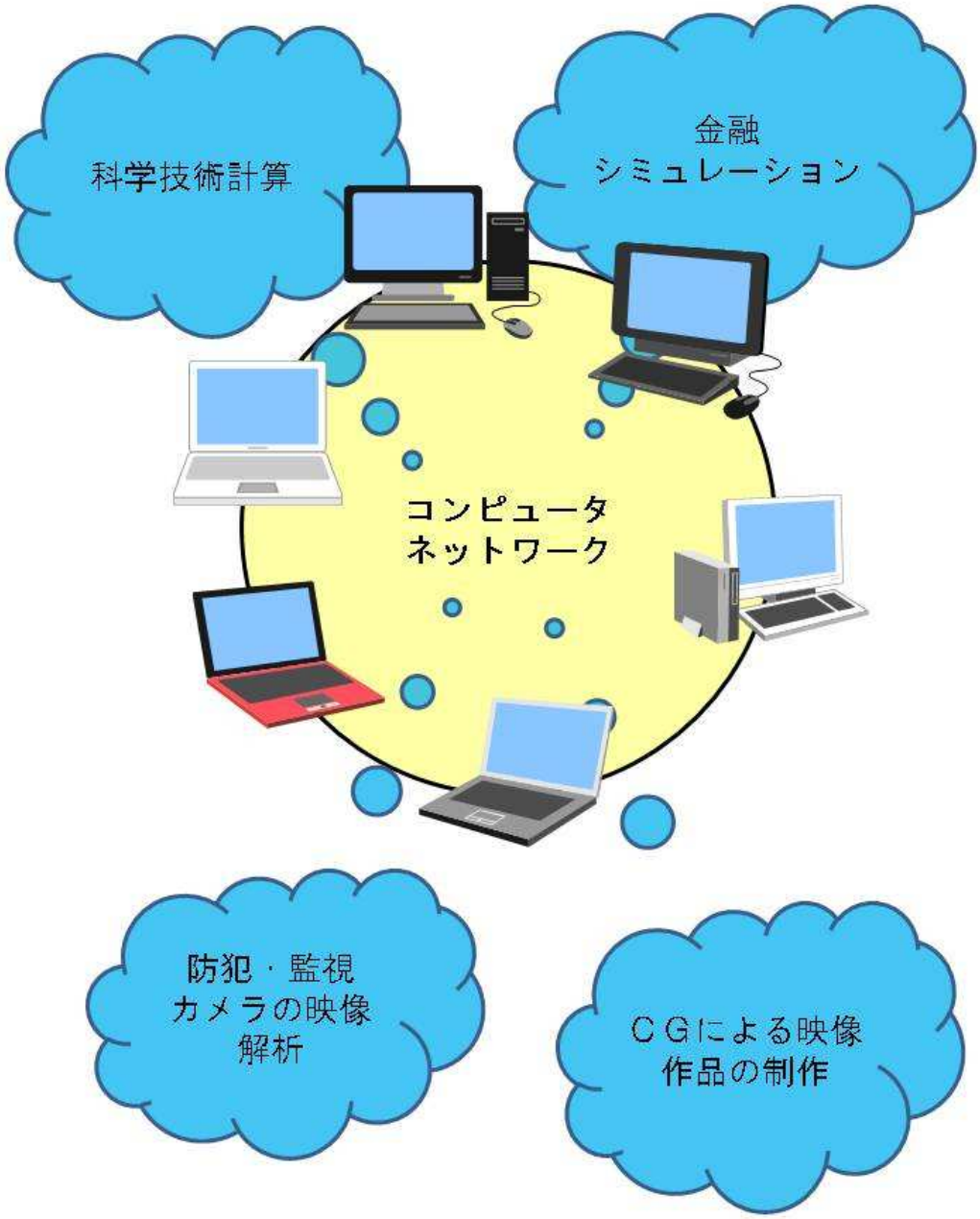


図 3.1: コンピュータネットワークを利用した大規模計算

3.2　計算資源間通信遅延の管理

本システムにおける参加ノードの検索要件について述べたが, 検索の確実性や参加ノードの性能による検索については参加ノード自身の情報をオーバーレイネットワークで分散管理することで実現可能である. 一方, 参加ノード間の通信遅延についてはその管理が難しい. まず, 単純に全ノード間の通信遅延をデータとして保持することが容易ではない. なぜならシステムへの参加ノードの数が計算環境の性能を左右するシステムであるため, 参加ノード数は莫大な数が想定される. またノードの参加・離脱が頻繁に起こる状況が想定されるために, ノードの参加が起こるたびに既に参加しているノードとの通信遅延を計測することは, ネットワークにおける通信コストから考えても現実的ではない. さらにデータの量が膨大であり, 分散して管理が行われるためにノード間の通信遅延自体を扱うためにはその更新に多大な手間が必要とされることが想定される.

以上の理由から通信遅延に関する管理システムとしては, 実際に通信遅延の測定を行わなくても推定可能であるものであり, さらに各ノード自身に付随する情報として表現可能であるものが望ましい.

3.2.1　座標系による通信遅延推定

座標系を利用し, 座標系におけるユークリッド距離でノード間の通信遅延を表現することを考える. この座標系がノード間の距離を正確に表現することが可能であるとすれば, 図 2.2 に示すとおり管理システムは全ノードの座標情報だけを保持すればよい. 各ノードはこの管理システムが保有する他ノードの座標情報にアクセスすることで他の全てのノードとの通信遅延情報を取得することが可能である. 新規ノードの参加が起きた場合には該当ノードの座標を決定するだけで管理システムへの更新が行われ, 全ノード間との通信遅延を測定せずに通信遅延を把握することができる. また全参加ノードにおいて統一的な座標系を構築することにより, 通信遅延が小さいノードを発見することが確実に可能であり, 利用可能な計算資源の中から最適なものを選択して利用することを可能とすることが期待できる.

3.2.2　座標系の欠点

ノード間通信遅延を座標系で表現することにより通信コストの削減や計算資源利用において近傍ノードの利用が確実に行えることが期待できることを述べたが, 座標系による通信遅延情報の管理には欠点が存在する. 最も重要な欠点として, そもそも精密な座標系を構築することがネットワークの構成上不可能であるということがあげられる. 実ネットワーク上では通信経路の性能差などにより, 図 2.3 に挙げるように三点間の通信遅延を表現する三辺が三角不等式を満たさないことがある. そのため座標系により表現不可能な場合が存在する. また, 低次元の座標系ではネットワークの複雑さによっては精密なノード間の位置関係を表現することが難しい. 一方高次元の座

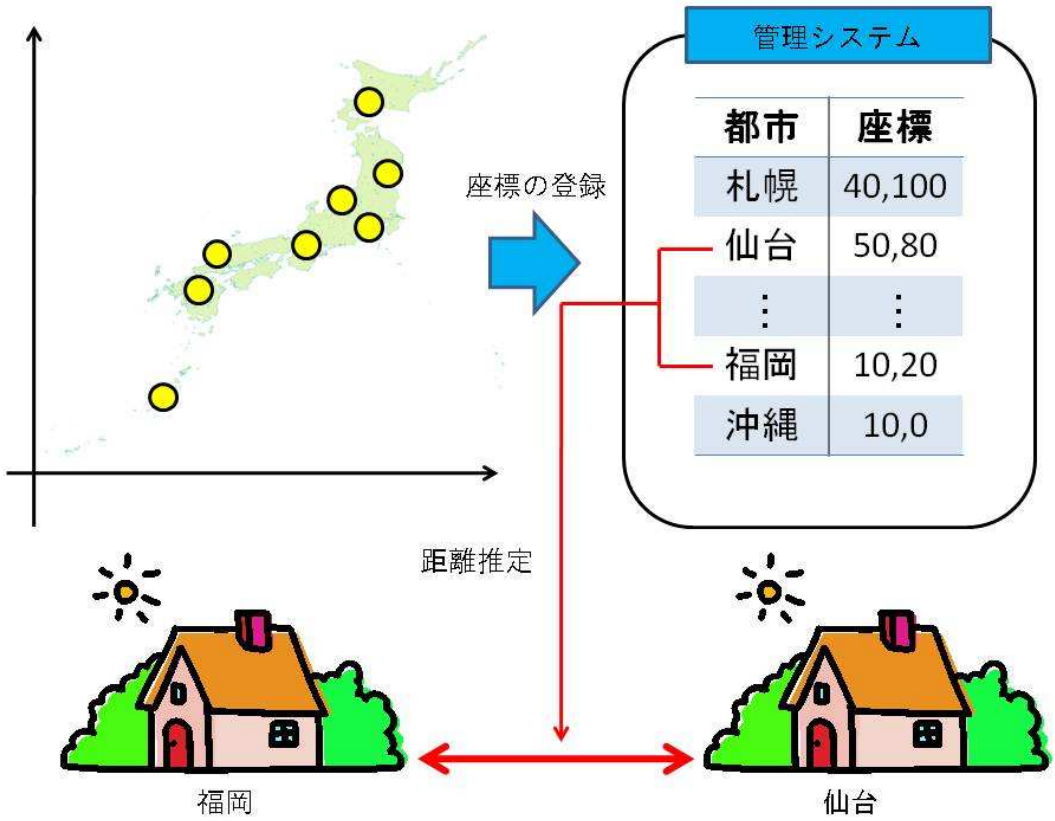


図 3.2: 座標系による通信遅延管理

標系を用いる場合には, 座標系を構築するために必要な通信遅延の計測が多く必要となり, 座標系の導入理由である通信遅延計測のための通信コストの削減を満足しない．さらにシステムへの参加ノードが刻一刻と変化することから, 全参加ノード間の適切な位置関係も常に変動し, 仮に精密な座標系が存在するとすれば, その座標系も常に変化を続けるといった問題がある．

3.3　ネットワーク座標系 Vivaldi

3.3.1　Vivaldi 概説

Debek らはネットワークに存在するノード間の通信遅延を座標系で表現する手法としてネットワーク座標系 Vivaldi の提案を行った．Vivaldi は通信を行ったノード間をばねでつなぐ力学モデルに基づく手法で, 通信を行うたびに自ノードの座標修正を逐次的に行うものである．ばねモデル

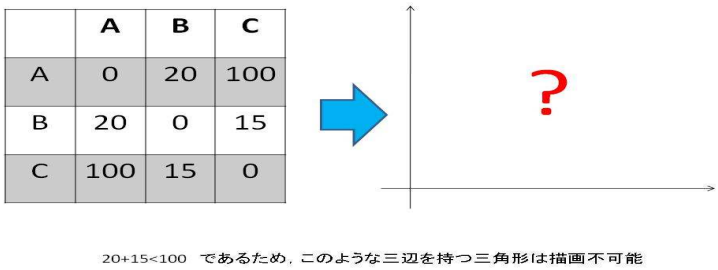


図 3.3: ノード間通信遅延が三角不等式を満たさない例

による修正を実現するため Vivaldi における座標修正は以下の式に基づいて行われる.

$$N_i(x,y) = N_i + \delta \times (rtt_{i,j} - |N_i - N_j|) \times u(N_i - N_j)$$

N_k はノード k の座標, δ は定数, $rtt_{i,j}$ はノード i とノード j との間で計測される通信遅延 (RTT), u は単位ベクトルを表す.

上記のアルゴリズムと式に従った座標修正を各ノードが自律的に行う. さらに Vivaldi の特徴としてあげられるのが, アプリケーションレベルの通信に便乗して通信遅延を計測することである. この特徴のため座標系を構築・維持するための通信が必要ないという利点を持つ. また通信を行うたびに通信対象との座標修正を逐次的行うため, ネットワークへの参加が保証されるノードとの位置関係を正しく保つ方向へ修正が行われるという特徴を持つ.

Vivaldi を 2D メッシュのネットワークに対して利用を行い, 撹拌されたネットワークが再構成される様子を図 2.4 に示す. 基本的にネットワークに対して正解となる座標系マップが存在する場合, Vivaldi は高い精度で座標系マップの構築を行うことが可能である.

3.3.2　Vivaldi の成果

座標系によるマッピングが 2D メッシュなどのネットワークにおいてはほぼ完全に可能であることを示す一方, Debek らは PlanetLab を利用した実ネットワークモデルに対する Vivaldi の利用・評価を行った. Debek らは Vivaldi の性能評価として, 座標系から推測される各ノード間の通信遅延と実通信遅延の相対誤差を評価指標として用い, 座標系として用いる次元や形状による誤差分布を評価した. その結果, 単純に次元を増やすことで誤差を小さくすることが可能であることや球形空間表示による座標空間は 2D 座標系に劣ることなどが示された.

特筆すべきは, 2D に高さベクトルを加え 2D+h 次元の座標系が考案され, この座標系においては図 2.5 のようにノード間の通信遅延が推定されるが, 単純に次元数を増やすよりも誤差を小さく

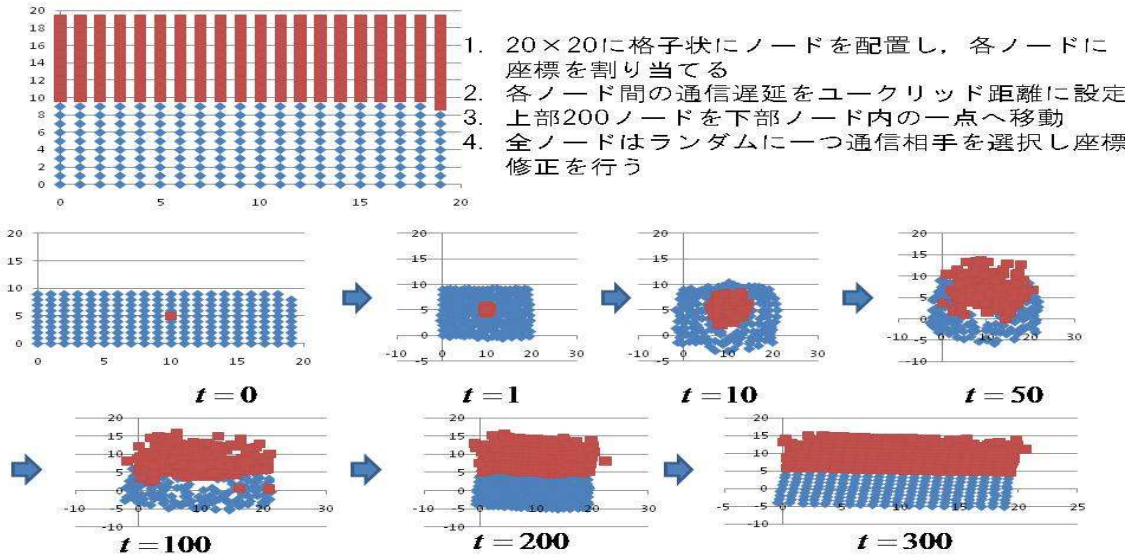
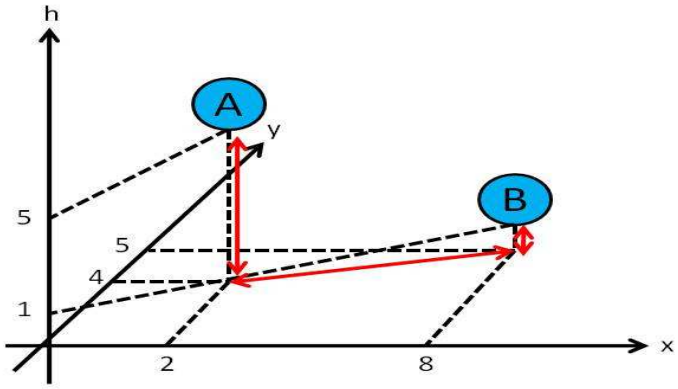


図 3.4: 2D メッシュネットワークにおける Vivaldi の動作例

できることが示されたことである．Debek らはこの結果から，次元数の増加とともに管理するデータ量が必要なことや性能を考えれば，2D+h 次元で表される座標系がネットワークにおけるノード間の通信遅延を表現することに対して最も優れていることを示唆した．



$$RTT = \sqrt{(x_a - x_b)\times(x_a - x_b) + (y_a - y_b)\times(y_a - y_b)} + h_a + h_b$$
$$= \sqrt{(2-8)\times(2-8) + (4-5)\times(4-5)} + 5 + 1$$
$$\approx 12.08276$$

図 3.5: 2D+h 次元座標系における通信遅延推定

3.4　Vivaldi の問題点

3.4.1　誤差の収束と通信遅延推定の差異

Vivaldi の座標修正式における δ はシステム全体の振動を小さくし, 座標を収束させるためにある程度小さい値を指定する必要がある. この理由によってネットワーク上で近傍に存在するノード群を複数座標系でマッピングした場合, 図 3.1 に示すような座標の収束が起こり適切な位置どりをすることができないノードが存在する可能性が考えられる.

各参加ノードが自身と最も通信遅延が小さいノード (もしくはノード群) の探索を可能とすることが本研究における座標系の導入理由である. したがって Debek らが評価したシステムにおけるノード間相対誤差は本研究が評価するものと若干の違いがある. つまり, 本研究では座標系から推定される近傍ノードが実ネットワークにおいても近傍ノードであることが重要であり, 相対誤差が小さくなったとしても, 座標系において推定される通信遅延により近さの順に並べられた順序が実ネットワークにおけるものと異なっては座標系による管理の効果が薄くなってしまう. 前述の適切な位置どりができないノードの存在により, 高さベクトルの導入により誤差の収束がなされたとしても, 正確な順序推定がなされない可能性は高く本研究において座標系の利用を行う際に障害となると考えられる.

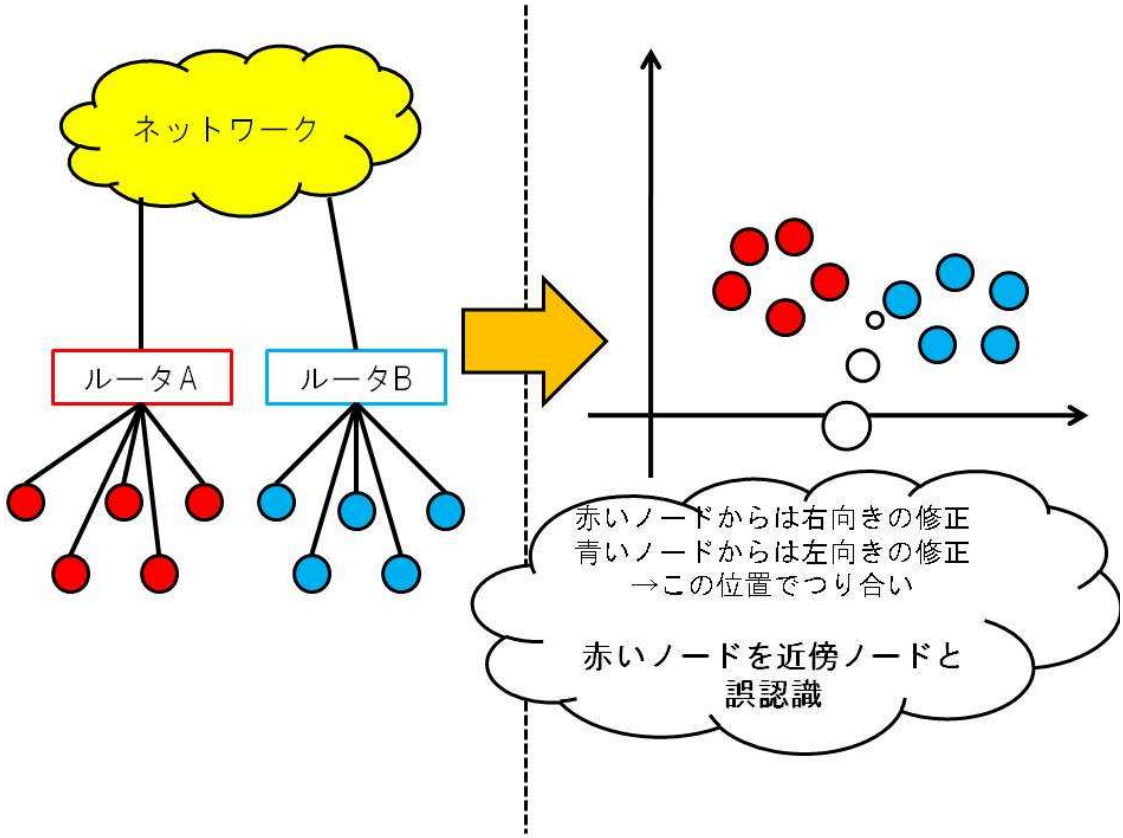


図 3.6: 近傍ノードの誤認識が起こる例

3.4.2　通信遅延推定精度の格差

Vivaldi はネットワーク全体における誤差を最小にする方向に動作するシステムである．そのため, ネットワーク上で近傍に存在するノードの集まりをひとつのノード群とすると, システムに参加しているノードの中で存在比率が高いノード群を中心に他のノード群が位置を修正していく．したがって存在比率が高いノード群は座標系から高い精度で他ノードとの通信遅延を推定することが可能である．しかしその他のノード群では中心ノード群との位置を保つ状態で座標を修正しなければいけないために, 低い次元の座標形では他のノード群との区別が困難になり推定精度が低くなってしまうという問題が起こりうる．これは前述の問題と同様に本研究における障害となると考えられる．

3.5　第 3 章まとめ

本章では計算資源の管理システムに求められる要件について述べ, 実現のために障害があるもののネットワーク座標系による通信遅延管理を導入することで情報の分散管理が容易になることや通信コストの削減が期待されることを述べた. また, ネットワーク座標系の既存手法として Vivaldi をとりあげ, その概要と成果, および問題点について述べた.

第4章

階層型座標系による通信遅延管理

本章では前章で取り上げた Vivaldi を本研究が目標とするシステムで利用する際に問題となる事項について述べ, その解決策として階層型座標系による通信遅延管理を行うことを提案する. また, 階層型座標系における各ノードの座標値を効率よく分散管理するためのオーバーレイネットワークの構築手法について提案する.

4.1 階層型座標系による拡張

計算資源共有を行うために参加ノード間の通信遅延を座標系で管理する際に起こる問題について述べたが, これらを座標系を拡張することで解消することを提案する. 本研究では座標系を図 4.1 のように階層型に拡張することで問題の解決を図る. 階層型に拡張する狙いは, 一つの座標系では不正確な推定しかできないノードを別の座標系に移動することにより, 推定が不正確なノード間で再び座標の適正化を行うことで推定精度をあげることである.

4.1.1 ノードの安定性

推定が正確に行えているノードと不正確なノードを分離することは階層型座標系を形成する上で不可欠である. 座標系から通信遅延を精密に推定できているかどうかを判断する指標として, ノードがその座標系で安定しているかどうかは役に立つ指標であると考えられる. なぜなら, 通信遅延を精密に推定できているノードは実際に通信を行った際に通信相手との座標系から得られる推定通信遅延と実通信遅延の誤差が小さく, 座標修正がほとんど行われないからである. したがってノー

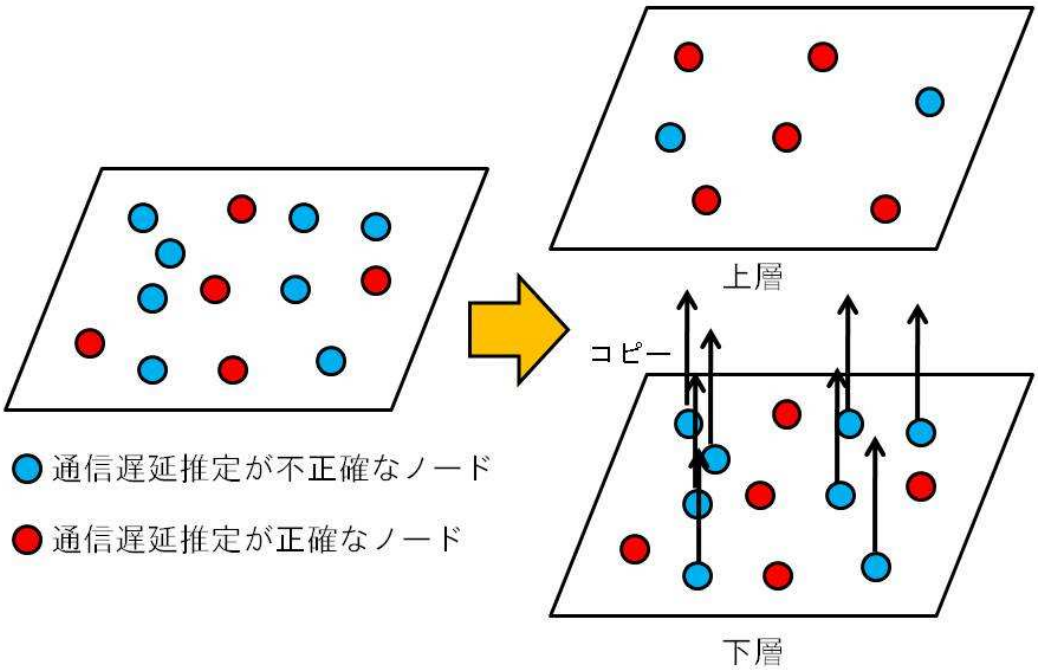


図 4.1: 階層型座標系概案

ドが座標修正を行う際に取得される誤差を元にノードの安定性を求めることで, ノードの分離が行えると考えられる.

4.1.2　安定度によるノードの階層移動

安定性の指標として, 以下の要領で求められる安定度 (*Stability(i)*) を各ノードの持つ情報として追加する.

$$Stability(i) \begin{cases} +1 & (Error(i,j) < e) \\ -1 & (Error(i,j) \geq e) \end{cases}$$

Error(i,j) はノード *i,j* 間で通信が行われた際に, 計測された実通信遅延と座標を修正する前の座標系から得られる推定通信遅延との誤差を表し, e は定数を表す.

ネットワークにおいて通信が十分に行われ座標修正によるノードの振動が微小になり, 座標系が一定の落ち着きを得た際には中心ノード群の安定度は増加を続け, 逆にその他のノードの安定度は減少を続けることが予想される. したがって一定の安定度を取得したノードが存在する座標系において, 安定度が負の値を持ったノードは不安定で座標修正を繰り返していると判断することができる. 提案手法では一定の閾値 (C) を設け, ある層の座標系において閾値を越える安定度を持ったノードが出現した際には安定度が負の値を持つノードを全て上層へコピーをする. これにより, 安定しているノードは同一座標系内で他のノードとの通信遅延推定が精密に行えるまま, 不安定なノードの安定化を別の層で行うことができる. これらのノードの上層への移動について図 4.2 に示す.

階層型座標系におけるノード間の通信遅延推定手法及び座標修正方法は後述するが, 一度不安定と判断し上層へコピーされたノードも下層において再び安定する場合が想定される. その際には安定した階層より上の階層に存在する自身を消去し, 階層の下降が行われることとなる.

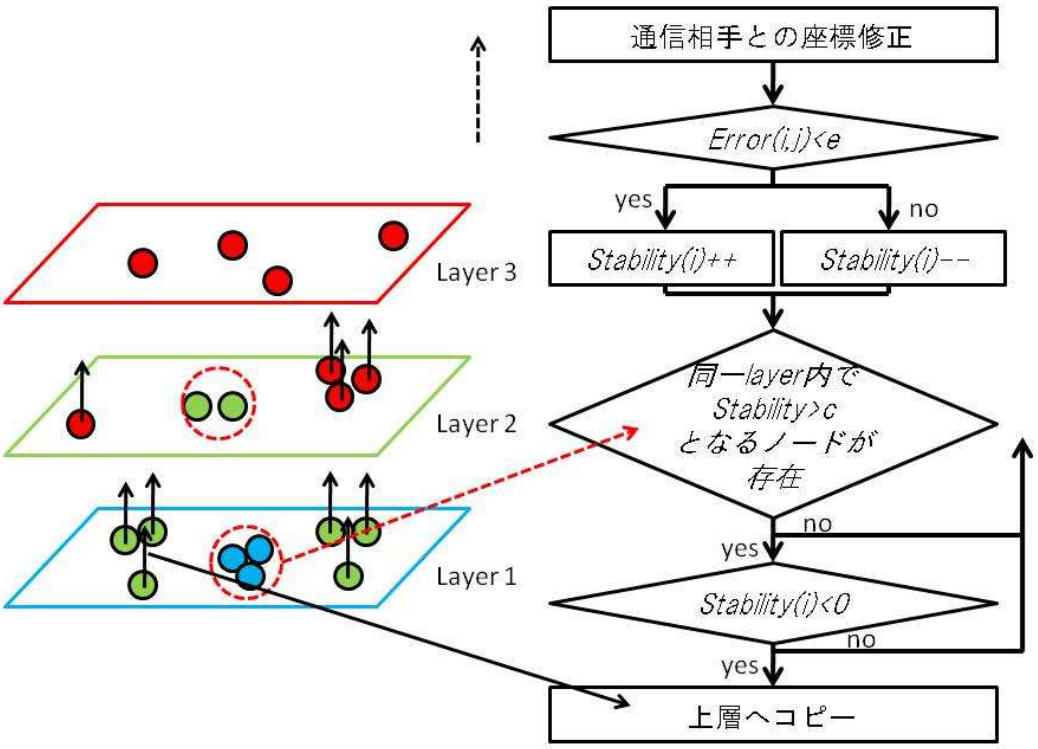


図 4.2: 階層型座標系におけるノードの上層への移動

4.1.3　階層型座標系における通信遅延推定

不安定なノードが上層へコピーされることにより, 一つのノードが多数の層に跨って存在するようになることが本手法の特徴であるが, その結果座標系から単純に通信遅延を推定することができなくなる. そこでふたつのノード間の通信遅延を推定する際に座標情報を取得する階層を, 同一に存在している階層のうち最も上の階層で行うこととする (図 4.3). ノードが存在する階層のうち上層にて比較を行うことで, 安定した状態で通信遅延推定を行うことを可能とし, また無駄に下層において座標修正を行って他のノードの安定を崩すこともない.

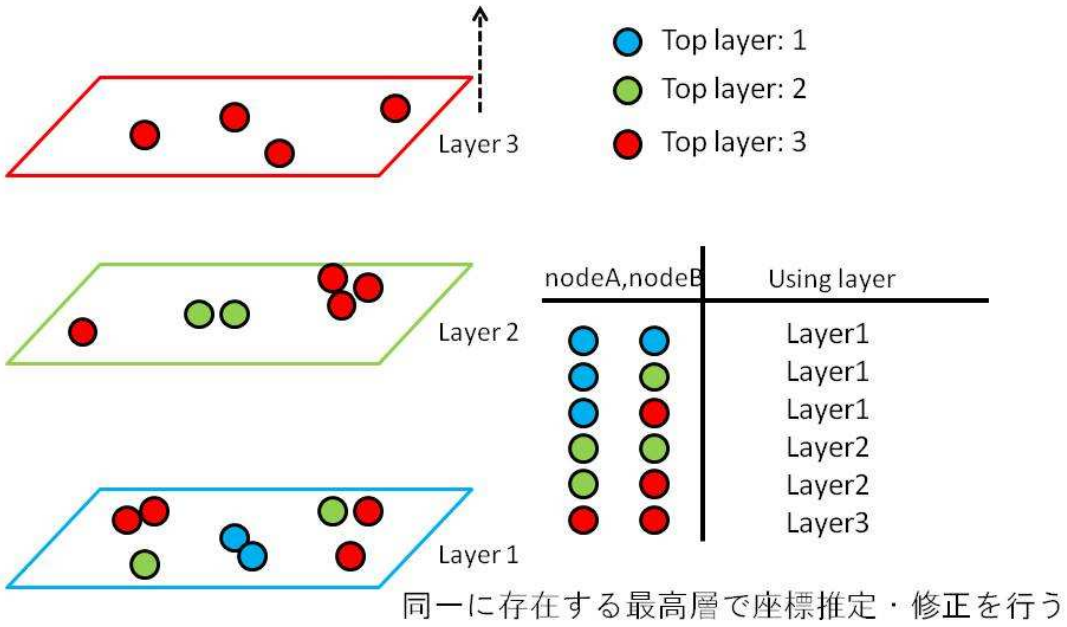


図 4.3: 階層型座標系での通信遅延推定・座標修正

4.2　参加ノード管理を行うオーバーレイネットワークの構築

システムへの参加ノードを管理するためのネットワークを構築することを考える. このネットワークが管理するべき情報は, 全参加ノードの座標値・性能・そして個別に認識を行うためのノー

ド ID である. 座標値については, 各ノードごとに所属する階層の数だけ座標値が存在する. また, 性能についても単純に CPU の処理性能のみについての情報や, GPU の性能, 平均遊休時間など様々な要素が存在することが想定される. これらの多様な情報を管理することを考えるが, 特に検索速度を気にしながら構築を行わなければいけないのは座標値であるから, 座標値に重点をおいてネットワークの構築を行う.

4.2.1　管理ネットワーク

各参加ノードは繰り返しではあるが, 所属する階層の数だけ座標値を持つ. したがって形成されている座標の数だけネットワークを構築し, そのネットワークを用いて各階層の座標値を管理することを考える. 各階層のネットワークにおいては, 安定ノード同士での検索が高速に行われるような設計をする必要がある. なぜなら階層型座標系では, その階層に存在するノードの中で最も存在比率が高いノード群とそのノード群と通信遅延が小さいノードが安定しやすいからである. すなわち, 安定ノード間の通信遅延は小さいことが期待される. 安定ノードについては全安定ノード間でリンクを持つネットワークを構築する. これにより安定ノード同士は 1hop で全ての安定ノードの座標値を検索することが可能となる.

次に各階層において所属するノードの座標を管理し, 効率的に階層に存在するノードの座標値を管理することを考える. これについては多次元空間においての検索効率や管理ノードの負荷分散を考えてネットワークを構築すればよく, Znet [**?**] [13] や GLOBASE.KOM [14] などその手法についても数多く提案されているため, これらを用いることで簡単に実現が可能である.

4.3　第 4 章まとめ

本章では Vivaldi の問題点を解消する手法について提案し, 提案手法で導入した指標や座標修正の方法について述べた. また提案手法を実現するためのオーバーレイネットワークの構築方法について述べた.

第5章

評価実験

シミュレーションにより提案手法と既存手法 Vivaldi の比較実験を行う. またシミュレーションから得られた座標値を元にオーバーレイネットワークを構築し, その性能について評価を行う.

5.1 実験方法と条件

実験は後述の仮想ネットワークにおいて通信遅延を各ノード間で設定し, この通信遅延を基に座標系を構築することで行う.

5.1.1 仮想ネットワークの構築

まず日本の都市人口分布とインターネットサービスプロバイダー (ISP) の利用人数動向を元に, ネットワーク上のノード群ごとのノード数を決定する. 本設定においては人口分布については総務省統計局が公表している統計データ [15] を参考にし, 表 4.1 を用いた. ISP 利用人数動向についてはインターネット白書 2010 [16] を参考に表 4.2 を用いた.

ネットワークのトポロジーについては「NTT コミュニケーションズグローバル IP ネットワーク」[17] を参考にし, 図 4.1 のように設定を行った. また, ISP 間を接続する Internet eXchange(IX) [18] との接続を全ての ISP で東京と大阪において行うこととし, ISP を跨ぐとしても東日本に存在するノード間の通信は東日本, 西日本に存在するノード間の通信は西日本で完結することとした.

表 5.1: 日本の主要都市人口

都市	人口 [百万人]	12 都市内での割合 [%]
Tokyo(区合計)	8.5	30.5
Yokohama	3.6	12.9
Osaka	2.6	9.3
Nagoya	2.2	7.9
Sapporo	1.9	6.8
Kobe	1.5	5.4
Kyoto	1.5	5.4
Fukuoka	1.4	5.0
Kawasaki	1.3	4.7
Saitama	1.2	4.3
Hiroshima	1.2	4.3
Sendai	1.0	3.5

表 5.2: 日本の ISP 利用動向

ISP	利用者割合 [%]	6ISP 内の割合 [%]
OCN	16.5	29.4
Yahoo!BB	13.3	24.1
BIGLOBE	7.2	13.1
@nifty	6.7	12.2
ぷらら	6.6	12.0
au one net	5.0	9.2

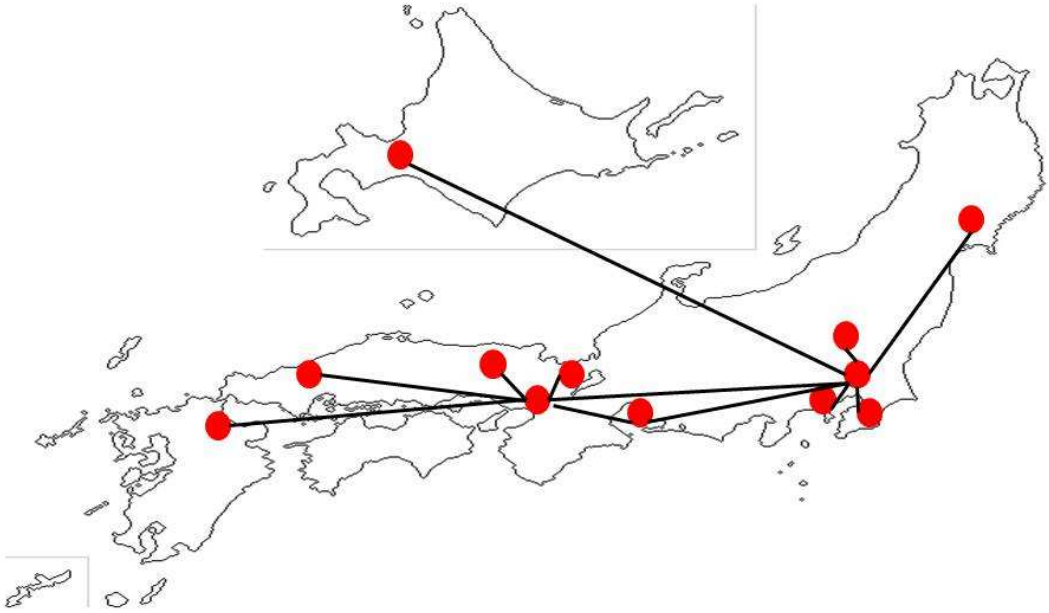


図 5.1: ISP 内 12 都市間ネットワーク

5.1.2　実験手順

システムに参加するノード数を 2000 とし, 各ノードが他のノードと通信を行い座標の修正を行う動作を 2000 回繰り返した. 通信相手の選定方法は以下である.

$$\text{階層型座標系}\begin{cases}\text{同一階層かつ推定 } RTT < 5 & (95\%)\\ \text{ランダム} & (5\%)\end{cases}$$

$$Vivaldi\begin{cases}\text{推定 } RTT < 5 & (95\%)\\ \text{ランダム} & (5\%)\end{cases}$$

ランダムな通信相手を選択する理由として, ノードのネットワーク座標系全体における位置関係を正しく保つために近傍ノード以外との通信が必要とされるためである. なお, 95%で選択される条件式から得られる通信相手が存在しない場合もランダムに通信相手を選定する.

5.1.3　その他の設定

座標の修正を行う際に用いられる δ とし, 階層型座標系の階層移動に関するパラメータとして, 安定度の増減に関する指標 *Error*, 安定ノードの出現を判断する閾値 c を以下のように定めた.

$$\delta = 0.05$$
$$Error = 1.0$$
$$c = 200$$

5.1.4　評価項目

まず提案手法で用いる安定度が目的に沿って求められることを確認するため, 既存手法において安定度分布の推移と座標系をマッピングした際の中心ノード群の安定度を確認する.

その後既存手法と提案手法の比較評価を行うため, 2 つの手法を用いて top-k 検索の精度を比較する. 本研究における top-k 検索とは, システムに参加しているノードが座標系から推測される通信遅延に基づき自身に近い順に k 個のノードを検索することである. 並列分散環境を構築するうえで, 一つの処理内容が実行完了するまでの時間は, 並列化を行った際に利用した計算資源の中でもっとも通信遅延が大きいものに左右されると考えられるため, top-k 検索によって取得された k 個のノードのうち実際の通信遅延が最も大きいノードとの通信遅延を本実験での評価指標とした. この指標として以下で定義される *topk_e* を用いる.

$$topk\_e(k) = \frac{estimateRTT(k) - realRTT(k)}{realRTT(k)}$$

estimateRTT(k) は座標系から推測された最近傍 k 個のノードとの間で実通信遅延と設定された値の最大値, *realRTT(k)* は仮想ネットワークの設定の時点で取得できる最近傍 k 個のノードとの間に設定された通信遅延の最大値である.

最後にに階層型座標系によるマッピングの変化を確認する.

5.2　実験結果

5.2.1　安定度の確認

既存手法における安定度分布の推移を図 4.2 に, 安定度が *c(=200)* を越えるノードの所属ノード群推移を図 4.3 に示す. また OCN に所属するノード群の 2000Step 座標修正を行った際のマッピングを図 4.4 に示すが, 座標系の中心に位置するノード群の安定度が, ノードの安定条件を満たす数値を満足しているものの中で高い割合を占めていることが確認できる.

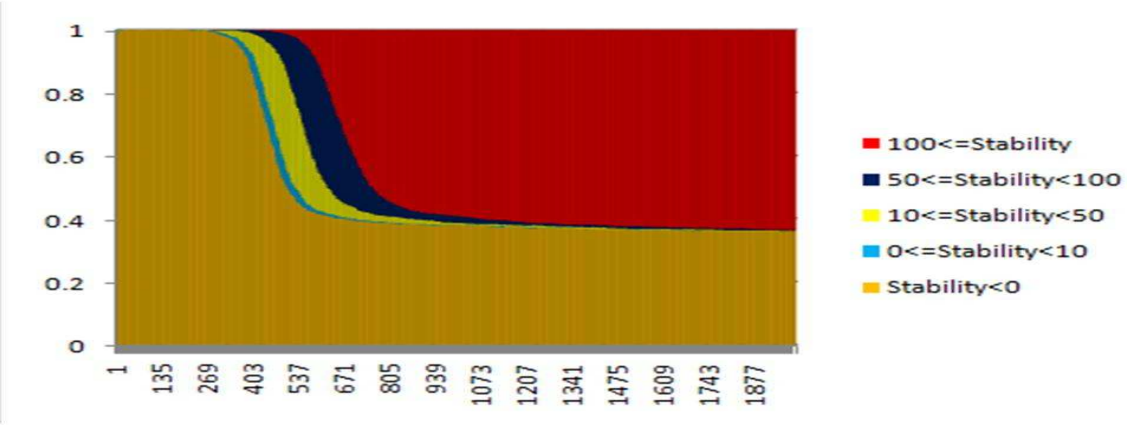


図 5.2: 安定度分布の推移

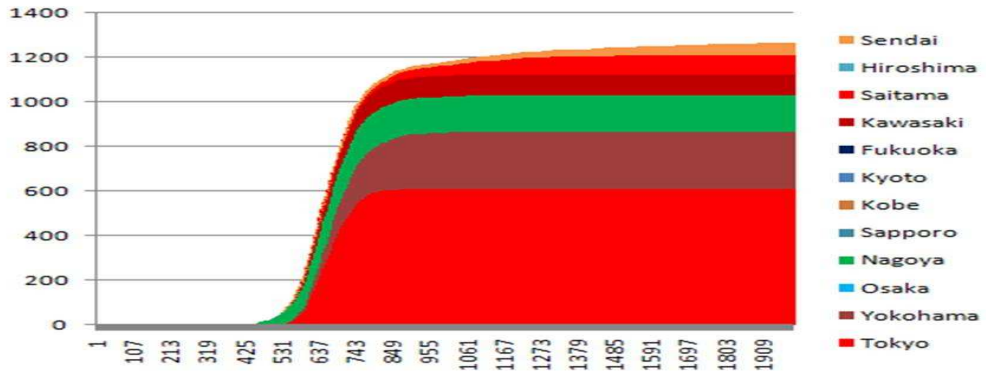


図 5.3: 高安定度ノードの所属ノード群

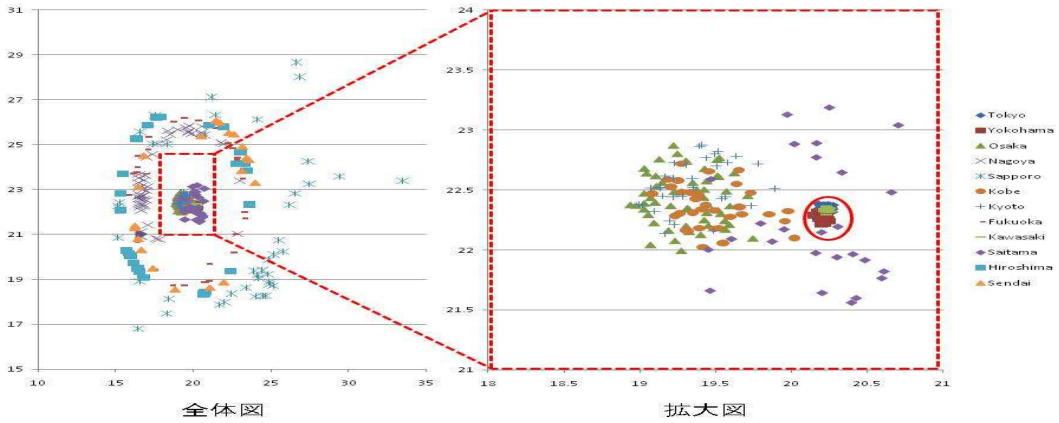


図 5.4: 単一レイヤーによる座標系収束

5.2.2　top-k 検索

k=1,5,10,30,100 の 5 つのパターンについて top-k 検索の精度を比較した．それぞれ既存手法と提案手法の結果を図 4.5 から図 4.9 に示す．

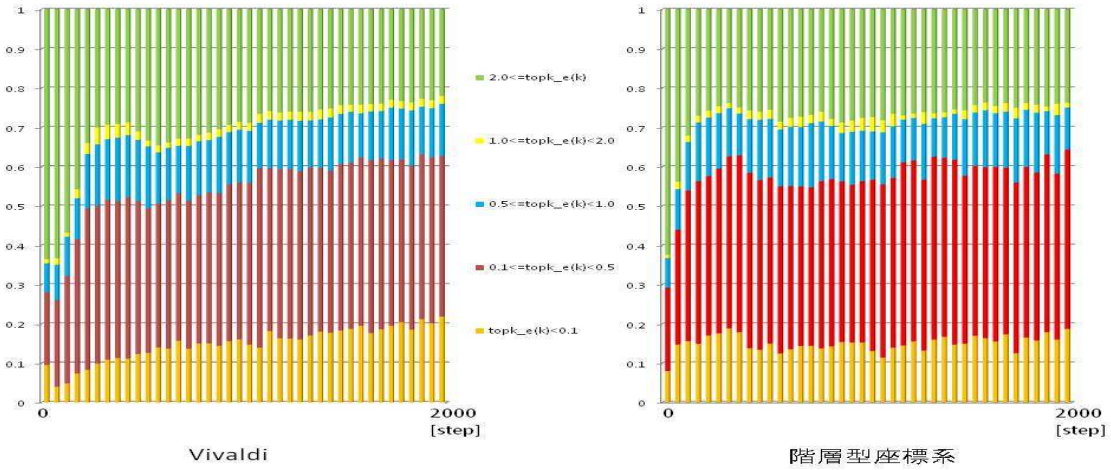


図 5.5: top-1 検索精度評価結果

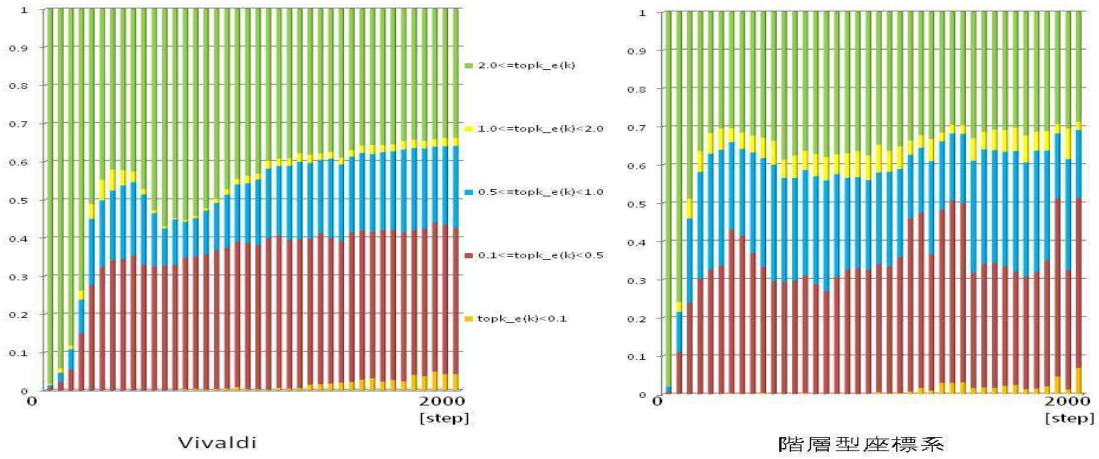


図 5.6: top-5 検索精度評価結果

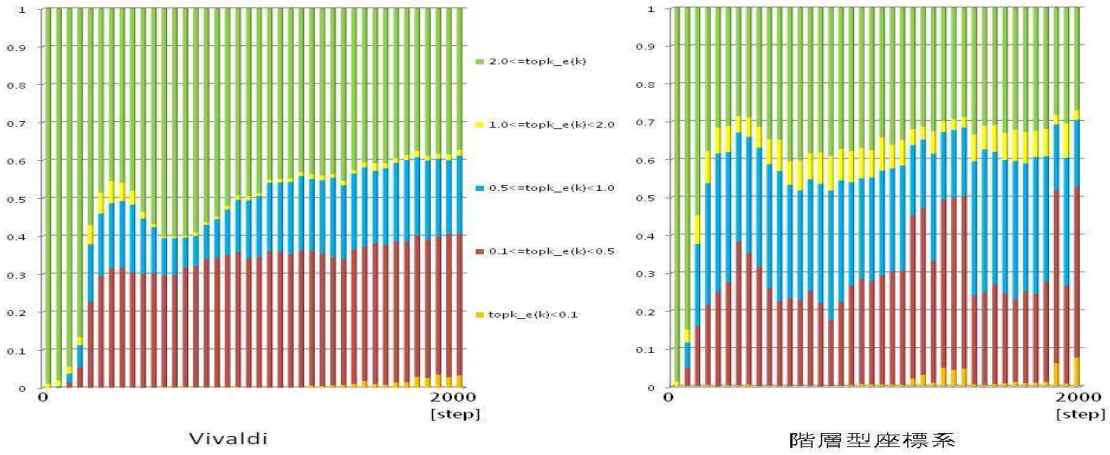


図 5.7: top-10 検索精度評価結果

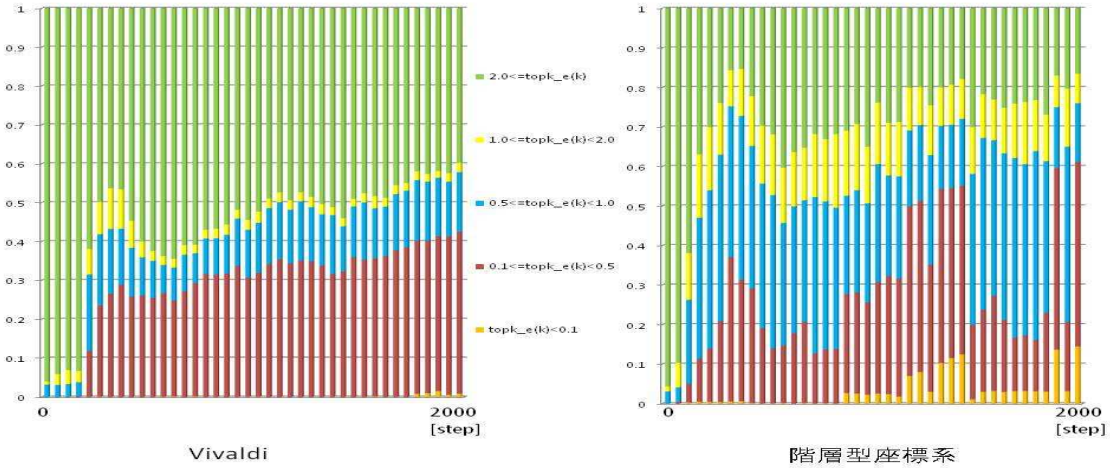


図 5.8: top-30 検索精度評価結果

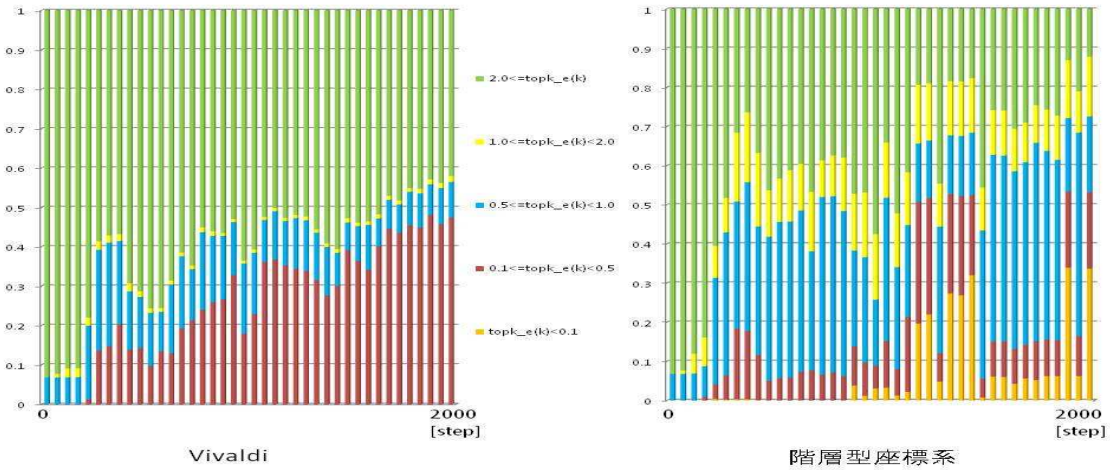


図 5.9: top-100 検索精度評価結果

5.2.3　マッピングの変化

2000Step の通信を完了した際 OCN に所属するノードについて, 各階層ごとにマッピングを行った. その結果を図 4.10 から図 4.14 に示す.

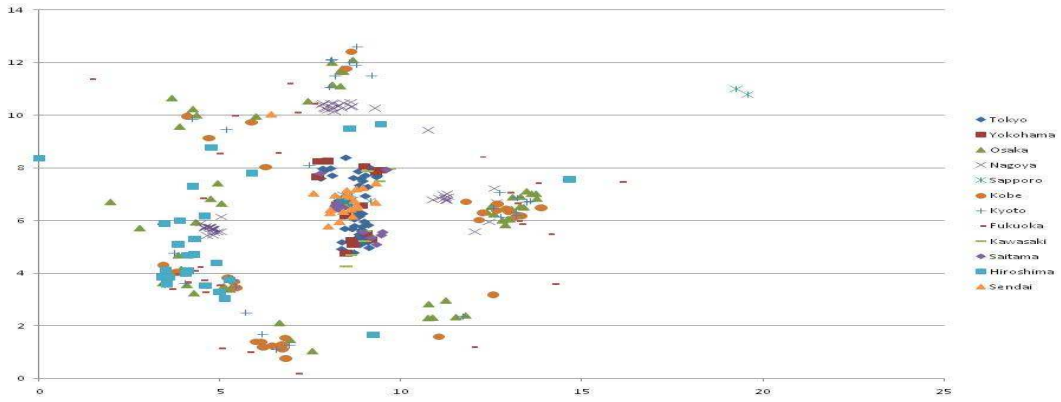


図 5.10: layer0(最下層) におけるマッピング結果

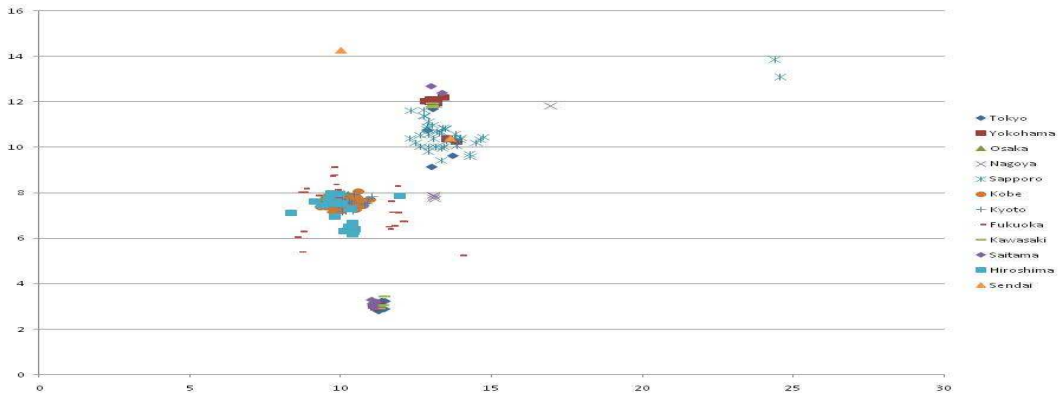


図 5.11: layer1 におけるマッピング結果

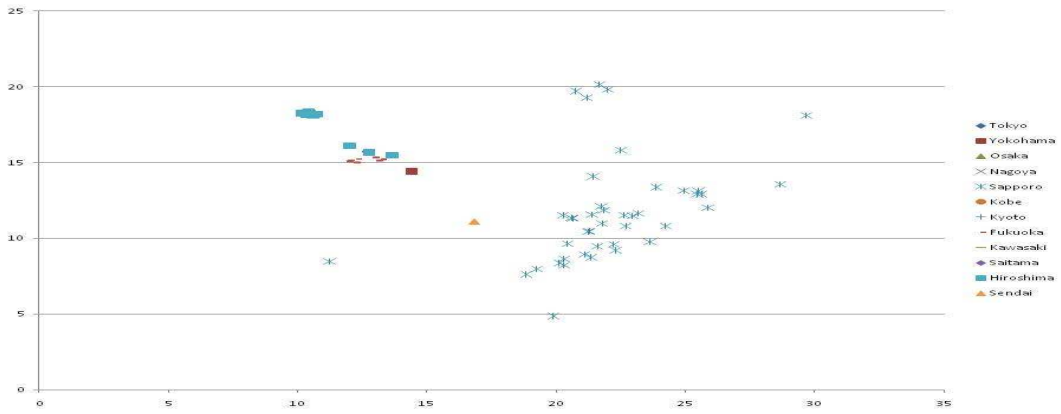


図 5.12: layer2 におけるマッピング結果

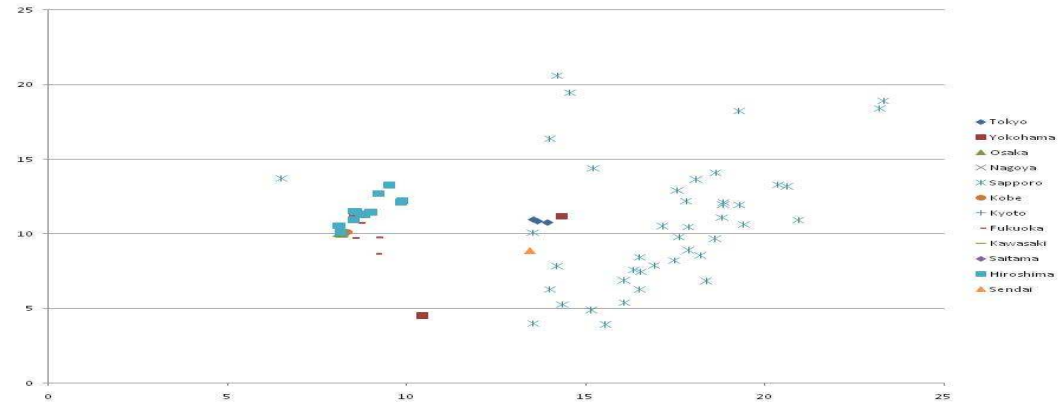


図 5.13: layer3 におけるマッピング結果

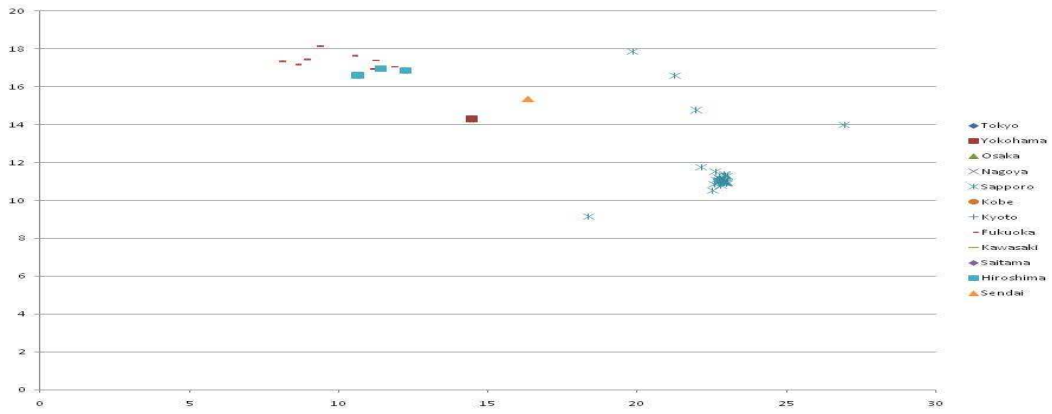


図 5.14: layer4(最上層) におけるマッピング結果

5.3　考察

top-k 検索精度やマッピングの変化から, 既存手法に対する問題点が提案手法によって若干ながら改善されていることが, top-k 検索における $topk\_e(k) < 0.1$ となる割合の増加によって示された. しかしながらさらなる精度向上が目的とするシステムに必要とされると考えられるため, さらに精度の向上を行うことを今後の課題としたい.

top-k 検索において, $topk\_e(k) < 0.1$ を満たす割合について k=5,30 とした際の変化と提案手法におけるノードの所属レイヤの割合について図 4.15 に示す. 提案手法が既存手法よりも優れる Step 帯が多いものの, 提案手法の推定精度が安定せず上下に大きく振動していることがわかる. ノードの層移動が起きている Step 帯において振動が著しいが, これはノードの層移動によって移動先の層における不安定さが招いていると考えられる.

5.4　第 5 章まとめ

シミュレーションにより提案手法と既存手法の比較を行った. その結果として, 満足できるほどではないものの近傍ノードの検索制度の向上が確認できた. また, 実際に座標系についてマッピングを行ったところ, 提案手法の狙いである不安定ノードの分離ができていることを確認できた.

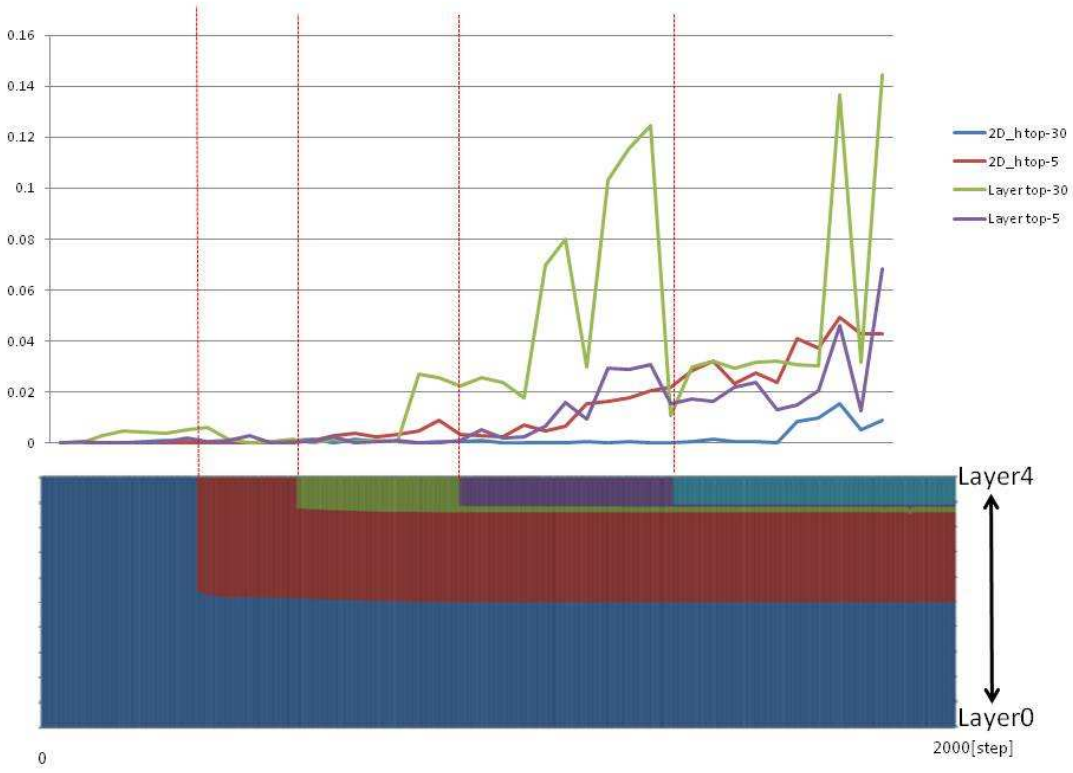


図 5.15: top-k 検索精度と階層分布の比較

第6章

結論

6.1 本研究の成果

大規模並列分散環境をネットワーク上に存在する計算資源によって構築するために, 参加ノード間の通信を最適化するための手法として座標系による通信遅延の管理手法が効果的であることを述べた. 本研究では階層型座標系の構築による通信遅延管理、及び階層型座標系による管理を実現するためのオーバーレイネットワークの構築手法の提案を行った. 提案手法の性能評価としてネットワーク座標系 Vivaldi との比較実験を行ったところ, 近傍ノード推定を行った際に精度の向上がみられることを示した. また, 提案手法によって形成された座標系を実際にマッピングすることで, 階層型座標系に拡張したことで不安定ノードの分離が可能となったことを示した.

6.2 今後の課題

不安定ノードの分離を行う際に単純に不安定ノードをひとつ上の階層にコピーすることで分離を行っているが, これが一時的な座標系の乱れを招いていると考えられるため, 分離を行った際に座標系の乱れが大きくならない改良が必要である. コピーを行った際に座標の修正値を大きくし, 早期に安定するようにする手法や, 下層において安定しているノードをランドマークとして上層にコピーする手法が考えられるが, これらの手法について検討・導入を行うことを今後の課題としたい.

また, 実際にネットワーク上で運用を行うためにはノードの情報を分散管理するためのオーバーレイネットワークの構築が必須であるので, その構築及び評価実験を今後の課題としたい.

参考文献

[1] インテルミュージアム「マイクロプロセッサーの歴史」: http://www.intel.co.jp/jp/intel/museum/hof/index.html

[2] Fred Pollack: New Microarchitecture Challenges in the Coming Generations of CMOS Process Technologies. Micro-32, 1999.

[3] Top500 : http://www.top500.org/

[4] Charles E. Leiserson, Zhi S. Abuhamadeh, David C. Douglas, Carl R. Feynman, Mahesh N. Gamukhi, Jeffer V. Hill, W. Daniel Hills, Bradley C. Kuszmaul, Margaret A. St. Pierre, David S. Wells, Monica C. Wong-Chan, Shaw-Wen Yang, and Robert Zak: The Network Architecture of the Connection Machine CM-5, Thinking Machines Corporation, Cambridge, Massachusetts 02142, February 7, 1996.

[5] SETI@home: http://setiathome.berkeley.edu/

[6] Ian Foster: What is Grid? A Three Point Checklist, Argonne National Laboratory and University of Chicago, July20, 2002.

[7] 金子勇, Winny の技術, アスキー, October 2005.

[8] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. Proceedings of A CM SIGCOMM 2001, pp. 149-160, August 2001.

[9] Peter Maymounkov and David Maziéres. Kademlia: A Peer-to-peer Information System Based on the XOR Metric. *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS02)*, pp. 53-65, March 2002

[10] D.Eastlake, 3rd and P.Jones. RFC3174: US Secure Hash Algorithm 1(SHA1): http://fhp.rfc-editor.org/in-notes/rfc3174.txt, September2001.

[11] Frans Kaashoek, Frank Debek, Russ Cox and Robert Morris. Vivaldi: A decentralized network coordinate system. *Proceeding of the ACM SIGCOMM '04 Conference*, pp. 149-160, Portland, Oregon, August2004.

[12] Shu Y., Ooi B., C. Tan, K.-L, and Zhou A.: Supporting multi-dimentional range queries in peer-to-peer systems, *Proceedings of Fifth IEEE International Conference on Peer-to-Peer Computing(P2P 2005)*, pp. 173-180, 2005.

[13] Tropf H., and Herozog H.: Multidimentional Range Serch in Dynamically Balanced Trees, *Angewandte Informatik(Applied Informatics)*, Wiesbaden, Germany, Viewg Verlag, pp. 71-77, February 1981.

[14] Aleksandra Kovavević, Nicolas Liebau, and Rald Steinmetz, Globase.KOM - A P2P Overlay for Fully Retrievable Location-based Search, *proceeding of Seventh IEEE International Conference on Peer-to-Peer Computing*, pp. 87-94, 2007.

[15] 総務省統計局　政策統括官 (統計基準担当)　研修所 : http://www.stat.go.jp/data/kokusei/2010/index.html

[16] 「インターネット白書 2010」, インプレス R&D インターネットメディア総合研究所 (著), 財団法人インターネット協会 (監)

[17] IP バックボーン : http://www.ocn.ne.jp/business/bocn/backbone/

[18] WIDE プロジェクト: http://www.wide.ad.jp/index-j.html

謝辞

本研究を進めるにあたり, ゼミやさまざまな機会を通じて御指導を頂きました, 東北大学大学院工学研究科教授 大町真一郎先生に深く感謝を申し上げます．この大町研究室で研究する機会を与えてくださったことに感謝いたします．本論文をまとめるにあたり, 貴重かつ有益なご助言を数多く頂きました東北大学大学院情報科学研究科応用情報科学専攻 加藤寧教授, 東北大学大学院工学研究科技術社会システム専攻斎藤浩海教授に深く感謝致します. 助教 菅谷先生から研究を進めていく上でたくさんの御指導と, 激励を頂きました．暖かく研究を見守ってくださったことに深く感謝申し上げます.

大町研究室の先生方をはじめ, 先輩, そして同期のみなさんと過ごすことのできたこの期間に特別に感謝申し上げます．みなさんと学生生活を共有することができたことに感謝しています．みなさんがいたからこそ, この素晴らしい日々がありました．深く感謝申し上げます.

また, 私に勉強する機会と生活を与えてくださっている父母, 家族に深い感謝を申し上げます.

研究業績

学会発表等 (査読なし)

ロ頭発表

中村貴, 福士将, 堀口進, 菅谷至寛, 阿曽弘具
ネットワーク座標系 Vivaldi の適性評価
平成 21 年度 電気関係学会東北支部連合大会 講演論文集,
p.71, 2009-3.

中村貴, 菅谷至寛, 大町真一郎
余剰計算資源共有を行うための通信遅延を考慮したネットワークの構築
情報処理学会研究報告,2010-DPS-142 No,11,
2010-3.

中村貴, 菅谷至寛, 大町真一郎
ネットワーク座標系によるノード間の通信時間管理手法の検討
情報処理学会研究報告,2011-DPS-146,
2011-3.