分散ハッシュテーブルChordにおけるノードの動的再配置方式

†嶋野裕太　†佐藤文明
東邦大学理学部情報科学科

1　はじめに

分散ハッシュテーブル(以下 DHT)は、P2P 通信におけるノードの検索方式として注目されている。Chord は、ノードを論理的なリング構造に構成し、リング状に検索要求が転送される。効率的な検索をするためには、検索を始めてから結果が返ってくるまでの時間が短いほうがいいが、リングを構成するノードの物理的な配置がランダムだと、検索要求を点祖する遅延が大きくなる問題がある。本研究では、Chord リングのノードの配置を動的に再配置することで、検索要求の転送時間を削減する方式を提案する。

2　関連研究

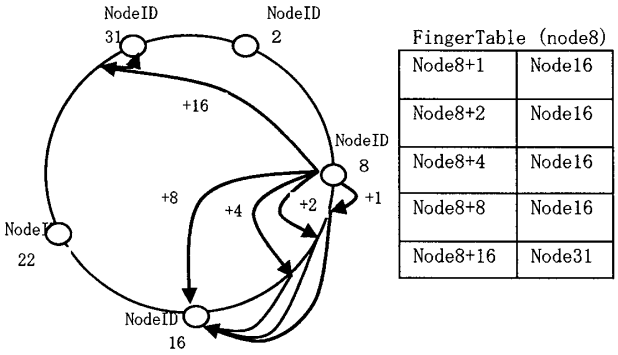
本研究の基礎となっている Chord[1]では、キーID およびノード ID は SHA-1 などのハッシュ関数を用いて作成され、一次元ハッシュ空間にマッピングされる(キーID＝hash(key)、ノード ID＝hash(IP address))。

Chord ネットワークにおいて、ID を持った各ノードは仮想的なリングを形成し、時計方向に接続を行っている。この接続はノード同士で作られる接続であるため、ノード間には複数のルータが存在し、ネットワークの近さをまったく考慮しない仮想的なネットワークである。各キーの値とデータのペアは、時計回りに最も近いノードに格納される。

検索をする場合、時計回りに隣のノードへ検索要求を転送していく場合、ノード数Nで構成されるネットワークの検索にO(N)のホップ数がかかることになってしまう。Chord では性能を改善するために、フィンガーテーブルを用いた高速な参照方法を定義している。

フィンガーテーブルとは、5bit のハッシュ空間を用いて 32 の ID が存在する場合、自ノードの ID より 2＾k(0<=K)先のノードの IP アドレスを常に保持しているテーブルであり、このテーブルを用いて検索キーをショートカットしクエリの転送を行う。よって logN のルーティングテーブルを保持することになるので、O(logN)のホップ数に抑えることができる。

Dynamic relocation method of node in Distributed Hash Table Chord
†Shimano Yuta , †Fumiaki Sato ・Toho University , Faculty of Science , Department of Information Science.



FingerTable (node8)

Node8+1	Node16
Node8+2	Node16
Node8+4	Node16
Node8+8	Node16
Node8+16	Node31

図1.Chordアルゴリズム

また、Chord を基礎とし仮想ネットワークの問題点を改良した手法として HIERAS[2]、LCLV[3]、ネットワークアドレスを元にノード ID を生成する[4]、等の手法が提案されている。

これらの手法に共通することは、実ネットワークの構造をノード ID 生成時に反映させることで検索の効率化、高速化を図っている。ただし、仮想ネットワーク構成後、動的にノード ID を変更していくことはされていない。

3　提案方式

3．1　目的

前節で述べたように Chord において構成される仮想ネットワークは実ネットワークの近さをまったく考慮していない。関連研究では、ノード ID 生成時にのみ実ネットワークを考慮している。そこで本研究では、ノード ID 生成時ではなく、一度構成された仮想ネットワークでのノード ID を交換していく。実ネットワーク上で遅延の少ない近いノードを仮想ネットワーク上でも近くに置くことで無駄なホップを減らし検索の高速化を図る。

3．2　基本設計

Step1.自ノード発の検索クエリを転送するとき、フィンガーテーブルを参照した場合のみ転送先のノードを target_node とし、遅延時間を計る。計測された遅延時間を自ノード前後のノードとの遅延時間と比較し、target_node への遅延が小さい場合に Step2 に移行する。

Step2.自ノードは target_node に交換候補先を伝えるよう要求する。target_node は前後のノード間の遅延が大きいノードを要求元ノードに

交換候補先として伝えStep3へと移行する。

Step3. 現在前後にあるノードの遅延総和より、交換した場合前後に来るノードの遅延総和が小さい場合に交換手続きを開始する。

その理由として交換要求を出したAは付近のノードが実ネットワーク上で近いノードになったとしても、交換に応じたBの付近がBにとって遠いノードにならないようにするためである。

以下に具体的な流れを示す。

図ではノード14からノード28へ検索クエリが転送され、ノード14の前後のノード10及びノード16への遅延より、ノード28への遅延が小さいとする。ノード14はノード28をtarget_nodeとし、その前後どちらかのノード(22か31)を交換候補とする。

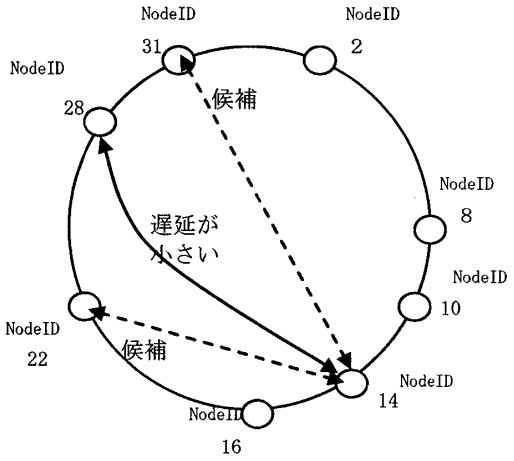


図2. 動的なノードの変更

ノード28からノード31への遅延のほうが大きいと仮定し、実際にIDを交換するに当たって満たすべき条件は、『ノード14からノード10(以下14-10)の遅延、14-16の遅延、31-28の遅延、31-2の遅延』の総和よりも、『14-28の遅延、14-2の遅延、31-10の遅延、31-16の遅延』の総和が小さい場合とする。

このような交換を行っていくことで、実ネットワークで近いノードが仮想ネットワーク上でも近くなるようにしていく。

4 シミュレーション

4. 1 シミュレーション環境

10bitハッシュ空間に100から500ノード、計5パターンのノード数を用意した。各ノードから約50回検索し、それを10回繰り返した。総遅延時間から検索一回あたりの平均遅延時間を計算し比較した。ネットワークトポロジーはGT-ITM[5]というトポロジー生成ツールを使用した。

4. 2 シミュレーション結果

図3に、シミュレーション結果を示す。

各ノード数それぞれ約11%、10%、8%、6%、7%、遅延時間改善が見られた。

早い段階で交換先を発見していった場合、大きく遅延時間の改善が見られたが、交換先が見つからない、または発見が遅れてしまうと改善はあまり見られなかった。

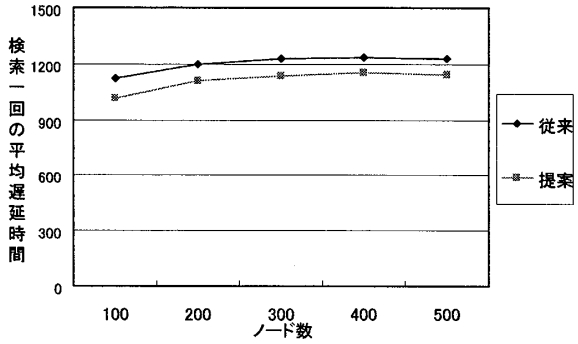


図3. 平均遅延時間の測定結果

5 おわりに

本稿では、Chordの仮想ネットワークを改善する手法を提案した。一度仮想ネットワークを構築した後にノードのIDを交換していくことで検索の高速化を図り、従来方式よりも約11%から6%程の検索時間の改善が見られた。

しかしながら、交換手続きにおける通信及び仮想ネットワーク維持の通信のコストが増える問題点がある。

また、現段階ではまだ無駄なホップをする検索が残っているので改善の余地は残っている。

今後の課題はそれらの問題点の解決である。

参考文献

[1]Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp "A Scalable Content-Addressable Network", In Proceedings of ACM SIGCOMM, 2001

[2] Zhiyong Xu、Rui min、Yiming Hu "HIERAS: A DHT Based Hierarchical P2P Routing Algorithm" icpp, pp.187, 2003 International Conference on Parallel Processing (ICPP'03), 2003

[3] 羽場 裕介、 松尾 啓志 "物理ネットワークの状況を考慮した階層型分散ハッシュ法の提案" 情報処理学会研究報告. UBI, [ユビキタスコンピューティングシステム] 2006(14) pp.43-48 20060216

[4] 縣 亮、金子 豊、堀内 幸夫 "DHTを利用したデータ検索システムの高速化手法" 電子情報通信学会技術研究報告. IN, 情報ネットワーク 106(237) pp.121-126 20060907

[5] Megan Thomas and Ellen W. Zegura. "Generation and Analysis of Random Graphs to Model Internetworks." Technical Report GIT-CC-94-46, College of Computing, Georgia Tech