Kuriltai: 局所性を考慮したID空間を持つDHT

入江 道生　中山 泰一
irie-m@igo.cs.uec.ac.jp, yasu@cs.uec.ac.jp
電気通信大学 情報工学科

概要

DHT はオーバレイネットワーク上に分散配置されたハッシュ表であり、多くの分散システムにおいてデータの管理に利用されている。本稿では DHT の幾つかの問題点のうち、データの探索時間とデータの可用性の問題に注目した。既存の研究では DHT における探索時間の縮小やデータの可用性の向上は個別に論じられており、複数の特性を同時に持たせた DHT については議論されていない。本研究で提案する Kuriltai は、ルータレベルの局所性を考慮した ID 空間を持つ DHT である。更に ID 空間を幾つかの部分 ID 空間に分割し、各部分空間に全データの複製を配置することで高い可用性と効率の良い探索を実現した。シミュレーションの結果から、Kuriltai が規模が大きく不安定な状態のネットワークにおいても高速な探索と高いデータの可用性を共に実現していることを確認した。

1　はじめに

近年、個々の計算機の性能向上や通信環境の向上を受けて、従来のクライアント・サーバシステムの他に、Peer-to-Peer (P2P) 技術を用いた分散システムが多く利用されてきている。その P2P 技術における重要な基礎技術の一つに、Distributed Hash Table (DHT) がある。DHT とは、複数のノードで構成されるオーバレイネットワーク上でハッシュ表を分散・共有するもので、様々な実現方法が提案されている [1, 2, 3, 4]。DHT は分散ファイルシステム [5] やコンテンツ配信システム [6] を始めとして、様々な分散システムでデータの管理のために用いられている。

DHT の主な機能は、データの格納と対象データの取得である。DHT を構成するノードはそれぞれ一意の ID を持ち、他の幾つかのノードの ID とアドレスを記したアドレス表を持つことによって相互に接続されている。ID 空間とはそのノードの ID の位置関係をマッピングした仮想的な空間のことで、各ノードを 2 次元の円周上へと対応付けしたものが用いられることが多い。DHT 上のデータはキーと呼ばれる値を持ち、そのキーに最も近い ID のノードに保持される。DHT においてデータを取得する場合、対象データのキーを目標としてデータの探索要求が出される。探索要求を受け取ったノードは、自身のアドレス表を参照することで探索要求を次にどのノードに転送すべきか判断する。最終的に、探索要求は目標のキーの値に最も近い ID を持つノードに転送されることが保証されている。最初に探索要求を発行したノードが、対象のデータを保持するノードからこれを受け取ることで、データの取得は完了する。

DHT を構成するノードの数を N とした場合、従来の DHT の多くでは O(log$N$) 回のホップで探索を行うことによるスケーラビリティに重点が置かれていた [2, 3, 4]。しかし一方において、既存の最大規模の P2P システムにおいてもそのノード数は 1,000,000 前後であり、利用されているオーバレイネットワークの多くが更に小規模であるという現状がある。ホップ数に比例したネットワーク遅延によって、O(log$N$) ホップの DHT の多くが現在用いられている規模のオーバレイネットワークで利用するには低速であるという問題から、1, 2 ホップの固定ホップ数で探索を行う DHT が提案されている [7, 8]。これらの DHT は個々のノードに大きなアドレス表を持たせることで、探索時のホップ数を O(1) まで減らしているものの、扱えるネットワークの規模に限界がありスケーラビリティが無い。また、ホップ数の縮小による高速化とは別に、ルータレベルの局所性を考慮することで 1 ホップ毎の遅延を縮小する研究も行われている。

更に、データの可用性や一貫性の問題等、DHT の様々な問題についての研究がなされているが、それらは個々に独立しており、別の問題に対する他の手法と組み合わせた場合に生じる問題等については論じられていない。我々は、DHT の更なる実用化のためにはこれらの問題が個別に議論され

Kuriltai: An Efficient DHT with Locality-Aware ID Space
Michio IRIE and Yasuichi NAKAYAMA
Department of Computer Science, The University of Electro-Communications

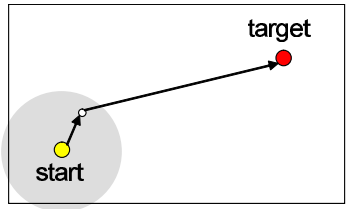

図 1: 転送元から近い転送先の選択

るだけでは不十分だと考え、それらの手法を統合した例として Kuriltai という DHT を提案し、その設計と性能について検討を行った。

Kuriltai ではルータレベルの局所性を考慮した ID 空間を用いることによって、探索時のホップ毎の遅延を縮小している。更に、Tourist[9] というネットワークの規模と安定度に適応性の高い DHT を基に、ID 空間を局所性に沿って分割することで探索毎のホップ数をも縮小する。Kuriltai では全てのデータを分割された各部分 ID 空間に複製することで、探索効率を上げながら高い可用性を実現している。

本論文の構成は次の通りである。2 章ではまず関連する研究について述べ、その後 Kuriltai のベースである Tourist について説明する。3 章で局所性を考慮することによる探索速度の高速化について述べ、4 章で ID 空間を分割することで得られる特性について論じる。5 章でシミュレーションによる評価を行い、最後に 6 章で本稿をまとめる。

2 関連研究

Kuriltai はネットワークの規模によらない高速な探索速度と高いデータの可用性の獲得を目標とした DHT である。そこで本章ではまず、探索速度とデータの可用性に関連する研究について述べ、その後で Kuriltai のベースである Tourist という DHT について論じる。

2.1 探索の高速化

DHT におけるデータの探索は一般に低速であるが、それには主に二つの原因がある。一つは、データの探索時に対象キーを保持するノードへ到達するまでのホップ数が多いという問題である。これは、各ノードの保持するアドレス表のサイズが小さい、あるいは全体のノード数が膨大であるために相対的に個々のアドレス表に保持されるノード数が少ないことによる。そしてもう一つは、ネットワーク遅延のために探索の 1 ホップ毎にかかる時間が長いという問題である。

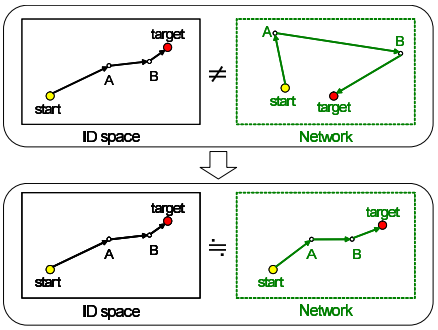

図 2: 局所性を考慮した ID 空間

前者の問題については、前章で説明したように各ノードの保持するアドレス表のサイズを大きくすることでホップ数を減少することができる。Gupta ら [7] は、1 ホップで探索を行うものと 2 ホップで探索を行うものの 2 種類の DHT を提案している。また、Kelips[8] は 2 ホップで探索を行う DHT で、アドレス表の更新の通知にゴシップアルゴリズムを用いる。ゴシップアルゴリズムは小規模なネットワークにおいては有効に機能するが、大規模なネットワークにおいては通知に 1 時間以上を要するため、利用できる状況が限定される。これらは共に探索にかかるホップ数が固定であることからスケーラビリティが無く、極度にノード数の多いシステムでの利用については考えられていない。

後者の問題である 1 ホップ毎の遅延については、ルータレベルでの局所性を考慮したルーティングを行うことでこれを縮小できる。ここでも、局所性を考慮したルーティングの手法は主に二種類に大別できる。

一つは、自身の保持するアドレス表に含まれるノードのうち、ルータレベルで距離の近いノードを優先的に探索要求の転送先として選択する方法。Kademlia[4] や後述する Tourist 等の DHT がこれに該当する。この分類には、Pastry[3] や Tulip[10] のように、通常のアドレス表の他にルータレベルで近いノードを集めたアドレス表を保持する手法をも含む。これらの手法はともに、探索要求の転送元と転送先のノード間のルータレベルにおける距離が短いこと、即ち初めに行う 1 ホップにかかる時間が短いことを保証する (図 1)。この分類の手法では、探索要求の転送先たりうる候補が多ければ多いほどその効果が大きく現れる。

もう一つの分類は、ID 空間そのものにルータレベルの局所性を反映する手法である。一般に DHT における探索は、経由するノードの ID が目標の

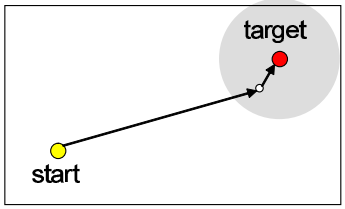

図 3: 目標のキーに近い転送先の選択

キーに近づくよう、探索要求の次の転送先を選択していく。これに、ルータレベルで距離の近いノードが ID 空間上においても近距離にあるという性質を与えることで、通常の探索と同じ手順を踏みながら局所性を考慮したルーティングが可能となる (図 2)。この手法を用いた DHT には、局所性を考慮した CAN[11], Toplus[12], Zhou ら [13] によるものがある。この手法を用いることで、従来の DHT から探索の方法を変えることなく局所性への考慮を採り入れることが可能となる。一方で、ID 空間の均一性が失われるためにノード毎に負担するデータ量や通信量に不公平が生じてしまう。Toplus はその最たる例である。これは IP アドレスを直にノードの ID とした DHT で、アドレス表としてルータのルーティング表をそのまま利用する。探索はルータレベルの経路を模倣するため、非常に高速に行われる。しかし ID 空間においてノードの密度に極端な偏りが生じるためにノード間での不公平が大きく、実用的ではない。Zhou らの手法はノード ID の上位ビットをルータレベルでの位置を表す値で置き換えることにより、局所性への考慮と ID 空間の均一性との間でバランスを取っている。これらの手法は、探索要求の転送先のノードと目標のキーを持つノードとの距離が短いことを保証する (図 3)。

Kuriltai では Tourist を拡張し、Zhou らの提案する局所性を考慮した ID 空間と統合することで、ホップ数の縮小と局所性への考慮の両面から探索速度の向上を図っている。

2.2 可用性の向上

DHT におけるデータの可用性とは、あるデータを探索した際にそのデータが正しく取得できる可能性のことである。DHT においてはノードが予告なく DHT ネットワークから離脱する可能性が高く、個々のデータの可用性が低くなっている。そこでデータの可用性を高めるため、複製や Erasure Coding を用いてデータに冗長性を持たせることが多い。

複製を利用する研究として、Knezevic ら [14] の研究を挙げる。これは全ての DHT で利用可能な、データの複製を配置して可用性を高めるラッパーである。高い汎用性と可用性を備えている反面、複製したデータを配置するノードの選択はランダムで、複製を利用した探索の高速化には言及されていない。

Erasure Coding は、データを幾つかの断片に分割し冗長符号を付与することで、断片の一部が失われた場合にも元のデータを復元することを可能とする手法である。これは単純な複製に比べて記憶領域の使用効率に優れるものの、DHT のようにノードの参加と離脱の頻度の高いシステムにおいて小さいデータに対して用いる場合は目立った効果が得られないことが報告されている [15]。

Kuriltai では ID 空間を分割し、部分 ID 空間毎に複製を配置することでデータの可用性を高めると同時に探索の効率をも向上させている。

2.3 Tourist

Tourist は Twins[16] という 2 ホップ DHT を拡張したもので、SmartBoa[17] と同様のネットワークの規模と安定度[1] に対する適応性を持つ DHT である。Tourist はごく小規模で安定したオーバレイネットワークにおいては 1 ホップで探索を行い、ネットワークの規模が拡大し安定度が減少するにつれて 2 ホップないし 3 ホップ以上の探索を行うよう、動的に変化していく。

Tourist の各ノードは 2 つのアドレス表を持つ。一方は ID 空間において近いノードのアドレスを持ち、他方は ID 空間において遠くのノードのアドレスを持つものである。探索にはこの 2 つのアドレス表を用いる。アドレス表には他のノードの ID とアドレスに加え、そのノードのアドレス表のサイズを表すレベルという値を含む。Tourist のノードがアドレス表の管理のために割ける帯域幅は標準で最大帯域幅の 1% とされ、各ノードはその 1% 以下の帯域幅で扱えるだけの大きさのアドレス表を保持する。ネットワークの規模が小さい間は、個々のノードがアドレス表に他の全ノードのアドレスを保持できるため、全てのルーティングは 1 ホップで対象のノードへと辿り着く。ネットワークの規模が拡大し、ノードの参加や離脱の頻度が上がるにつれて、アドレス表の管理に使われる帯域幅が増えていく。管理に使われる帯域幅が上限を越えたノードは、自動的に自身のレベル

[1] 安定：ノードの参加と離脱の頻度が低い状態を指す

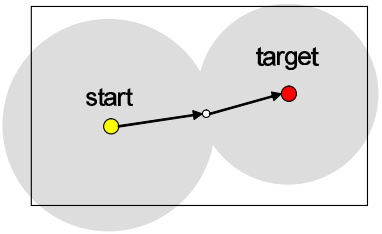

図 4: Kuriltai におけるルーティング

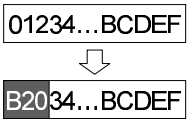

図 5: ノード ID

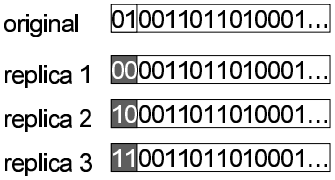

図 6: 複製のキー

を下げアドレス表を縮小する。レベルの低いノードは、全ノードのうち自身から離れたノードの一部と近いノードの一部だけをアドレス表に持つようになる。このネットワークの規模と安定度に応じたレベルの増減により、小規模で安定したネットワークにおいては 1 ホップでの探索を実現し、大規模で不安定なネットワークにおいてはマルチホップでの探索へと適応していく。

これにより、Tourist ではノード数 5,000,000 以下の安定したオーバレイネットワークにおいては 2 ホップという O(1) ホップによる探索を実現し、規模と安定度に応じて O(log$N$) ホップの探索へと適応する。

Tourist には残り 2 ホップで探索が目標へ到達することがわかった際、探索要求の転送先の複数の候補のうち、ルータレベルで最も近いノードを選ぶ仕組みがある。しかしながら、3 ホップ以上の探索については特に局所性は考慮されておらず、O(log$N$) ホップの探索の効率に問題がある。

Kuriltai は、Tourist を拡張し局所性を考慮した ID 空間を用い、更にこの ID 空間を分割しそれぞれに複製を配置することでデータの可用性と探索速度の向上を目指したものである。

3 局所性を考慮した ID 空間

前章で述べた局所性を考慮した 2 つの方法のうち、探索要求の転送先の候補の中からルータレベルで近いものを選ぶ手法では、転送元と転送先との距離が近いことのみを保証している。これは候補の数が多い際には効率の良いルーティングを可能とするものの、候補数の減少に伴って効果が半減する。また、3 ホップ以上の探索においては、経由するノードの ID を目標のキーに近づけることがルータレベルで近いノードへと転送することよりも優先されるため、効果が得にくい。探索要求を発行したノードの近傍のノードへとメッセージを転送していった結果、最終的な目標のノードからは徐々に遠ざかってしまう可能性もある。

一方、局所性を考慮した ID 空間を用いる手法は、探索要求の転送先と目標のキーを持つノードとの距離が近いことを保証している。この手法では 3 ホップ以上の探索においても効果を発揮するものの、ID 空間の均一性とのバランスを取るために粒度の粗い距離情報を用いざるを得ない。

そこで本研究で提案する Kuriltai では、上記の 2 つの手法を併用することで、転送元と最終的な目標との双方に近いノードを転送先として選択する (図 4)。これにより、局所性と ID 空間の均一性とのバランスを取りながら遅延を最小に抑えるルーティングを行う。

3.1 Kuriltai における ID 空間

Kuriltai では Zhou ら [13] の提案する手法により局所性を考慮した ID 空間を構築する。

Kuriltai では Tourist と同様、ノードの ID 及びデータのキーに 128 ビットの値を用いる。各ノードに割り当てられる ID は、その IP アドレスをハッシュ関数[2] にかけた返り値のうち、上位数ビットをルータレベルの局所性に基づく位置を表す値で置き換えたものとなる (図 5)。本稿ではこのルータレベルでの位置を表す値を領域 ID と呼ぶ。領域 ID を求める方法は幾つか考えられるが、ここでは最も単純な Ratnasamy ら [11] の手法を用いる。[13] では GNP[18] によってノードのネットワーク上での座標を求め、その座標を基に領域 ID を得る方法を取っている。更に ID 空間の不均一を回避する手法も述べられているが、全ノードに渡って役割が平等である DHT において、領域内に含まれるノード数の計測や領域の調整等の操作をどのノードが行うかについては言及されていない。

ノードは Kuriltai への参加時に複数のランドマークとの間の遅延を計測し、[11] によってネットワーク上の位置を表す識別子を取得する。ここで余り粒度の細かい位置情報を ID に適用すると ID 空間の均一性が大きく崩れてしまうため、識別

[2] ここでは SHA-1

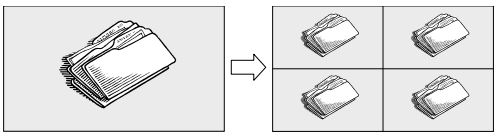

図 7: ID 空間の分割

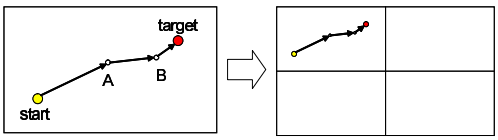

図 8: ID 空間分割時の探索

子の上位の数文字のみを数字列に変換し、ノード ID の上位数ビットと置き換える。これにより、ID 空間上で近いノードはルータレベルにおいても近くに位置するという特性が得られる。

以下、Kuriltai における局所性を考慮した ID 空間による探索の手順について述べる。あるノードが探索要求を受け取った時の処理の流れを次に示す。

1) 目標のキーを管理しているノードが自分かどうかを判断する。もしそうであれば、探索要求を発行したノードに結果を返す。

2) 2 つのアドレス表のうち近くのノードを集めた表から、目標のキーを管理するノードがいるか判断する。もしあれば、そのノードに探索要求を転送する。

3) 遠くのノードを集めたアドレス表の中に、目標のキーを管理するノードをそのアドレス表に含むノードがいるか判断する[3] 。もしあれば、そのノードに探索要求を転送する。もし条件に該当するノードが複数あれば、それらの中から現在のノードからの距離が近く、また目標のキーからも近くに位置するノードを選択する。

4) 全アドレス表の中から、目標のキーに最も近い値の ID を持つノードを選択し、これに探索要求を転送する。

手順 3) では、局所性を考慮した ID 空間と局所性を考慮した候補選択の 2 つの手法を併用しており、無駄の無い探索を実現している。また、手順 4) では従来の探索手法と同様に経由するノードの ID を目標のキーへと近付けているだけだが、局所性を考慮した ID 空間を利用しているためにルータレベルにおいても距離が接近し、効率の良いルーティングが行われる。

4 ID 空間の分割

DHT ではデータの可用性を高めるために、複製を配置することで冗長性を持たせることが多い。以下、ただ複製と記述する場合はデータのオリジナルとレプリカの両方を指し、複製の数とはオリジナルとレプリカの数の合計とする。一般に、レプリカは ID 空間上でオリジナルの隣に配置されることが多い。そのため、オリジナルの所在を知っていればレプリカの位置を容易に特定できるため、オリジナルが消失していた場合もすぐにレプリカを取得することができる。

しかし、このレプリカの配置方法を局所性を考慮した ID 空間を持つ DHT に適用する場合、問題が発生する。ID 空間で隣り合うノードは、局所性を考慮した ID 空間においてはルータレベルにおいても近隣に位置する。そのため、ネットワークの分断が発生した際に、オリジナルのみならずレプリカにも同時に到達不可能になってしまう可能性が考えられる。これはデータ可用性を高める目標とは矛盾しているため、Kuriltai では ID 空間において遠距離にあるノードにレプリカを保持させる。即ち、複製の数を 2^n とした時、オリジナルのデータキーの上位 n ビットが異なるキーにそれぞれレプリカを配置する (図 6)。これにより、目標のデータが失われていた際にはキーの上位 n ビットを異なる値に置き換え探索を継続することで他の複製から結果を得ることができる。

このレプリカの配置を全てのデータについて行うと、上位 n ビットの値によって ID 空間を分割した各部分空間ごとに、全てのデータの複製が含まれることになる。Kuriltai ではこの特性を利用し、ID 空間を 2^n 個の部分 ID 空間に分割する (図 7)。領域 ID の上位 n ビットを分割の基準としているため、各部分 ID 空間にはルータレベルにおいて近いノード同士が集まる。そのため全ての探索要求の目標を、探索を発行するノードと同一の部分 ID 空間内の複製とすることで、全ての探索は各部分 ID 空間内で完結するようになる (図 8)。各ノードは自身の属する部分 ID 空間内のノード以外と通信する必要が殆ど無くなるため[4] 、アドレス表には同一部分 ID 空間内のノードのみを持つこととなる。

以下は、この ID 空間の分割と各部分 ID 空間への複製の配置によって得られる特性である。

- 全てのルーティングはルータレベルにおいて狭

[3] これはアドレス表のレベルの値から判断できる

[4] 複製の探索やデータ更新時の一貫性の維持のため、平均 2^{n+1} 個の他の部分 ID 空間のノードのアドレスは保持する

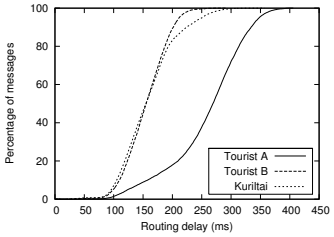

図 9: ($r = 1$) における遅延の分布

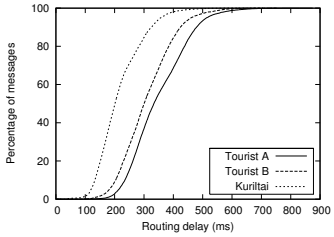

図 10: ($r = 0.01$) における遅延の分布

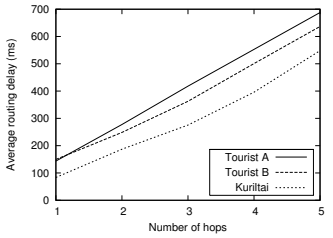

図 11: ホップ数ごとの平均遅延

い地域で完結するため、1 ホップごとの遅延の最大値が小さくなる。

- アドレス表に含むノードの範囲が ID 空間全体から部分 ID 空間に絞られたことから、およそ 2^n 倍の同一部分 ID 空間のノードのアドレスを持てるようになる。これに伴い、ルーティングのホップ数が減少する。
- ノードの状態変化を通知するマルチキャストの対象も同一部分 ID 空間内に限られるために、状態変化の通知が高速化される。
- 探索要求を発行したノードも目標のキーを持つノードも同一部分 ID 空間内に存在するため、探索の完了後に結果を返す際の遅延も縮小される。

以上のように、Kuriltai では ID 空間を局所性に沿って分割することで、高いデータの可用性を得るだけでなく、同時に探索等の効率も更に向上させている。

5 評価

Kuriltai と 2 種類の Tourist の探索速度についてのシミュレーションを行い、それぞれの結果を比較し考察を行った。

シミュレーションではインターネットのモデルとして Transit-Stub[19] 型のトポロジを用いた。設定は以下の通り。120 の Transit ドメインのそれぞれが 4 つの Transit ノードを含み、各 Transit ノードは 5 つの Stub ドメインを持つ。各 Stub ドメインは 4 つの Stub ノードを含む。Transit 間の遅延には 50 から 150 ms のランダムな値を用い、Transit-Stub 間の遅延には 10 から 30 ms のランダムな値を用いた。Kuriltai 及び Tourist の全てのノードはこのいずれかの Stub ノードに接続される。

各ノードの平均生存時間は Gnutella の測定結果 [20] に基づき 2.3 時間とし、r をその比として定義した。また、ノードの帯域幅の分布についても Gnutella の測定結果より取得した。

ノード数は 1,000,000 とし、生存時間の比 r を変化させることで探索のホップ数を増やし、それにより生じる変化を観測した。

領域 ID の取得には 8 つのランドマークを使用した。それぞれのランドマークとの間の遅延をもとに各ノードの属する領域の 11 ビットの領域 ID を取得し、ノード ID の上位 11 ビットをこれで置き換えた。

シミュレーションの結果は、トポロジを 10 度構築し直して計測した総計である。トポロジごとに任意のノードから任意のキーへの探索要求を 1,000 回発行し、これを記録した。

探索に用いた手法は以下の 3 つ。“Tourist A” は探索時に局所性を全く考慮せずに候補の中から探索要求の転送先を選択する Tourist である。“Tourist B” は標準通りに動作した時の Tourist である。探索要求の転送先の候補のうち、転送元に最も近いノードを選択する。“Kuriltai” は Tourist のランダムな ID 空間の代わりに局所性を考慮した ID 空間を使用している。また、探索要求の転送先の候補のうち、転送元と目的のキーの双方から近い候補を転送先として選択する。

5.1 局所性を考慮した探索

図 9 は 1,000,000 のノードの平均生存時間を 2.3 時間 ($r = 1$) とした場合の、探索にかかった遅延時間の累積分布関数である。ノード数が 1,000,000 であっても、Kuriltai と Tourist では全てのルーティングが 2 ホップ以下で行われる。探索要求の転送先の候補が大量にあることから、Tourist B はその転送候補の選択手法が最大限の効果を発揮しており、非常に効率の良い経路を選択できている。そのため、Kuriltai の結果には Tourist B との差が殆ど見られない。

続いて、3 ホップ以上のルーティングや転送先の候補数が少ない場合の探索時間を見るため、ノードの平均生存時間を約 83 秒 ($r = 0.01$) という極端に短い値に設定した上でシミュレーションを行っ

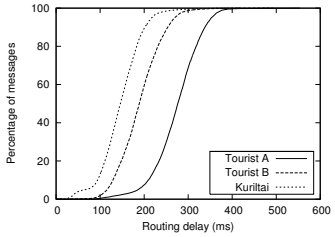

図 12: ID 空間 8 分割時の遅延の分布

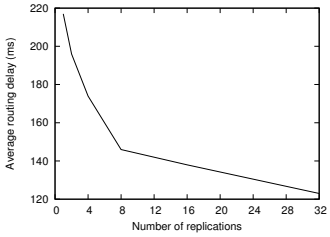

図 13: 分割数に対する平均遅延

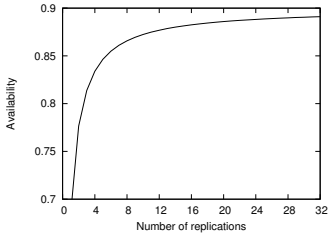

図 14: 分割数に対するデータ可用性

た (図 10)。$r = 0.01$ という条件下においては最大で 5 ホップまでのルーティングが発生しており、図からも Kuriltai による探索が高い効率を発揮していることが見て取れる。図 11 は探索にかかったホップ数と探索時間の関係を表したものだが、3 ホップ以上を要した探索のみならず、1, 2 ホップで完了した探索の時間においても Kuriltai が良い結果を出している。これは、アドレス表のサイズが縮小したことにより 2 ホップで目標へと到達可能である場合にも、候補の数が少ないために出てきた結果だと考えられる。Tourist B では転送先の候補が少ないために効率の良い経路を選択することができず、Tourist A の探索時間と大差の無い結果になってしまっている。対して、Kuriltai のノードは少なくとも目標のキーに近い ID を持つノードにさえ転送すれば一定の効果が得られるため、転送先の候補数の少ない 1, 2 ホップのルーティングにおいても高い効率を示している。

5.2 データ複製時の探索と可用性

次に、ID 空間を分割した際の探索時間とデータの可用性について述べる。平均生存時間の比を $r = 0.01$ とし、ID 空間を 8 分割した際の探索時間の累積分布関数を図 12 に示す。ID 空間を 8 分割することで、各ノードはアドレス表で自身の属する部分 ID 空間のノードのみを扱えるようになった。そのため、分割前からは一転して殆どの探索が 2 ホップで行われるようになっている。転送先の候補が多量に存在することから Tourist B の候補選択が効果を発揮しており、探索時間が再び小さくなっていることがわかる。にも関わらず Kuriltai による探索が最も短い時間で完了しているのは、Kuriltai が局所性に沿って ID 空間を分割したことによると考えられる。

ここで、ID 空間を分割し各部分 ID 空間へデータの複製を配置することにより得られるデータの可用性 A は、以下の式で求められる。

$$A = 1 - (1 - a)^R$$

ここで R は ID 空間の分割数及び複製数を表し、a は複製が 1 つの時点での可用性を表す。a は次の式で求められる。

$$a = \frac{MTTF}{MTTF + MTTR}$$

$MTTF$ は平均故障時間 (Mean Time to Failure) であり、データを保持するノードが離脱するまでの平均時間である。これはノードの平均生存時間に対応する。$MTTR$ は平均修復時間 (Mean Time to Recover) であり、消失したデータが新たにそのデータを担当することになったノードに受け継がれるまでの時間である。これはノードの離脱を通知するマルチキャストにかかる時間に、データのコピーにかかる時間を加えたものに等しい。各ノードの保有するデータの数は複製の数に比例するため、データのコピーにかかる時間も複製数に比例して大きくなる。

ここで d を、複製数 1 の時にノード 1 つの持つ全てのデータをコピーするのにかかる平均時間 (秒) とする。図 14 は $r = 0.1$, $d = 360$ という条件のもとで、複製数の変化に伴うデータの可用性を表したものである。ノードの平均生存時間は約 14 分と極めて短く、一方の新規ノードにその管理するデータをコピーするのにかかる時間は 5 分と長くなっている。ここでは図の見やすさから上記のような値を用いたが、複製数に対する可用性の変化は他の値を用いた場合にも同様である。

図 13, 14 から、Kuriltai において複製による探索速度の向上と高い可用性を得ようという場合は、8 前後の複製数が妥当であると考えられる。

尚、平均生存時間を 1 時間 ($r = 0.435$) と短めに設定し、データのコピーにかかる時間を 3 分 ($d = 180$) とした場合でも、複製数 8 の際のデータの可用性は 0.99995 と非常に高い値を示している。

6 おわりに

本稿では局所性を考慮した ID 空間を持ち、高速な探索と高いデータの可用性を実現する DHT,

Kuriltai を提案した。シミュレーションにより、ルーティングのホップ数が増加するノードの平均生存時間の短い状況においても、探索の速度とデータの可用性の両面において高い性能を発揮することを確認した。

今回の実験ではノード数を 1,000,000 より増やした状態ではメモリが足りずにシミュレーションを実行することができなかった。今後は今回確認できなかった、よりノード数を増やした状態でのシミュレーションを行い、そのスケーラビリティについても評価していきたい。また、ID 空間の構築時の局所性に関する情報の取得に、GNP[18] や Vivaldi[21] 等の異なる手法を用いた際の結果についても評価を行い、比較・検討したい。

参考文献

[1] S. Ratnasamy, P. Francis, M. Handley, and R. Karp, "A Scalable Content-Addressable Network," *Proc. ACM SIGCOMM 2001*, 2001.

[2] I. Stoica, R. Morris, D. Liben-Nowell, D.R. Karger, M.F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications," *IEEE/ACM Trans. Networking*, vol. 11, no. 1, pp. 17–32, 2003.

[3] A. Rowstron and P. Druschel, "Pastry: Scalable, Distributed Object Location and Routing for Large-scale Peer-to-peer Systems," *Proc. IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001)*, 2001.

[4] P. Maymounkov and D. Mazieres, "Kademlia: A Peer-to-peer Information System Based on the XOR Metric," *Proc. 1st International Workshop on Peer-To-Peer Systems (IPTPS '02)*, 2002.

[5] A. Muthitacharoen, R. Morris, T. M. Gil, and B. Chen, "Ivy: A Read/Write Peer-to-Peer File System," *Proc. 5th Symposium on Operating Systems Design and Implementation (OSDI '02)*, 2002.

[6] M. J. Freedman, E. Freudenthal, and D. Mazieres, "Democratizing Content Publication with Coral," *Proc. 1st USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI '04)*, 2004.

[7] A. Gupta, B. Liskov, and R. Rodrigues, "One Hop Lookups for Peer-to-Peer Overlays," *Proc. 9th Workshop on Hot Topics in Operating Systems (HotOS IX)*, 2003.

[8] I. Gupta, K. Birman, P. Linga, A. Demers, and R. van Renesse, "Kelips: Building an Efficient and Stable P2P DHT Through Increased Memory and Background Overhead," *Proc. 2nd International Workshop on Peer-To-Peer Systems (IPTPS '03)*. 2003.

[9] J. Hu and M. Li, "Tourist: Self-Adaptive Structured Overlay Network," in Submission. `http://hpc.cs.tsinghua.edu.cn/granary/`

[10] I. Abraham, A. Badola, D. Bickson, D. Malkhi, S. Maloo, and S. Ron, "Practical Locality-Awareness for Large Scale Information Sharing," *Proc. 4th International Workshop on Peer-To-Peer Systems (IPTPS '05)*. 2005.

[11] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Topologically-Aware Overlay Construction and Server Selection," *Proc. IEEE INFOCOM 2002*, 2002.

[12] L. Garces-Erice, K. W. Ross, E. W. Biersack, P. A. Felber, and G. Urvoy-Keller, "Topology-Centric Look-Up Service," *Proc. 5th International Workshop on Networked Group Communications (NGC '03)*, 2003.

[13] S. Zhou, G. R. Ganger, and P. Steenkiste, "Balancing Locality and Randomness in DHTs," Technical report CMU-CS-03-203, CMU-CS, 2003.

[14] P. Knezevic, A. Wombacher, and T. Risse, "Enabling High Data Availability in a DHT," *Proc. 2nd International Workshop on Grid and Peer-to-peer Computing Impacts on Large Scale Heterogeneous Distributed Database Systems (GLOBE '05)*, 2005.

[15] R. Rodrigues and B. Liskov, "High Availability in DHTs: Erasure Coding vs. Replication," *Proc. 4th International Workshop on Peer-To-Peer Systems (IPTPS '05)*. 2005.

[16] J. Hu, H. Dong, W. Zheng, D. Wang, and M. Li, "Twins: 2-hop Structured Overlay with High Scalability," *Proc. International Conference on Computer Science (ICCS '04)*, 2004.

[17] J. Hu, M. Li, W. Zheng, D. Wang, N. Ning, and H. Dong, "SmartBoa: Constructing p2p Overlay Network in the Heterogeneous Internet Using Irregular Routing Tables," *Proc. 3rd International Workshop on Peer-To-Peer Systems (IPTPS '04)*. 2004.

[18] T. S. E. Ng and H. Zhang, "Predicting Internet Network Distance with Coordinates-Based Approaches," *Proc. IEEE INFOCOM 2002*, 2002.

[19] E. W. Zegura, K. Calvert, and S. Bhattacharjee, "How to Model an Internetwork," *Proc. IEEE INFOCOM 1996*, 1996.

[20] S. Saroiu, P. K. Gummadi, and S. D. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems," *Proc. Multimedia Computing and Networking 2002 (MMCN '02)*, 2002.

[21] F. Dabek, R. Cox, F. Kaahoek, R. Morris, "Vivaldi: A Decentralized Network Coordinate System," *Proc. ACM SIGCOMM 2004*, 2004.