File: wide-draft-ideon-ydoi-dht-dns-00.pdf
Title: DHT を用いた DNS 上の非構造な名前空間の検索効率化
Author: 土井 裕介, ydoi@isl.rdc.toshiba.co.jp
Date: 2005-01-27

1 はじめに

1.1 ID を中心とした情報管理における課題

ユビキタスネットワーク、つまりどこにいてもコンピュータネットワークが利用可能な状態において有用と考えられる応用の一つに、コンピュータによる状態認識に関連する応用がある。このような応用の具体例には、物流の各段階おいてその場での商品の状態や流通に関する情報を記録し、消費者や生産者からの追跡要求に対して情報を開示するサービス (トレーサビリティ) の実現等がある。

トレーサビリティシステムでは、例えば図 1 のようなシステムが考えられる。生産された商品は、流通業者・加工業者・小売業者を経て最終的に消費者のもとに届けられる。消費者は商品に対して与えられた付加価値としてトレーサビリティシステムから様々な情報を取得可能であり、この商品がどのように生産され、加工され、輸送されたかを知る。なお、このような応用については文献 [9] にも詳しい。

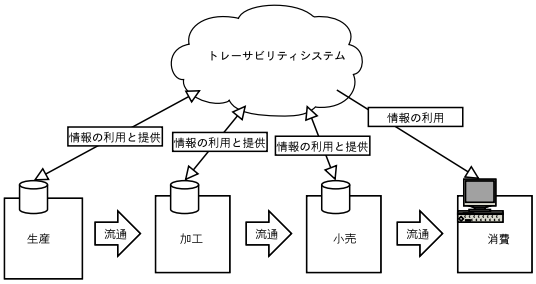

図 1: トレーサビリティシステムのイメージ

トレーサビリティシステムにおいて重視すべきは、スケーラビリティである。例えば、書籍の 2001 年の国内の出荷量は約 42 億冊 (全国出版協会・出版科学研究所調べ) である。これらは再販制度や中古品市場があるため、ID ひとつ当りの問い合わせ要求は消費して終わる食品等よりも数倍多くなると予想できる。

その上、トレーサビリティシステムでは、多量の ID に関連する情報の取り扱いができるだけでなく、柔軟な資源追加・管理ができなければならない。

例えば、ある商品がヒット商品になると、事前の予測を超える規模の検索要求が発生する。このような場合でも計算機資源の柔軟な追加等により対応できなければならない。

トレーサビリティシステムを実際の利用者に結びつけるためには、商品等に対応付けられる「モノの ID」をどのように管理するかが問題となる。特に、膨大な ID が存在する空間の中で、分散して配置されたデータベースのうちどのデータベースがその ID に関する情報を保持しているかを発見する方法が技術的課題となる。本研究では、特に ID を検索の鍵として、ID に関連する分散した情報源へのインデックス情報を取得する技術を総称して、分散インデックス技術と呼ぶ。

なお、このような技術領域をより一般的に捉えると地理的および管理ドメイン的に分散したデータベースに対して、ある ID に関連する情報がイベント駆動や受動的観測などにより蓄積されていく、というモデルとなる。この ID は世の中全ての事物に関連づけられる位に巨大な空間を持ち、また分散データベースの数は、ID の空間に比べれば小さいものの、世の中の観測・管理の主体となる存在 (会社・ビル・行政地区等) の数に比例するため、簡単に全数を探索できるような数ではないだろう。このようなモデルはユビキタスネットワーキング応用の一般的なモデルの一つとして考えてもよい。

1.2 分散インデックス技術の実現

分散インデックス技術に求められる機能は、優れたスケーラビリティと柔軟な資源管理である事は節 1.1 で述べた。

ここでは、既存の名前管理システムである DNS を分散インデックス技術に利用する場合を考える。DNS は、木構造を持つ名前空間の管理に特化しており、物品のシリアル番号等のように構造を持たない名前空間の管理には向かない。理由は 2 つあり、第一に、構造を持たない名前空間の上位の空間の名前サーバに問い合わせが集中する点がある。第二に、集中を緩和するためのサーバの分散方式の設計と名前空間の結節点 (ゾーンカット) の設計が密接に関係するため、動的な負荷に対応するための負荷分散方式の設計変更が困難である点である。

一方、分散インデックス技術が持つ要求に対して有効な技術の一つに、Stoica らによる Chord[7] といった、分散ハッシュテーブル (DHT) と呼ばれる一群の方式がある。ハッシュテーブルのような、一連の $key \rightarrow value$ の対応関係を登録・検索するシステムを、完全に自律分散したノード群によって実現するための方式である。 DHT では、分散した DHT ノード全てが、同一のハッシュ関数を利用してメッセージを経路付けし、ハッシュ値の空間を分散分割統治する。

DHT では、名前空間の内容は全ての DHT サーバにほぼ一様に分布する。

また、サーバの追加および削除への対処は自律的に行われるため、非構造な名前空間を柔軟に管理するという分散インデックス技術の要求を満たすことができる。

同時に、DHT はスケーラビリティ特性に優れる。全ノード数を N とした時の問い合わせの通信量は $O(log_b N)$ である (b は方式によって決定される)。ノードを追加し、N を増大させることでシステム全体の容量 (全体で利用可能な通信帯域・メモリ領域) が増加し、一方で通信量はそれほど増加しないことから、ノードの追加により性能が相対的に向上するものと考えられる。

しかし、DHT には二つの問題が存在する。第一に、DHT は既存の DNS を前提として発展してきた基盤とは異なる方式であり、クライアント側の対応が必要になる。第二に、DHT の名前空間は一様であることが前提であり、DNS と異なり、空間内の特定の部分を特別に扱うような処理に向かない。後者により、例えばコストをかけて自社に関連する名前空間の問い合わせを向上させる (サービスの差別化) といったことは DHT では実現できない。

なお、本研究では、DHT に関して、Chord を前提としている。しかし、著者は本論文における貢献は DHT として総称される技術全般にあてはまると考える。

1.3 本論文の貢献

分散インデックス技術を実現する既存の技術としては、EPCglobal Inc. が提唱している ONS[1] のように、DNS を利用してデータベースと ID との対応づけを行っている方式が存在する。しかし、節 1.2 で説明した通り、DNS を利用した分散インデックス技術にはスケーラビリティの問題と柔軟性の問題がある。 一方で、DNS は結節点を境界にして名前空間に固有の性質を与えられる (名前空間の多様性の許容) という利点がある。

一方、分散インデックス技術に DHT を用いる場合を考える。特にトレーサビリティシステムを考えると消費者 (一般家庭) がそのシステムの末端となる。分散インデックス技術の実現のために全てのクライアントの変更を前提とする DHT は、導入に高いハードルがあり、ONS では既存の基盤である DNS が最初の分散インデックス技術として提案されている。また、DHT は名前空間の性質は一様であることを前提としており、DNS の代替のような機能には適さない。

本論文ではこれらの背景を踏まえ、DNS の名前空間木の一つのゾーン (DNS 名前空間のサブセットの単位) として DHT 名前空間を導入し、ID を登録・検索する名前空間にのみ DHT が持つ特性を与える、という方式 (名前空間マウント方式) を提案する。同時に、提案方式が持つ問題点を明らかにした上で、シミュレーションによりその問題が発生する条件を絞り込む。

本論文は次のように構成する。第 2 章において、関連研究について述べる。

第 3 章では、DNS から DHT をアクセスする場合の課題について述べる。課題を解決する手法と、名前解決が失敗する場合について、第 4 章で述べた上で、特に名前解決の失敗に関して第 5 章でシミュレーションを行う。第 6 章で本研究の評価を行った上で、 本研究で触れられなかった話題について第 7 章で述べる。最後に、 第 8 章にて本研究の結論と今後の課題について説明する。

2　関連研究

関連研究として、Cox らによる、DNS の機能を Chord により代替させる研究 [2] が存在する。Cox らは、Chord の全域を用いて DNS の代替機能を提供するシステムを実装し評価した。このシステムは DDNS と呼ばれる。評価の結果、DDNS は次のような特性を持つとされる。

まず、DDNS では、Chord の持つ自己組織化機能により、名前サーバの管理が自動化され、また同時に良好な負荷分散が行われる。しかし、問い合わせに必要なメッセージの数が多いために遅延が全体的に多くなる。

特に遅延に関しては、DNS と DHT が持つ遅延の特性の違いがある。DHT では、自動的に構成されるネットワークは比較的整った構造になる。その結果、ほとんどの問い合わせの遅延はある一定の範囲に収まり、上限を大きく上まわる可能性が少ないと同時に、下限を大きく下まわる可能性も少ない。一方で、DNS では名前空間木の積極的な枝分かれにより、問い合わせの回数が比較的少数に抑えられる上、キャッシュの効果により繰り返される多くの問い合わせが短時間で解決される。一方、レスポンスが悪いサーバや、設定が誤っているサーバにも問い合わせを行うので、解決できない時はクライアントがタイムアウトするまで待ち時間が発生する。

また、DDNS 単体では、ホスト名とアドレスの分散した対応表以上の機能は果たせない。DNS はゾーン毎にサーバを分割可能なため、ゾーンに特有な機能、例えば負荷分散機能等をサーバ側にて実現できる。一方で、DHT では空間それぞれとサーバの関係のハッシュ関数による切断によりスケーラビリティと自律性を実現しており、DNS のような名前空間毎に異なる機能を実装するためにはクライアント側へ実装が必要になり、非現実的である。

本研究ではこの研究成果を踏まえて、DHT が持つ優れたスケール特性及び柔軟性を、DNS が持つゾーン固有の特性として実現する方法を提案する。

3　DNS と DHT の名前空間接合における課題

第 1 章で述べたように、DNS の不均質性を活かして DNS の一部ゾーンの管理に DHT を用いることは、特に既存のアプリケーション基盤に対してシリアル番号のような非構造かつ膨大なスケールの名前空間を導入する際に合理的である。そのためには、DHT と DNS の名前空間を接続する部位 (名前空

間接合部) において、DHT が持つスケーラビリティを損わないように注意する必要がある。

名前空間マウント方式には、2 つの課題が考えられる。第一に、第 1 章で述べた導入への障壁である。いずれかのサーバでプロトコルを変換し、DNS 名前空間の一部として DHT 名前空間を導入し、導入障壁を下げる必要がある。

もう一つの課題は、第一の課題に従属する。名前空間接合部では DNS と DHT のプロトコルを相互に変換する。DNS と DHT はそれぞれスケーラビリティに優れたシステムであるが、名前空間接合部がボトルネックとなるようでは DHT のスケーラビリティを活用できない。

例えば DNS と DHT の接合の平易な実現方法として、あるゾーンを管理する名前サーバがプロトコル変換を行い、DHT の名前空間を検索する、という方法が考えられる。この方法は、DNS が持つ名前権限委譲 (name delegation) 機能を利用したプロトコル変換ゲートウェイ方式である。しかし、他のプロトコル変換ゲートウェイ方式と同様、ゲートウェイ自体がボトルネックとなり、DHT のスケーラビリティを損なう可能性が高い。

図 2 は、この方法におけるボトルネックの存在を示している。図の左側は名前空間に対応し、右側は実際の通信の模式図である。既存のリゾルバと DHT ネットワークは量的に大きい。ゲートウェイを設置し、DHT の名前空間をその直下に位置させることによってリゾルバからの問い合わせをゲートウェイに誘導したとして、量的に大きい 2 者の間を有限 (図では一つ) のゲートウェイが仲介することになり、ここが深刻なボトルネックとなる。

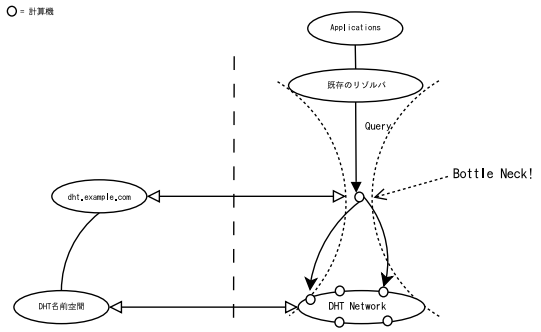

図 2: ボトルネックの存在 (左側は名前空間、右側は問い合わせの流れを示す)

4　名前空間マウント方式

本章にて、DHT に対する DNS からのアクセス手法を論じる。説明のために DNS の構成要素を整理した上で、提案方式及び具体例を述べる。

4.1 DNS の構成

DNS には、名前情報の提供および解決という対になる二つの機能がある。実際には、これら二つの機能は一つのプロセスで提供されている場合が多い。しかし本稿では、名前情報の保持という機能 (DNS-NS: 名前サーバ機能) と名前情報の解決という機能 (DNS-RS: リゾルバサーバ機能) は独立の構成要素として扱う。なお、アプリケーションは名前解決サービスが必要な場合、スタブリゾルバと呼ばれる専用のライブラリを利用し、一つないし複数の DNS-RS に再帰的名前解決を依頼する。

DNS では、名前空間の接合には名前権限委譲 (delegation) という機構が用いられる。名前権限委譲においては、上位のゾーンにおいて、直に接続する下位のゾーンの DNS-NS の名前を NS リソースレコード (以下 RR) という形式に記述することにより、下位のゾーンに対する権限を該当する DNS-NS に委譲する。必要な場合は IP アドレス (Glue A RR) を同時に記述する。DNS-RS は、この情報を頼りにして、木構造の名前空間を上層から下層に再帰的に探索する。

なお、DNS の詳細な動作は、RFC1034[6] 等によって定義されている。

4.2 2 つのアプローチの併用による構成

通信方式の異なる二つの方式を接合する以上、何らかのゲートウェイの設置は避けられない。この際、ゲートウェイに対応する部分からボトルネックとなりうる要因をできるだけ排除する必要がある。本研究では、ボトルネックを「**高負荷**」が「**限られた装置に集中**」する現象と捉え、これらをそれぞれに排除する 2 つのアプローチを採用する。

1. 負荷削減アプローチ: 名前空間接合部を構成する装置単体での負荷をできるだけ下げる
2. 分散化アプローチ: DHT とほぼ同等のスケーラビリティを持たせる

提案方式はこれら 2 つのアプローチを別個に追求するために、DNS と DHT の名前空間接合部において 2 段階のゾーンを設ける。本節では、この 2 段階のゾーンを利用して DNS と DHT のスケーラビリティを損なわずに両者を結合する手法について述べる。

図 3 に提案する名前空間マウント方式を示す。図中左側が DNS と DHT の名前空間を示しており、上層には DNS の名前空間が広がっている。DNS の名前空間は、ゲートウェイゾーン・トランスレートゾーンからなる名前空間接合部を経由して DHT 名前空間に接合される。図中右側には、実際のサーバ等の情報取得の関係が示されている。本提案では、1 段目のゾーン (ゲートウェイゾーン) において負荷削減アプローチを実現し、2 段目のゾーン (トラ

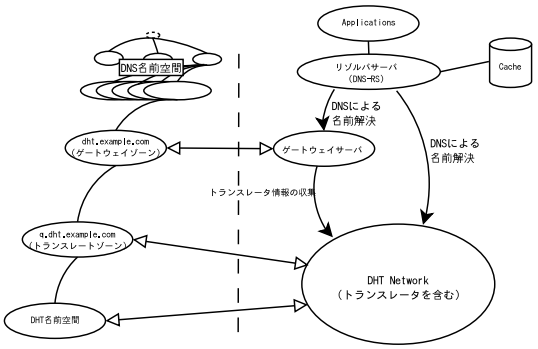

図 3: 2 層構造による名前空間マウント方式の実現 (左側は名前空間、右側は問い合わせの流れを示す)

ンスレートゾーン) において分散化アプローチを実現させる。その結果、DNS から DHT への問い合わせにおいてそれぞれのアプローチにおいて最大限に最適化する余地が生まれ、ボトルネックを解消できる。

DNS の名前権限委譲においては通常、名前権限の委譲先は限られた静的なサーバ群であり、分散化アプローチの適用は難しい。そこで、負荷削減アプローチの適用を前提としたゾーンを設置する。このゾーンをゲートウェイゾーンと呼ぶ。ゲートウェイゾーンを構成するサーバの仕事は、次に述べるトランスレートゾーンを構成するサーバ群を調査し、これを DNS-RS に応答することである。

負荷削減アプローチを取るため、ゲートウェイゾーンでは資源を消費する複雑な処理は避ける。従って、実際の DHT の問い合わせは DNS-RS からそれぞれの DHT ノードに依頼させる必要がある。加えて、DNS-RS からの要求をそれぞれの DHT ノードに振り分けるために、名前権限委譲の機構を用いる。そのために、トランスレートゾーンと呼ぶゾーンを、ゲートウェイゾーンの直下に設ける。

DHT ノードはそれぞれトランスレートゾーンに対する名前サーバとして動作する。トランスレートゾーンにおける名前解決は、DNS による名前解決を受信し、DHT の名前解決に翻訳して行う。ここで、DHT 自身が持つ量的な拡張性を利用する結果、前述の分散化アプローチが達成できる。

4.3 動作の流れ

`example.com.` の DNS-NS は、`dht.example.com.` に対する静的な権限委譲を行っている。例えば 3 台の名前サーバが存在し、それらがゲートウェイサーバ (ゲートウェイゾーンの名前サーバ) となる。ゲートウェイサーバにより構成される名前空間 `dht.example.com.` がゲートウェイゾーンとして

機能する。

ゲートウェイサーバは、それぞれ各個に開始点 (種) となる DHT ノードを与えられ、DHT ネットワーク上の隣接ノードを順次探索して、DHT ノードの存在を自律的に学習し続ける。またこの際、各 DHT ノードと通信して、トランスレータとして動作可能かを問い合わせる。

トランスレータは DHT ノードのうち、ネットワーク資源やポート番号 (DNS は 53 番) が利用可能なものにおいて起動される。そして、トランスレートゾーンに対して権限があるかのように振舞い、DNS によって受信した問い合わせを DHT の問い合わせに変換し、DHT による解決結果を DNS による応答に合成して返す。

問い合わせ時の動作の流れを、図 4 に示す。この例では、RFID あるいはバーコードによって読み出された ID を元に、NAPTR RR[5] を読み出して、サービスを利用するまでの流れを示している。

1. リーダから ID が読み出される
2. アプリケーションが RR の問い合わせを発行する
3. リゾルバ (ライブラリ) は、DNS-RS へ問い合わせを転送する
4. 権限委譲あるいはキャッシュにより、DNS-RS の問い合わせはゲートウェイサーバへ転送される
5. いくつかの DHT ノード (トランスレータ) に権限委譲するメッセージが返される
6. DNS-RS はトランスレータへ問い合わせを転送する
7. トランスレータは受信した問い合わせを分析し、対応する情報を DHT 側に問い合わせる
8. DHT 側が応答する
9. トランスレータによって DNS の形式に変換された情報が、DNS-RS 経由でアプリケーションに返される
10. DHT に格納された情報の内容に基づき、アプリケーションはサービスを発見し、利用する

ここで、`12345678` というシリアル番号が RFID 等から読み出された (図 4 中矢印 1、以下同様) 場合、次のような NAPTR RR と合成し、`12345678.q.dht.example.com` というドメイン名を連鎖的に導出できる。アプリケーションは、リゾルバを用いてこのドメイン名からサービスへのポインタを得るための問い合わせを DNS-RS に送信する (矢印 2 および 3)。

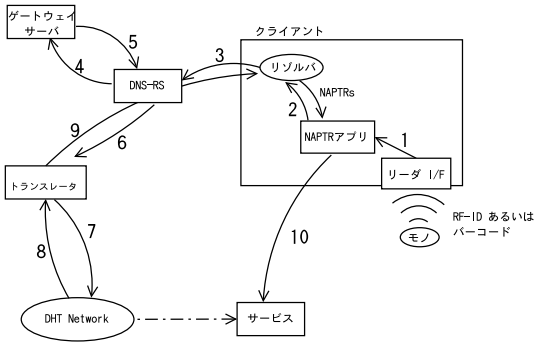

図 4: ゲートウェイゾーンを経由した問い合わせの流れの例

```
example.com. IN NAPTR 10 10 "" ""\
 "/(.*)/\1.q.dht.example.com./" .
```

(便宜上二行で記述したが、実際は 1 レコード)

`example.com.` からの名前権限委譲により、DNS-RS はゲートウェイサーバを `dht.example.com.`(トランスレートゾーン) に対する権限を持つ DNS-NS であると学習する。その結果、DNS-RS からの `12345678.q.dht.example.com.` への名前解決要求はゲートウェイサーバに送信される (矢印 4)。

ゲートウェイサーバは、DNS-RS から受信した名前解決要求への応答として、`q.dht.example.com.`(トランスレートゾーン) に対する NS RR を返答する (矢印 5)。ゲートウェイサーバはこの NS RR に、学習した DHT ノードの中からトランスレータとして動作可能なノードを格納する。例外的な場合を除いて、NS RR によって応答したトランスレータ情報は一度で廃棄する。

ゲートウェイサーバから権限を委譲されたトランスレータは、DNS-RS からの再帰問い合わせを受信する (矢印 6)。ここでトランスレータは、問い合わせを調べて DHT により解決する (矢印 7 および 8)。本研究ではトランスレータの動作方式については定義しないが、例えば、問い合わせドメイン名のうち、トランスレートゾーン以下を DHT における問い合わせの key とみなし、対応する値を取得する、などといった方式が考えられる。ただし、応答を合成するために必要なデータをどのような形式で格納するかは、DHT 利用者とトランスレータの間で合意しておく必要がある。

トランスレータによって合成された NAPTR RR の応答はアプリケーションに返され (矢印 9)、これを手がかりにアプリケーションはサービスを発見できる。

なお、同じ DNS-RS を利用した二度目以降の問い合わせは、DNS-RS に存在するキャッシュの効果により直接同じトランスレータに送られる。従って、ゲートウェイサーバが応答する RR の TTL によりキャッシュの効果時間が決定されるため、DNS-RS が再びゲートウェイサーバへ問い合わせを行うまで

の間隔は制御可能である。ゲートウェイサーバの負荷を下げるためには、TTL は長い方が良い。しかし、長い TTL は名前解決失敗の可能性を引き上げる (節 4.4 にて述べる)。

また、他の DNS-RS を利用した問い合わせは異なる DHT ノード上のトランスレータに送信される。これは、ゲートウェイサーバが問い合わせに対して応答するトランスレータを変化させ、分散させるからである。

4.4 キャッシュが原因の名前解決失敗

ゲートウェイサーバにより返されたトランスレータの情報を、DNS-RS はキャッシュする。TTL が長ければキャッシュが有効に作用し、結果的にゲートウェイサーバの負荷を下げられる。しかし、長すぎる TTL を持つキャッシュは、トランスレータの DHT からの離脱ないし故障が原因で名前解決が失敗する可能性を高める。

情報源 X に対するキャッシュC において、C の TTL 期間内で C が有効な間に、情報源 X が変化した結果 $X \neq C$ となり、一見有効な情報に見えるが実際には誤った情報を示すキャッシュを、本研究では劣化キャッシュ(stale cache) と呼び、キャッシュが劣化キャッシュとなることを劣化キャッシュ化と呼ぶ。劣化キャッシュに関する研究については、例えば文献 [8] を参照されたい。

特に本研究では、ゲートウェイサーバを情報源として DNS-RS へ通知されるキャッシュに着目する。この場合、ゲートウェイサーバにより通知されたトランスレータのキャッシュが劣化キャッシュ化する原因には、トランスレータの停止、故障、あるいはネットワークの不調による離脱等がある。

DHT では、アプリケーションが前提とする内容にもよるが、一般的に中央集権的な管理がなじまないアプリケーションにおいて多く導入されるものと考えられている。従って、名前サーバに比べて、DHT ノードは頻繁に DHT への参加・離脱を行うものと考えられる。そして、DHT ノードすなわちトランスレータの離脱は、DNS-RS に存在するそのトランスレータに対応する NS RR および対応する Glue A RR 等が劣化キャッシュ化するという結果を招く。

このような NS RR の劣化キャッシュ化は深刻であり、存在しないトランスレータへの問い合わせを継続するため、名前解決に失敗する。少なくとも ISC bind 9.2.3 で検証した限りでは劣化キャッシュ化したレコードは対応する DNS-NS に対するアクセスがタイムアウトしても有効であり続け、その結果、対応するゾーンへのアクセスが不能になる。

5 劣化キャッシュの影響評価

節 4.4 で述べたように、劣化キャッシュの影響により名前解決が失敗する可能性がある。ここでは、DHT ノードが離脱するまでの時間と、ゲートウェイ

サーバが返す NS RR の TTL の関係を軸として、シミュレーションによる劣化キャッシュの影響評価を行う。

5.1 検証の焦点

本研究では、特にノードの平均接続持続時間に対してゲートウェイサーバが DNS-RS に返す TTL をどの程度の長さにすると、無視できない程度の劣化キャッシュが発生するかを調査する。なお、本研究では、無視できない程度の劣化キャッシュの発生を、失敗した問い合わせの比率が 0.001 以上となる場合、と定義する。

シミュレーションの出力として R_f :失敗した問い合わせの比率を得る。名前解決の失敗は、問い合わせを行った際に DNS-RS が持つキャッシュの全てが劣化キャッシュだった場合に発生する。

5.2 シミュレーションモデル

シミュレータは次の要素より構成される。DHT の要素は、ネットワークに参加しているノードの集合と、ノード自身である。一つの DHT ネットワーク上にて、単位時間あたり一定数のノードが到着する。また、それぞれのノードは単位時間あたり一定の確率で離脱する。この結果、系全体のノード数はある程度の規模で安定する。また、DHT ネットワークに対して一つのゲートウェイサーバが存在し、DHT ネットワークから単位時間ごとに一つのノードを取得し、キューに保存する。

今回のシミュレーションによって明らかにする点は、ある DNS-RS における劣化キャッシュの作用である。従って、実際には DNS-RS は多数存在するが、ここでは代表して DNS-RS が 1 つ存在するモデルを作成し、着目する。

DNS-RS に対応するクライアントは 1 つ存在し、クライアントは単位時間毎に一定数の問い合わせを発行する。DNS-RS は問い合わせに際して、キャッシュ情報をチェックし、もしキャッシュに情報が存在する場合はそのキャッシュの内容に基づき、問い合わせが成功するか否かを判断する。一方で、キャッシュが存在しない場合はゲートウェイサーバへ問い合わせて、その結果をキャッシュに記入する。問い合わせの正否は、クライアントがログファイルに記録する。

また、単位時間毎にキャッシュの検査し、キャッシュ全体の中で劣化キャッシュがいくつ存在するかを記録する。

5.3 パラメータと手順

シミュレーションのパラメータは次の通り。

表 1: シミュレーションの条件

パラメータ	値
L	25000
N_a	5.0
R_d	0.004 ($\frac{1}{250}$)
M	$3, 5, 10$
T	$25 \ldots 500$
N_q	0.1

- L: シミュレーションの長さ (単位時間)
- N_a: 系全体に対する、DHT ノードの単位時間当りの到着数
- R_d: 単一 DHT ノードの、単位時間あたりの離脱確率
- M: ゲートウェイサーバが一度に返すノードの数、あるいは、トランスレートゾーンを提供するトランスレータの並列度
- T: ゲートウェイサーバが提供する NS RR の TTL
- N_q: 単一の DNS-RS に対してリゾルバが発行する名前解決問い合わせの、単位時間当りの到着数

シミュレーションの条件を表 1 に示す。ここで、各ノードの平均接続持続時間を 250 と定め、L は平均接続持続時間の 100 世代分とした。また、R_d は平均接続持続時間の逆数として、DHT の参加ノード数が 1000 ノード程度で安定するよう N_a を定めた。M は、UDP に格納可能な上限と比較的近いと思われる値 ($M = 10$) と、最低限 DHT ノードと 2 隣接ノード ($M = 3$)、および中間的な値 ($M = 5$) を代表値として選んだ。T は等比数列に近くなるような N_q は単一の DNS-RS に対する問い合わせの到着頻度であり、今回のシナリオでは DHT ノードの取得速度に比べて十分に遅い 0.1 とした。

シミュレーションの手順は次の通り。

1. 初期状態 (ノードが存在しない状態) の生成
2. ノード数が安定するのに十分な時間、ノードの到着・離脱を継続
3. 安定した時点で DNS-RS およびクライアントを生成し、問い合わせ成功率等を測定開始
4. 手順 3 より L 単位時間経過時点で終了

5.4　シミュレーション結果

以上の条件で行ったシミュレーション結果を図 5 に示す。縦軸が全問い合わせの試行うち、劣化キャッシュが原因の失敗した問い合わせの比率、横軸がゲートウェイサーバが返す T の値である。

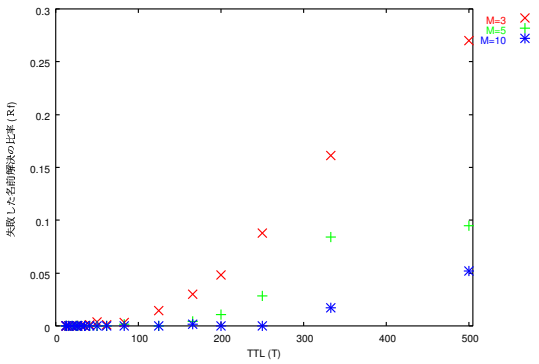

図 5: 失敗した問い合わせの比率 (R_f) と T の関係 (抜粋)

6　考察

6.1　本方式の効果と限界

第 3 章では、名前空間マウント方式の課題として既存のクライアントとの適合性と、名前空間接合部におけるスケーラビリティの確保を挙げた。

本方式による効果は、ゲートウェイゾーンとトランスレートゾーンの分割により、2 つの異なるアプローチにより名前空間接合部のボトルネックを軽減する点にある。極論すると、ゲートウェイサーバが返す NS RR の TTL が無限大であれば、世界中の DNS-RS はゲートウェイサーバに一回しか問い合わせを行わず、DNS-RS とトランスレータは分散した対応関係を作り、ゲートウェイゾーンによるボトルネックは存在しなくなる。

トランスレータは必要に応じて追加可能であり、世界中の DNS-RS からの問い合わせ量が増大した場合でも処理を継続できると考えられる。

従って、本方式により、DHT のスケーラビリティを損なうことなく DNS から DHT に対する問い合わせを解決でき、その結果既存の情報基盤に対して透過的に DHT を導入できた。これにより例えば、トレーサビリティのようなアプリケーションが DHT の特性を必要とし、かつ、専用アプリケーションの導入が困難である、という場合に、問題を解決できる。

一方で、本方式における明らかな性能の限界を決定する要素は、ゲートウェイサーバが利用可能な通信帯域であることがわかった。ゲートウェイサーバ

とトランスレータの関係において、性能の限界が発生しうる代表的なシナリオを以下に示す。

1. 劣化キャッシュ限界シナリオ: ゲートウェイサーバの帯域を使い切りそうになり、対応して TTL を長くした結果、劣化キャッシュによる問い合わせエラーが頻発

2. トランスレータ負荷限界シナリオ: ゲートウェイサーバの帯域を使い切りそうになり、トランスレータの情報の取得の並列度を減らした結果、新しい情報の取得が間に合わず、同じトランスレータを多数の DNS-RS へ応答し、その結果トランスレータに性能以上の負荷が集中

説明のため、ゲートウェイサーバの設計例を、図 6 に示す。ゲートウェイサーバは、名前サーバとして動作し受け答えを行う部分 (DNS サービスモジュール) と、DHT のノードを収集する部分 (ノード情報収集モジュール) の二つから構成される。詳細には以上 2 つの部分に加えて、応答する NS RR の TTL を決定する機能 (TTL 決定モジュール) と、収集したノード情報を管理する機能 (トランスレータ情報管理モジュール) が存在する。

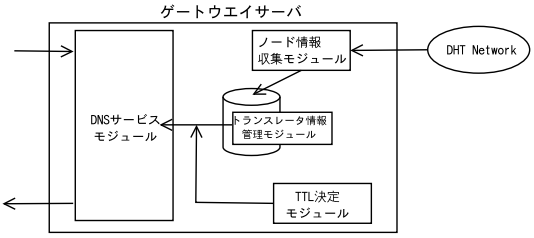

図 6: ゲートウェイサーバの設計

DNS サービスモジュールは、通常用いられている名前サーバと比べて特殊な処理を行うものではなく、通常の使用条件であれば計算量、ネットワーク入出力等における問題はないと考えられる。

一方、ノード情報収集モジュールは、各 DHT ノードとのやりとりが通信の全てである。すなわち、DHT ノードに接続し、その隣接ノードの情報を収集し、セッションを切断した上で、次のノードに接続する、というプロセスの繰り返しである。通信を暗号で守るなどの計算量の多い処理を行わない限り、最低限の通信処理だけでノード情報を収集できる。

ただし、通信に必要な時間、特に RTT や TCP の場合ネゴシエーションにかかるオーバーヘッドがあるので、DHT ノードを収集する速度には一定の上限があると考えられる。DHT ノードの収集速度 (node/sec) に比べて問い合わせ到着速度 (query/sec) が高い場合は、収集した DHT ノードを複数回使うか、余剰帯域があれば DHT ノードの収集を並列化して対処する。一方、DHT ノー

ドの収集速度が十分に高い場合は、古い情報を使い劣化キャッシュのリスク上昇を避けるため、新しい DHT ノードの情報と入れ替える。ここで、問い合わせ応答に帯域を消費し、DHT ノードの収集の速度あるいは並列度が低下した場合、前述したトランスレータ負荷限界シナリオが発生する。

また、ゲートウェイサーバは、DNS-RS に返す NS RR の TTL の調整によって負荷を調節可能であると考えられる。具体的には、ゲートウェイサーバの負荷が高い時には TTL を長くすることにより、DNS-RS からゲートウェイサーバの問い合わせ頻度を減少させられる。一方で、TTL を長くすることにより節 4.4 で述べた劣化キャッシュの問題が発生する、というトレードオフの関係にある。ここでゲートウェイサーバの負荷が上がるにつれ TTL を延ばしすぎることで、前述した劣化キャッシュ限界シナリオとなる。

6.2 劣化キャッシュの影響

$T = 250$ (ほぼ平均接続持続時間) の時点で、$M = 3$ では 8.8%、$M = 5$ では 2.8% の名前解決に失敗しており、$M = 10$ ではほぼ全て成功していることが読み取れる。また、ほぼ全ての名前解決に成功する T ので範囲は、今回の実験では $M = 5$ の時に $T = 125$、$M = 3$ では $T = 35$ 程度が上限であった。

今回の結果から、それぞれの M における T の上限を推測可能である。また、$M = 3$ 程度では平均接続持続時間に比べて 15% 程度の T しか確保できない。これに対して、M を向上させることにより T の上限を改善でき、$M = 10$ であれば平均接続持続時間に比べて 100% 程度の T が利用可能になる。

一方、劣化キャッシュ限界シナリオを考えると、T はこの上限を下まわる範囲で制御する必要がある。今回の結果は R_f の値があってはじめて適用可能な結果であり、実環境では R_f の正確な推測が必要である。しかし、筆者の知る限り、効率的な観測方法は確立していない。その結果、簡単な測定に頼ることになり、R_d の推測値の精度は低くなる。当然、安全のためのマージンを多く取る必要が出てくるため、T の限界は短くなる。従って、T の上限を長く取るためには、高精度かつ低コストに R_d の期待値を測定する方法が必要である。

7 議論

提案した名前空間マウント方式は、節 4.4 で述べた劣化キャッシュの問題以外にも、以下のような問題や、調査が必要な曖昧な点を抱えている。

第一に、プロトコル上の問題がある。RFC2181[4] によると単一の RRSet(あるドメイン名に対する特定の TYPE の RR の組) は単一の TTL を持っている必要がある。例えば、複数のトランスレータを記録しておき、それぞれの古さや信頼性、処理能力や現在の負荷に応じて動的に TTL を変更してトランス

レータの負荷を均衡化する方式が考えられるが、この方式は明確に RFC に違反してしまう。

また、動的に NS RR の組を合成しているが、この場合 DNSSec [3] の適用が困難になる。NS RR の組に対して毎回署名を行うとすると、署名の計算量が多いためゲートウェイサーバの負荷が上がり、その分スケーラビリティは低下する。一方で、NS RR を再利用し、再生成の回数を抑えればゲートウェイサーバの計算量負荷は下がるが、これはトランスレータの並列度の実質的な低下に繋がる。また、トランスレートゾーンの鍵をどう管理するか、という問題もある。ただし、DNSSec を適用する場合は DHT 自身に同等あるはそれ以上の強度を持つセキュリティ機構が必要となる。

第二に、負荷制御がどこまで能動的に行えるかという論点がある。TTL 制御による負荷制御は、前述の通りトランスレータの負荷の均衡化に用いる場合はプロトコル上の注意が必要である。一方、ゲートウェイサーバの負荷調整の目的であれば RRSet の TTL を統一できるので適用は容易である。ここでは、TTL 制御に対して問い合わせ頻度がどの程度の追従性を持ち、どの程度時間的余裕を見て TTL 制御を行うべきか、などの論点が考えられる。

第三に、DNS の権限の考え方に起因する問題がある。具体的には、ゲートウェイサーバがトランスレートゾーンの権限を持つサーバとして DNS-RS に送信する NS RR の組と、NS RR に格納されたトランスレータ自身がトランスレートゾーンに権限を持つサーバとして DNS-RS に送信する NS RR の組が食い違った場合、トランスレートゾーン自身に権限を持つトランスレータの情報が優先される。

従って、DNS-RS にとって NS RR の組とはトランスレータから返される値であり、DNS-RS に返す NS RR の組を保持管理すべきはゲートウェイサーバではなくてトランスレータである、ということになる。と同時に、トランスレータ自身は、自身の DHT 隣接ノードのうち適当なノード (トランスレータ機能を有するノード) を NS RR の組として返せなければならない。具体的には、M 個に 1 つの割合で中心となるノード (仮に SOA ノードと呼ぶ) を決定し、これを中心とした DHT ノードの集合を自律的に形成した上で、ゲートウェイサーバは SOA ノードから情報を収集する、という方式にする必要がある。

8 おわりに

本研究では、製品 ID 等のような非構造な名前空間の管理に適したデータインデックス手法の一つである DHT を DNS から利用可能にする方式 (名前空間マウント方式) に関する検討を行った。これにより、既存の DNS を用いた利用者環境から、非構造な名前空間を効率的に検索できる DHT の特性を透過的に利用できる。

名前空間マウント方式の特徴は、スケーラビリティの上限を決定づける 2 つの要素、即ち処理・回線負荷の高低と、並列度の高低を分解するすることで、プロトコル的な制約から脱し、制約下におけるトレードオフを追求する必要性をなくした所にある。

しかし、名前空間マウント方式は、DNS のプロトコルの意図とは異なる動作、つまり問い合わせに応じて名前空間の権限委譲を動的に行う。同時に、それぞれのトランスレータはネットワークから離脱する可能性があり、その結果劣化キャッシュが発生し、名前解決失敗となるリスクがある。本研究では、名前解決失敗のリスクをシミュレータを通じて検証し、一定の条件下でどの程度のリスクが発生するかを明らかにした。

今後は、第 7 章で述べた課題の解決や、平均接続持続時間の計測方法の研究、ゲートウェイサーバの帯域を考慮したスケーラビリティの考察、実動作環境における実験等を行う必要がある。最終的に、商品トレーサビリティシステムや ONS のような枠組に DHT を適用し、多数の ID の取り扱いを安価・効率的に行える環境の構築を目指す。

Copyright Notice

参考文献

[1] Auto-id Object Name Service (ONS) 1.0. Auto-ID Center Working Draft, August 2003.

[2] R. Cox, A. Muthitacharoen, and R. Morris. Serving dns using a peer-to-peer lookup service. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS)*, Cambridge, MA, March 2002.

[3] D. Eastlake. Domain name system security extensions. IETF RFC 2535, March 1999.

[4] R. Elz and R. Bush. Clarifications to the dns specification. IETF RFC 2181, July 1997.

[5] M. Mealling and R. Daniel. The naming authority pointer (naptr) dns resource record. IETF RFC 2915, September 2000.

[6] P. Mockapetris. Domain names - concepts and facilities. IETF RFC 1034, November 1987.

[7] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of ACM SIGCOMM*, August 2001.

[8] 土井裕介. 移動計算機環境における名前キャッシュの影響に関する考察. インターネットコンファレンス '98, 1998.

[9] 國領二郎, 日経デジタルコアトレーサビリティー研究会. デジタル ID 革命 IC タグとトレーサビリティーがもたらす大変革. 日本経済新聞社, 2004.