2006－DSM－40 (6)
2006／3／29

DHT性能評価法の提案と評価基盤の構築

加藤大志 神谷俊之
ＮＥＣ インターネットシステム研究所

DHTはP2P環境で分散データベースを構築する技術であり、P2Pのアプリケーションを構築するための基礎要素となる。DHTは比較的新しい技術であるため、性能を評価する環境がまだ整っていない。既存の評価環境は、アルゴリズムをシミュレーションで評価する手法が主であり、そのアルゴリズムを実現する実装が実用的なものか判断できないという制限がある。そこで、我々はDHTの実装を評価する基盤を開発した。この基盤は、大規模ネットワークにおけるpeerの動作やパケット遅延を再現し、アルゴリズムが正常に動作しているか確認し、通信量などを測定することができる。本基盤を用いて複数のDHT実装を同一条件で動作させたときの性能を比較した。

Evaluation system for distributed hash tables

Daishi KATO Toshiyuki KAMIYA
Internet Systems Research Laboratories, NEC Corporation

A distributed hash table (DHT) is a technique to make a distributed database in a peer-to-peer network, and it can be a building block for peer-to-peer applications. Since DHTs are relatively new, evaluation systems for DHTs are not available, especially for DHT implementations. To evaluate a practicality of a DHT, we developed an evaluation system for DHT implementations, which allows us to emulate many nodes in a large emulated network, check if algorithms work correctly, and measure resource usages, such as network traffic. We ran a preliminary experiment to compare some existing DHT implementations.

1. はじめに

近年、P2P (Peer-to-Peer)ネットワークの分野では、DHT (Distributed Hash Table, 分散ハッシュテーブル)の研究が盛んである。DHTはすべて対等なpeerで構成されたネットワークで疑似的なデータベースを提供する技術である。このデータベースへは、PUT(KEY, VALUE)でデータを登録し、GET(KEY)でデータを取得するが、これらをどのpeerから実行しても同じ結果を得ることができる。DHTの評価法としてはシミュレーションによる方法が一般的で、特にアルゴリズムの評価のみを行うものが多い。我々は、今後のDHTの実用性向上には実装を含めた評価が必要であると考え、そのための基盤を開発した。

我々が開発したDHT評価基盤「peeremu」は、少数のPCを使って多数のpeerをエミュレーションすることができ、同条件での繰り返し実験も可能である。さらに、実際のインターネット環境で見られるパケット遅延などエミュレーションを加えた性能評価も実現している。この評価基盤を使うことで、DHTの性能を実装レベルで確認することができるようになる。

2. DHTの性能評価の課題

DHTの性能評価の課題は、a) モデル化が困

難、b) 標準的なテスト法がない、c) 大規模性を必要とする、の 3 つがある。以下それぞれについて説明する。

a) DHT の性能は複雑な環境、すなわち、ネットワーク、各 peer の処理速度、負荷パターンなどにより大きく影響され、単純なモデル化が困難である。そのため、DHT アルゴリズムの提案は、まず、アルゴリズムの理論的な背景を説明し、それをシミュレーションで評価することにより、アルゴリズムの有用性を示すという手法をとっている。

b) DHT の研究は比較的新しいため、標準的なベンチマークセットやテストコレクションが存在せず、簡単なモデルやユースケースをそれぞれ独自に作る場合が多い。

c) DHT はもともと大規模性が一番の特徴であり、その大規模性を検証するには多くのリソースを必要とする。多くのマシンパワーが必要という意味では、HTTP サーバなどの負荷テストと似ているが、DHT の場合は peer それぞれの動作が相互に影響するため、HTTP サーバ単体の性能測定のように単純化できない部分がある。

従来の DHT の評価手法は大きく 3 種類に分類できる。一つは、ルーティングのシミュレーションで、アルゴリズムの実装時によく用いられる。従来例としては、アルゴリズム専用の Chord [1] simulator や Bamboo [2] simulator、および、複数のアルゴリズムをサポートした p2psim [3]がある。もう一つは、複数のアルゴリズムを比較可能なように独自に実装し、DHT の機能を、主に実時間で評価する方法がある。従来例として、MACEDON [4]や Overlay Weaver [5]などがある。最後の一つは、既に実装されている DHT を用いて、実時間で評価する方法である。OpenDHT [6] のチームが PlanetLab [7]で実施している実地での実験などがこの分類に相当する。

我々は、DHT の実用性評価には DHT の実装を含めた評価が重要と考え、実時間環境での評価法を採用し、再現性のある実験室環境で評価する手法を提案する。DHT 実装の評価は、数 peer で行うのは容易であるが、大規模ネットワークの実現は大変困難である。例えば、1000peer の実験を直接的に行う場合には、インターネット上に分散した 1000 台のマシンと 1000 人の利用者が必要になる。P2P のアプリケーション開発では、このように実地でのベータテストを行う場合もあるが、動作を確認する程度に留まる。性能を評価するという意味では、再現可能なコントロールされた環境で動作させることが望ましい。また、実働規模の peer 数のマシンを用意することなく、より少数のマシンで大規模なネットワークを評価できることが望まれる。この場合はインターネット環境をどのように再現するかという問題も解決する必要がある。性能評価を容易にするためには、各 peer の実験結果データを収集して集計・分析するツールが必要であり、また、条件を変えて繰り返し実験できるようにする必要がある。

3. DHT 評価基盤「peeremu」

3.1. peeremu の概要

我々は、独自 DHT を実装する上で動作の確認やデバッグを行う目的と、既存 DHT の実装の性能を比較評価する目的で、DHT 評価基盤「peeremu」を開発した。peeremu は一台の操作 PC と複数台の評価 PC で構成され、これらの PC は高速 LAN で接続されている。評価 PC は peer を動作させる PC で、操作 PC は評価 PC を制御する PC である。図 1 に

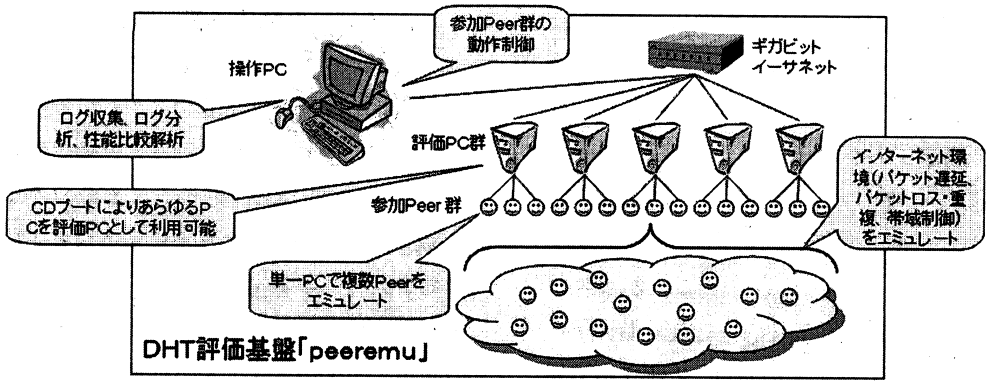

図 1:peeremu の構成図

peeremu の全体構成を示した。

peeremu は前節で述べた課題を解決するため、peer のエミュレーション、ネットワークのエミュレーション、試験のシナリオ化という特徴を持つ。

3.2. Peer のエミュレーション

実験室環境で大規模なネットワークを再現するためには、単一の PC で複数の peer を動作させることが必須である。そこで、一台の PC で動作する複数の peer がネットワーク上区別できるように、各 peer に仮想 IP アドレスを割り当て、外部からは完全に別の PC と認識されるようにした。また、Java で実装されているものは、JVM を多数起動することが困難であるため、一つの JVM で複数の peer を動作できるように改良を行い、単一 PC で多数の peer を動作できるようにした。

3.3. ネットワークのエミュレーション

インターネット環境での通信を LAN 環境で再現するためには、パケットの遅延を疑似的に発生させなければならない。パケットの遅延を実現するツールとして、dummynet [7] や NISTNet [8]などのツールがあるが、これらは単一のルータ機能として動作する。そのため、大規模な peer-to-peer ネットワークで使う場合、そのルータがボトルネックになり正常にパケットの遅延を発生させることができなくなる。これに対処するため、2 つの方法がある。一つは、ルータ機能をクラスタ化して大規模性を達成する方法(方法 1)で、ModelNet [10]で実現されている。もう一つは、ノード間のパケット遅延を各ノードで発生させる方法(方法 2)で、Linux の tc (traffic control)を使うことで実現できる。方法 1 は、ネットワークをモデル化するため輻輳を再現できるという特長があり、方法 2 は、ルータを構成する PC が不要なため導入が容易であるという特長がある。今回、我々は、導入の容易性を重視して、方法 2 を採用した。

3.4. 試験のシナリオ化

複数の PC に分散した peer を制御できるようにすべての peer の動作を事前にシナリオ化し、単一の PC (操作 PC) から評価 PC 群へのシナリオ配布、実行、データ収集機能を実装した。シナリオには各 peer の JOIN/LEAVE のタイミング、PUT/GET のタイミング、KEY/VALUE の内容が記述でき、複数回の実験や同じシナリオで異なる DHT 実装の比較評価ができる。さらに、peer 全体の JOIN/LEAVE のタイミングの分布パラメー

タから各 peer のシナリオを作るシナリオ生成ツールなども用意し、大規模なシナリオの生成を容易にできるようにした。

3.5. 測定データ

peeremu で収集できるデータは 4 種類ある。1 つ目は「GET 成功率」で、正しいデータを返した GET の割合を算出する。2 つ目は「GET 応答時間」で、GET を要求してから結果が返ってくるまでの時間 をミリ秒で算出する。3 つ目は「ネットワーク通信量」で、単位時間あたりに peer から送出されたパケットの数と総サイズを測定する。これらの測定データについては peer 単位で収集し、全 peer の平均値も算出できる。4 つ目は「リソース使用量」で、CPU のアイドル率とメモリの空き容量を測定する。リソース使用量は評価 PC 単位で収集される。

これらの測定データを、時間軸推移や異なるシナリオによる変動など、多角的に分析することにより、DHT の性能を評価することができる。

4. DHT 性能評価

4.1. peeremu の利用法

peeremu は主に以下の 2 つの利用法を目的とする。1 つの利用法は、ある DHT の実装がある「環境」のもとで正常に動作するか、また、どの程度のリソースを消費するかを確認することである。これは、新しい実装の開発における、問題点の発見やアルゴリズムの改良に有効である。もう 1 つの利用法は、複数の DHT の実装を同一の「環境」で動作させてリソース消費量などの性能を比較することである。ここで述べている「環境」とは、ネットワーク遅延、peer の総数や生存時間、各 peer の PUT/GET のパターンや頻度などである。

4.2. peeremu の動作確認・評価

peeremu が上記の利用法に適用できることを確認するために、peeremu のスケーラビリティ、安定性について評価を行った。確認項目は、1PC あたりで動作させることのできる peer 数の上限、同一シナリオによる複数実験での測定データの変化、および、同一パラメータから生成された複数シナリオによる実験での測定データの変化である。使用したマシン構成は操作 PC 1 台 (Pentium4 2GHz, メモリ 1GB)と、評価 PC 8 台(Pentium4 2.8GHz, メモリ 1GB)である。

まず、シナリオを作成(4.3 節で後述するパラメータを使用)して、PC あたりの動作 peer 数を変えて実験(実験 1)を行い、CPU のアイドル率とメモリ空き容量を計測した。この結果、1PC あたりの平均動作 peer 数は 60 くらいまでが、CPU アイドル率とメモリ空き容量に十分な余裕があり、妥当であることが分かった。

次に、同様のシナリオを用いて、同一パラメータ・同一シナリオでの 5 回繰り返し実験(実験 2)、同一パラメータ・別シナリオでの 5 回繰り返し実験(実験 3)を行った。実験 2 ではまったく同じ条件での繰り返し、実験 3 では、同一パラメータのため、peer の生存時間の分布の平均値等は同一であるがシナリオ生成時に異なる乱数を用いることで個別のシナリオを変化させた。総 peer 数を変化させて実験したところ、測定データの誤差率の最大値は表 1 のように、同一シナリオでは 0.1%未満、別シナリオでは 10%未満に抑えられることを確認した。これにより、これ以上の精度が必要な場合には、複数回の実験の平均をとるなどの方法が必要であることが分かった。

GET成功率	別シナリオ	0.07%
	同一シナリオ	0.05%
GET応答時間	別シナリオ	4.74%
	同一シナリオ	0.15%
通信量	別シナリオ	8.94%
	同一シナリオ	0.08%

表 1: 誤差率の最大値

4.3. 既存 DHT との比較評価

peeremu の一つの特徴は複数の DHT 実装を同じ「環境」で動作させて性能を比較できることである。そこで、既存の DHT 実装 2 つ(Bamboo, Chord)と我々が開発中の DHT 実装(GISPv5)を同一シナリオで比較した。

「環境」を定義する際には、p2psim を参考にしてランダムなイベントを生成した。具体的には、まず、peer 数の最大値(これを、ネットワークサイズと呼ぶ)を決め、各 peer がランダムに JOIN と LEAVE を繰り返すようにする。各 peer は JOIN 後に一定の間隔で、PUT と GET を実行する。PUT の KEY と VALUE はランダムに決め、GET の KEY は PUT されたものからランダムに抽出する。また、今回の実験では、PUT は初期に JOIN する peer のみが行うものとした。

life_mean	JOIN 時間の平均	60 分
death_mean	LEAVE 時間の平均	20 分
put_interval	PUT の間隔	20 秒
get_interval	GET の間隔	20 秒
end_time	試験時間	120 分

表 2: 実験パラメータ

表 2 に、今回の実験で使用したパラメータの一覧を載せる。ネットワークの遅延に関しては、p2psim と同様に king data [11, 12]と呼ばれる実測データを利用した。

この条件で、ネットワークサイズを 180 から 1120 まで変化させて実験を実施した結果を図 2、3、4 に載せる。図 2 のグラフでは、Chord の成功率の低下が見られるが、これは Chord のパラメータ(今回はデフォルトのものを使用)の影響であると考え、全体的には安定しているものと判断する。図 3 のグラフでは、Chord の応答時間の増加が顕著にあらわれている。図 4 のグラフでは、GISPv5 と Chord の通信量が少なく、Bamboo は比較的多いことが分かる。これらの結果から、Bamboo は通信量を多く使うことにより安定性と応答性を確保する一方、GISPv5 は少ない通信量を実現している点で有利であると考えられる。Chord の通信量は中間的な性能と考えられるが、詳細を知るにはさらなる実験が必要である。

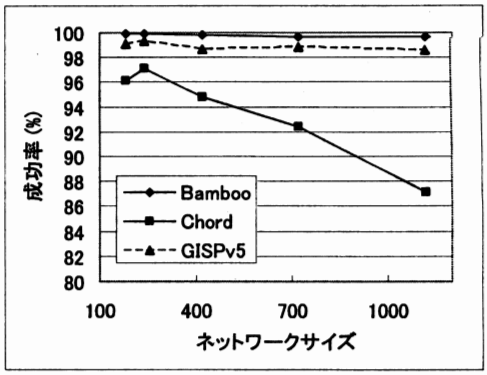

図 2: GET 成功率

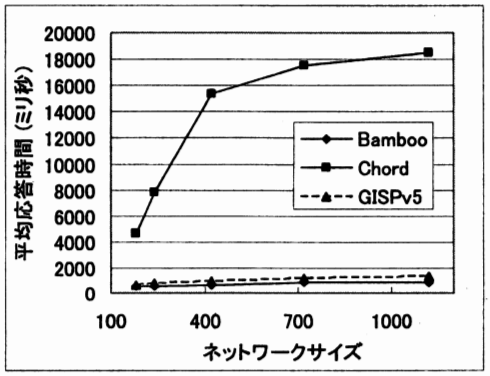

図 3: GET 応答時間

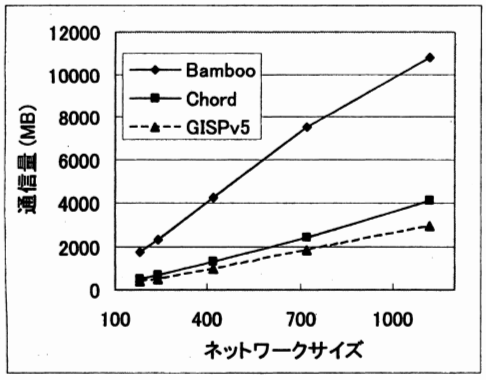

図 4: 総通信量

5. おわりに

DHT は学術的には理論が固まってきており、学術的実装に加えてアプリケーションでの実装も開発されている。しかしながら、分散ネットワークを必要とする性質上、性能評価が難しかった。そこで、我々は DHT の性能を実装レベルで評価可能にする評価基盤 peeremu を開発した。peeremu は大規模なネットワーク上での DHT の動きを再現でき、性能指標となるデータを収集することができる。3 つの DHT 実装を比較評価したところ、それぞれの特徴を明らかにすることができた。今後の課題としては、a) シナリオの多様化、b) パケットロスの導入、c) 評価対象の追加、d) GISPv5 の改良、がある。

a) peeremu のシナリオは、peer 毎に動作を記述する基本的なものであるため、記述力は非常に高い。しかし、シナリオ生成ツールに設定できるパラメータは限られており、peer の動作に偏りを導入することができない。今後、これらのツールを改良する。

b) Linux の tc は、パケットロスなども発生させることができる。今後、これも導入した実験を行う。

c) 評価した 3 つの DHT 実装以外にも、DHT 実装は存在する。今後、評価対象とする DHT 実装を増やすべきである。

d) 実験の結果を基に GISPv5 を改良し、より実用性のある実装を実現する。これが peeremu を開発した一つの動機でもある。

6. 参考文献

[1] I. Stoica, et al. *Chord: A scalable peer-to-peer lookup service for internet applications.* In Proc. of ACM SIGCOMM (Aug. 2001).

[2] S. Rhea, et al. *Handling Churn in a DHT.* In Proc. of the USENIX Annual Technical Conference, June 2004.

[3] J. Li et al. *Comparing the performance of distributed hash tables under churn.* In Proc. of the 3rd International Workshop on Peer-to-Peer Systems (Feb. 2004).

[4] A. Rodriguez et al. *MACEDON: Methodology for Automatically Creating, Evaluating, and Designing Overlay Networks.* In Proc. of NSDI 2004.

[5] http://overlayweaver.sourceforge.net/

[6] S. Rhea, et al. *OpenDHT: A Public DHT Service and Its Uses.* In Proc. of ACM SIGCOMM 2005 (August 2005).

[7] http://www.planet-lab.org/

[8] http://info.iet.unipi.it/~luigi/ip_dummynet/

[9] http://snad.ncsl.nist.gov/itg/nistnet/

[10] A. Vahdat, et al. *Scalability and Accuracy in a Large-Scale Network Emulator.* In Proc. of 5th Symposium on Operating Systems Design and Implementation (OSDI), (December 2002).

[11] K. P. Gummadi, et al. *King: Estimating Latency between Arbitrary Internet.* In Proc. of the SIGCOMM Internet Measurement Workshop (IMW 2002).

[12] http://pdos.csail.mit.edu/p2psim/kingdata/