構造化オーバーレイにおけるデータ間の関係性に基づくマルチクエリに対する応答の高速化

Speeding Up of Response to a Multi-query Using Relation of Data in Structured Overlay Networks

小泉 悠介　Yusuke Koizumi
渡部 康平　Kohei Watabe
中川 健治　Kenji Nakagawa

長岡技術科学大学 大学院 工学研究科
Graduate School of Engineering, Nagaoka University of Technology

1　概要

構造化オーバーレイネットワーク [1] では，負荷分散を行うためにデータがネットワーク上の各ノードへと分散して保存される．構造化オーバーレイネットワークにおいて，特定の条件に該当するデータを複数取得するという要求 (マルチクエリ) を行った場合，分散して保存したデータを取得するのに，複数のノードを探索することが必要となる．よって，マルチクエリに含まれるデータを全て探索するまでに多くの時間を要するという問題点がある．そこで本研究では，分散して保存されている各データ間の関連性に基づきデータの配置方法を工夫する．関連性に基づくデータの配置により，マルチクエリに含まれるデータを 1 つのノードで取得できるようにすることで，マルチクエリに対する応答を高速化する手法を提案する．

2　提案法について

提案法では，各ノードにおいて，保存されているデータと関係性が深いと考えられるデータを他ノードから複製して保存する．提案法を用いない場合に比べて，多くの保存領域が必要となるが，各ノードでマルチクエリに含まれるデータが全て保存されている場合が多くなるため，応答を高速化することができる．関係性が深いと考えられるデータは，過去に送られてきたクエリの集合 A を利用することで推定する．各ノードは，A に含まれるマルチクエリのうち，マルチクエリに含まれる全てのデータがノードに保存されているマルチクエリの数を最大化する．

上記の問題は，整数計画法により下記のように定式化することができる．各ノードにおいて，ソルバーを用いて下記の問題の解を求めることにより，どのデータを保存するか決定する．

$$\begin{cases} \text{maximize} & \sum_{i=1}^{n} x_i \\ \text{subject to} & \sum_{j=1}^{m} y_j \leq c, \\ & y_j \geq d_{ij}x_i, \quad \forall i,j \end{cases}$$

$$\begin{pmatrix} c & : & \text{他ノードから複製したデータの保存領域のサイズ} \\ A & : & \text{利用するクエリの集合} \\ B & : & A \text{ に含まれているデータの集合} \\ n & : & A \text{ の要素数} \\ m & : & B \text{ の要素数} \\ a_i \in A & : & A \text{ の } i \text{ 番目のクエリ } (1 \leq i \leq n) \\ b_j \in B & : & B \text{ の } j \text{ 番目のデータ } (1 \leq j \leq m) \\ d_{ij} \in \{0,1\} & : & a_i \text{に } b_j \text{が含まれている場合は 1，それ以外は 0 となる指示関数} \\ x_i \in \{0,1\} & : & a_i \text{に含まれるデータを保存する場合は 1，それ以外は 0 となる指示関数} \\ y_j \in \{0,1\} & : & b_j \text{を保存する場合は 1，それ以外は 0 となる指示関数} \end{pmatrix}$$

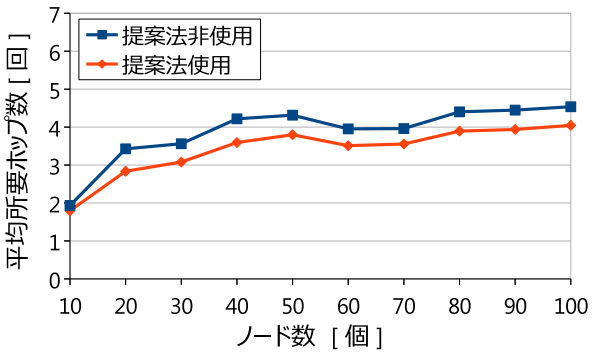

図 1　シミュレーション結果

3　シミュレーション

文献 [1] で提案された構造化オーバーレイネットワークを構成するノード数を変化させたとき，全データ取得までのクエリの平均所要ホップ数がどのように変化するかを，提案法を用いない場合と用いた場合それぞれについてシミュレーションを行い測定した．今回シミュレーションで用いたクエリ生成モデルでは，データは 2 次元トーラス上で関係性を表現できるとし，任意の長方形で囲む範囲のデータがマルチクエリとして要求されるとする．長方形の各辺および左下頂点の座標は形状パラメータ $s = 1.4$ のジップ分布に従う偏りのある乱数によって決定する．ネットワーク全体で保存してあるデータ数を 10000[個]，他ノードから複製したデータの保存領域のサイズを 30[個] として，ノード数を 10[個] から 100[個] まで変化させた．このときマルチクエリ処理を 1000[回] 行い平均所要ホップ数を算出した．各ノードで前述の問題に対する解を算出するのに要する時間は数秒程度であった．

シミュレーション結果を図 1 に示す．図 1 より，提案法を用いない場合に比べ，提案法を用いた場合の方が平均所要ホップ数が減少していることが分かる．これは，提案法を用いることによって，各ノードでマルチクエリに含まれるデータが全て保存されている場合が多くなるためである．

4　今後の方針

保存するデータを「マルチクエリに含まれる頻度が高いデータ」とした際に，平均所要ホップ数がどの程度削減されるか調査する．

謝辞

本研究の一部は，JSPS 研究活動スタート支援 26880008 の助成を受けたものである．

参考文献

[1] Ion Stoica *et al.*, “Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications”, IEEE-ACM Trans. Netw., VOL.11, NO.1, 2003.