情報処理学会論文誌 Vol. 50 No. 1 430–436 (Jan. 2009)

テクニカルノート

P2P ネットワークにおける top-k 検索のための軽量インデックス

成毛源樹[†1] 山口和紀[†1]

本研究では P2P ネットワーク上のファイルに対して top-k 検索を実現するためのインデックスとして，ノードごとに構築されるインデックスのうち，検索語に対する最高スコア値を持つファイルの参照を含む要素だけを集約した，軽量なインデックスを構築する手法を提案する．検索語を 1 つに限定した状況で，インデックスの更新が頻繁に起こる場合，提案手法が通信量の削減に有効であることをシミュレーションにより示した．

The Lightweight Index for Top-k Retrieval in P2P Networks

Motoki Naruke[†1] and Kazunori Yamaguchi[†1]

We introduce a lightweight index for top-k retrieval in P2P networks. The lightweight index consists of the references to the highest scoring files for every search word and node. In this paper, we use one word in search for simplicity. We conducted a simulation and it's result shows effectiveness in reducing traffic in the situation that index is updated frequently.

1. はじめに

既存のファイル共有ソフトウェアにおけるファイル検索では，検索要求として与えたキーワード（検索語）がファイル名にマッチするファイルすべてが検索結果として返されるため，価値があるファイルがどれであるかを判断することが難しいという問題がある．

この問題を解決するために，P2P ネットワーク上のファイルに対して，データベースで用いられている top-k 検索[1] を適用することを考える．top-k 検索とは，検索要求として与えられた検索語にマッチするファイルのうち，検索語に対してファイルに付与されたスコア値上位 k 件のファイルを検索する手法である．top-k 検索では，検索対象のファイルを直接参照するのではなく，検索に必要な情報を記録し，必要なときに素早く参照することができる，インデックスと呼ばれるファイルをあらかじめ構築しておき，検索時に用いることが一般的である，P2P ネットワーク上のファイルに対して top-k 検索を実現するものとして，すでにいくつかの手法[6),10)] が提案されているが，既存手法ではノードごとに構築されるインデックス（これをローカルインデックスと呼ぶ）を集約したインデックス（これをグローバルインデックスと呼ぶ）を構築することが必要となる．しかし，ローカルインデックスのサイズは，一般に，ノードが保有するファイル数に対して線形に増加するため，ローカルインデックスの内容すべてを集約してグローバルインデックスとすると，集約時に発生する通信量，構築されるグローバルインデックスのサイズもファイル数に対して線形に増加してしまうという問題がある．

この問題を解決するために，インデックスに記録する情報を様々に削減する手法[3),13)] が提案されているが，既存手法では，検索漏れなど，新たな問題が発生してしまう．そこで本研究では，ローカルインデックスごとに，検索語に対する最高スコア値を持つファイルの参照を含む要素だけを集約することで，インデックス構築時の通信量，構築されるインデックスのサイズを削減する手法を提案する．提案手法により構築されるインデックスを用いた検索では，既存の top-k 検索同様，複数の検索語による検索が可能であると考えられるが，本論文では単純化のため，検索語を 1 つに限定した結果を報告する．

提案手法により構築されるインデックスは，ローカルインデックスの内容すべてを集約して構築したグローバルインデックスと比較すると，検索時に，より多くの通信が発生する．本研究では，シミュレーションにより，検索時に発生する通信量のオーバヘッドが，ノード数およびファイル数の増加に従って一定の値に収束することを示し，ローカルインデックスが頻繁に更新されるような状況で提案手法が有効であることを示す．ただし本研究では，インデックスの更新については，追加のみを考える．

2. P2P ネットワークにおける検索手法

P2P ネットワークにおけるファイル検索手法として最初に用いられていたのは，フラッ

†1 東京大学大学院総合文化研究科
Graduate School of Arts and Sciences, The University of Tokyo

ディング方式[2] である．フラッディング方式では，事前にインデックスを構築せずに，検索を始めたノードから隣接ノードへ，隣接ノードの隣接ノードへと，検索要求をバケツリレーのように伝搬していくことで検索を行う．フラッディング方式を用いて top-k 検索を実現することも可能ではあるが[4]，検索時に著しく多数のノードと通信を行う必要があり，効率が悪い．よって，本研究では，フラッディング方式については考慮しない．

フラッディング方式の問題点を改善するための手法として，ネットワークを階層化するスーパノード方式[11),12)] と，ネットワークを構造化する DHT（分散ハッシュテーブル）方式[5),7)–9)] が提案された．

スーパノード方式では，ノードをスーパノードと一般ノードに分ける．スーパノードは他のすべてのスーパノードと通信するが，一般ノードはスーパノードのうち，1 つだけと通信する．通信する関係にあるスーパノードと一般ノードを，それぞれ親ノード，子ノードと呼ぶ．親ノードはすべての子ノードのローカルインデックスを集約し，子ノードが保有するファイルのインデックス（これをノードインデックスと呼ぶ）を構築する．スーパノード方式の検索は，検索を行いたいノードが，自分の親ノードに検索要求を送信することで行われる．検索を行いたいノードが親ノードである場合，親ノードへの検索要求の送信は省略する．検索要求を受信した親ノードは，ノードインデックスを用いて，検索要求にマッチするファイルを返すとともに，他のスーパノードに検索要求を送信する．検索要求を受信したスーパノードは，親ノードと同様に検索を行い，結果を返すことで，ネットワーク上のファイルすべてに対する検索が行われる．

DHT 方式では，キーを指定して，対応する情報をネットワーク上に登録したり取得したりすることが可能である．検索語をキーとして，検索語にマッチする要素を DHT 上に登録することで，検索語から要素を検索することが可能となる．以下，DHT 方式で，キーに対応する情報を登録するノードを，キーの担当ノードと呼ぶ．

3. top-k 検索のためのインデックス

top-k 検索を実現するために，ローカルインデックスを集約し，グローバルインデックスを構築する．ローカルインデックスは，検索語について，検索語にマッチするファイルの参照，ファイルを保有するノードの IP アドレス，検索語に対するファイルのスコア値の組を記録する．ローカルインデックスを用いた top-k 検索を，式 (1) で表す．

$$L^x(w,k,m) = \{(f_i, n_i, s_i) \mid m \le s_i, 1 \le i \le k\} \quad (1)$$

x はローカルインデックスを持つノードの IP アドレスである．検索語 w に対する $L^x(w,k,m)$ の結果は，w にマッチするファイルの参照 f_i，f_i を保有するノードの IP アドレス n_i，w に対するスコア値，上位 i（$1 \le i \le k$）番目の値 s_i（$m \le s_i$）の組の集合である，m は必要のないスコア値を含むタプルを切り捨てるための閾値で，本当に必要な結果だけを送信することで，通信量を削減するために用いる，以下，インデックスの要素である (f, n, s) のような組をタプルと呼ぶ．また，$L^x(w,\infty,0)$ は，w にマッチするすべてのタプルからなる集合を表すものとする．

3.1 節では，ローカルインデックスの要素をすべて集約した，完全なグローバルインデックスについて，3.2 節では，提案手法である，軽量なグローバルインデックスについて，それぞれ構築，検索手法を述べる．

3.1 完全なグローバルインデックス

完全なグローバルインデックスは，ネットワーク上のすべてのノードの，ローカルインデックス中の検索語それぞれについて，$L^x(w,\infty,0)$ の結果を集約し，記録する．完全なグローバルインデックスを用いた top-k 検索を，式 (2) で表す．

$$G^x(w,k) = \{(f_i, n_i, s_i) \mid 1 \le i \le k\} \quad (2)$$

x はグローバルインデックスを持つノードの IP アドレスである．検索語 w に対する $G^x(w,k)$ の結果は，w にマッチするファイルの参照 f_i，f_i を保有するノードの IP アドレス n_i，w に対するスコア値上位 i（$1 \le i \le k$）番目の値 s_i の組である．

以下，スーパノード方式と DHT 方式における，完全なグローバルインデックスの構築法を述べる．

スーパノード方式では，一般ノードが自分の親ノードに，ローカルインデックス中の検索語それぞれについて，$L^x(w,\infty,0)$ の結果を送信することで，スーパノード上にノードインデックスを構築する．次に，スーパノードが自分以外のすべてのスーパノードとノードインデックスを交換して集約することで，すべてのスーパノード上に，すべての検索語についての完全なグローバルインデックスを構築する．

DHT 方式では，ネットワーク上のノードが，ローカルインデックス中の検索語それぞれについて，検索語の担当ノードに $L^x(w,\infty,0)$ の結果を送信して登録することで，検索語の担当ノード上に，検索語についての完全なグローバルインデックスを構築する．

ネットワーク上のノードにファイルが追加され，ローカルインデックスが更新された場合，差分の情報をスーパノード，あるいは DHT 上のノードに送信することでグローバルインデックスを更新する．

完全なグローバルインデックスを用いた top-k 検索は，グローバルインデックスを持つ

ノードに検索語と k を送信し，$G^x(w,k)$ の結果を受信することで行われる．スーパノード方式では，親ノードがつねにグローバルインデックスを持っているが，DHT 方式では検索語のグローバルインデックスを持つノードを探索する必要がある．一般に，ネットワーク上のノード数を N としたとき，DHT 上でキーの担当ノードの探索には $\log N$ 回の通信が必要である．

3.2 軽量なグローバルインデックス

軽量なグローバルインデックスは，ネットワーク上のすべてのノードの，ローカルインデックス中の検索語それぞれについて，$L^x(w,1,0)$ の結果を集約したインデックスである．$L^x(w,1,0)$ の結果とは，検索語にマッチするタプルのうち，検索語に対するスコア値が最高のものである．以下，スーパノード方式と DHT 方式における，軽量なグローバルインデックスの構築法を述べる．

スーパノード方式では，一般ノードが自分の親ノードに，ローカルインデックス中の検索語それぞれについて，$L^x(w,1,0)$ の結果を送信することで，スーパノード上に軽量なノードインデックスを構築する．このとき，タプル中のファイルの参照は削除する．以上により構築された軽量なノードインデックスを用いた top-k 検索を式 (3) で表す．

$$N^x(w,k,m) = \{(n_i, s_i) \mid m \le s_i, 1 \le i \le k\} \tag{3}$$

x はノードインデックスを持つノードの IP アドレスである．検索語 w に対する $N^x(w,k,m)$ の結果は，w に対するスコア値，上位 i（$1 \le i \le k$）番目の値 s_i（$m \le s_i$），s_i を付与されたファイルを保有するノードの IP アドレス n_i である．次に，スーパノードはノードインデックス中の検索語それぞれについて，$N^x(w,1,0)$ のタプル中のノードの IP アドレスを，自分の IP アドレスに変更した結果を，自分以外のスーパノードと交換して集約したインデックスを構築する，このインデックスをスーパノードインデックスと呼ぶ．以上により構築されたスーパノードインデックスを用いた top-k 検索を式 (4) で表す．

$$S^x(w,k,m) = \{(p_i, s_i) \mid m \le s_i, 1 \le i \le k\} \tag{4}$$

x はスーパノードインデックスを持つノードの IP アドレスである．検索語 w に対する $S^x(w,k,m)$ の結果は，w に対するスコア値，上位 i（$1 \le i \le k$）番目の値 s_i（$m \le s_i$），s_i を付与されたファイルを保有するノードの親ノードの IP アドレス p_i である．以上の手順で構築された，軽量なノードインデックスとスーパノードインデックスを合わせて，スーパノード方式における軽量なグローバルインデックスと呼ぶ．

DHT 方式ではネットワーク上のノードが，ローカルインデックス中の検索語それぞれについて，検索語の担当ノードに $L^x(w,1,0)$ の結果を送信して登録する．このとき，タプル中のファイルの参照は削除することで，検索語の担当ノード上に，検索語についての軽量なグローバルインデックスを構築する．DHT 上に構築される軽量なグローバルインデックスは，式 (3) で表される，スーパノード上に構築される軽量なノードインデックスと同じものである．

軽量なグローバルインデックスでは，ネットワーク上のノードにファイルが追加され，ローカルインデックスの，検索語に対するスコアの最高値が更新されたときのみ，差分の情報をスーパノード，あるいは DHT 上のノードに送信することでグローバルインデックスを更新する．

スーパノード方式による，軽量なグローバルインデックスを用いた top-k 検索のアルゴリズムを**図 1** に，DHT 上方式による，軽量なグローバルインデックスを用いた top-k 検索のアルゴリズムを**図 2** に示す．

入力：検索語 w, 取得件数 k
出力：検索結果 r

$x \leftarrow$ スーパノードの IP アドレス
$m \leftarrow 0$
$r \leftarrow S^x(w,k,m)$
while (r がノードを参照する (s,n) を含む) do
　r から (s,n) を除く
　$m \leftarrow r$ の最低スコア値
　if (n がスーパノードである) then
　　$N^n(w,k,m)$ の各タプルを r に加える
　else
　　$L^n(w,k,m)$ の各 (s,f) に対して (s,f,n) を
　　r に加える
　r のスコア値上位 k 件のみを残す
done

図 1　スーパノード方式による，軽量なグローバルインデックスを用いた top-k 検索のアルゴリズム
Fig. 1　Top-k retrieval algorithm with lightweight index on super-node.

入力：検索語 w, 取得件数 k
出力：検索結果 r

$x \leftarrow w$ の担当ノードの IP アドレス
$m \leftarrow 0$
$r \leftarrow N^x(w, k, m)$
while (r がノードを参照する (s, n) を含む) do
　r から (s, n) を除く
　$m \leftarrow r$ の最低スコア値
　$L^n(w, k, m)$ の各 (s, f) に対して (s, f, n) を
　　r に加える
　r のスコア値上位 k 件のみを残す
done

図 2　DHT 方式による，軽量なグローバルインデックスを用いた top-k 検索のアルゴリズム
Fig. 2　Top-k retrieval algorithm with lightweight index on DHT.

4. 実　　験

完全なグローバルインデックスと，軽量なグローバルインデックスを比較して，構築時に削減される通信量と，検索時に発生するオーバヘッドを，シミュレーションにより確認した．

シミュレーションでは，ネットワーク上のノード数を 10 から 10,000 まで変化させて，それぞれのノードが 1,000 ファイルを保有するものとした．ネットワーク全体のノード数を N としたとき，スーパノード方式におけるスーパノードの数を $\sqrt{N}$ とした．これは，スーパノードの負荷と通信効率のバランスを考えてのものである．ノードが，他のノードを探索するときに必要な通信量を 4 バイト，インデックス中のタプル 1 つを送信するために必要な通信量を 8 バイトとした．検索語にはアルファベット 2 文字の全組合せ（2-gram）を用いた．検索語とファイルのマッチングは “ファイル中に検索語が出現するか否か” で行った．検索語に対するファイルのスコア値は “ファイル中に出現する検索語の出現頻度” とした．ノードが保有する文書は，http://gutenberg.org/より取得した 10,000 文書をもとに，2-gram の出現頻度の平均と分散を算出し，10,000,000 文書分の出現頻度のデータが正規分布となるようにランダムに生成した．

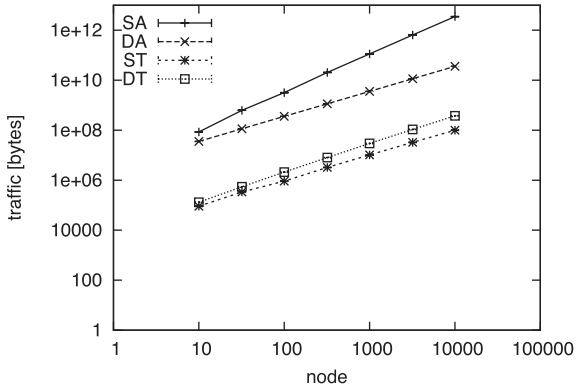

図 3　グローバルインデックス構築時に発生する通信量
Fig. 3　Construction cost of global index.

以下，完全なグローバルインデックスをスーパノード方式で構築したものを SA，DHT 方式で構築したものを DA，軽量なグローバルインデックスをスーパノード方式で構築したものを ST，DHT 方式で構築したものを DT と呼ぶ．

4.1　グローバルインデックス構築時の通信量

各手法において，グローバルインデックス構築時にネットワーク全体で発生する通信量を**図 3** に示す．横軸はノード数で，縦軸は発生する通信量である．通信量は 10 回の試行の結果を平均して用いた．

図より，DA は ST および DT に対して，およそ 1/保有ファイル数（= 1/1000）程度の通信量となっていることが分かる．これは，軽量なグローバルインデックス構築時は，検索語それぞれについて，ノードごとに，検索語に対する最高スコア値を持つタプル 1 つだけを登録すればよいためである．SA の通信量が極端に多いのは，スーパノード間でノードインデックスを交換する通信量が反映されていると考えられる．

4.2　グローバルインデックス更新時の通信量

各手法において，ノード数 10,000 のネットワーク上のノードを 1 つ選び，選んだノードに 1,000 ファイルを追加するという操作を 1,000 回行った．各回目のファイルの追加により発生する，グローバルインデックス更新に必要な通信量をプロットしたものが**図 4** である．図の横軸はファイルを追加した通算回数で，縦軸は各回目のファイル追加により発生する通信量である．通信量は 10 回の試行の結果を平均したものを用いた．

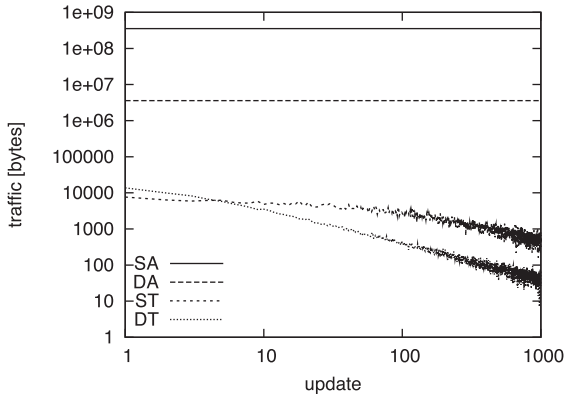

図 4　グローバルインデックス更新時に発生する通信量
Fig. 4　Update cost of global index.

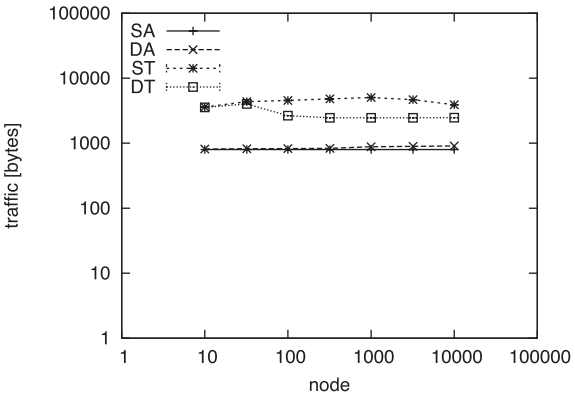

図 5　上位 100 ファイル検索時に発生する通信量
Fig. 5　Retrieval cost of top-100 files.

SA，DA ではファイルが追加されると，そのすべてをグローバルインデックスに反映させる必要があるため，ファイル追加時に必ず通信が発生していることが分かる．ST，DT では，ノードごとに，検索語に対するスコアの最高値が更新されたときのみ，グローバルインデックスを更新すればよいので，ファイル追加時の平均的通信量は少ない．また，ファイルを追加する回数が増えると，ノードごとの，検索語に対するスコアの最高値が更新される頻度が減るため，グローバルインデックスを更新するために必要な通信量が減少していることが分かる．

ST，DT において，毎回，追加されたファイルに含まれる，すべての検索語に対するスコアの最高値が更新される場合，毎回目において，1 回目と同じ通信量が発生することになる．このときの通信量が 1,000 回続けて発生した場合の累積値と，実際に発生する通信量 1,000 回の累積値を比較すると，実際に発生する通信量は半分程度となっている．

4.3　グローバルインデックスを用いた検索時の通信量

各手法において，検索時に発生する通信量を**図 5** に示す．横軸はノード数で，縦軸は $k = 100$ としたときの top-k 検索時に発生した通信量である．通信量は 10 回の試行の結果を平均したものを用いた．

SA では親ノードに検索語 w，k を送信するために 8 バイト，$G^x(w,k)$ の結果を送信するために $8 \times k$ バイトの通信量が発生する．SA で発生する通信量はノード数によらず，一定の値となる．

DA では，w についてのグローバルインデックスを持つ，w の担当ノードを発見するために $4 \times \log N$ バイト，w の担当ノードに w，k を送信するために 8 バイト，$G^x(w,k)$ の結果を送信するために $8 \times k$ バイトの通信量が発生する．DA では，ノード数の増加により担当ノードを発見する通信量が増加するが，全体で発生する通信量はほぼ一定となる．

図より，ST ではノード数の増加により通信量が増加しているが，ノード数 1,000，約 5,000 バイトを境に，通信量が減少していき，ノード数 10,000 で約 4,000 バイトになっていることが分かる．

DT ではノード数の増加により通信量が増加しているが，ノード数が 32 から 100 に変化すると，通信量が約 4,000 バイトから約 2,500 バイトへと減少した後，再びノード数の増加により通信量が増加していることが分かる．

以上より，軽量なグローバルインデックスを用いた検索では，ノード数の増加により通信量が増加するが，特定の条件により通信量が減少することが分かる．これは，ノード数とともにネットワーク上のファイル数が増加すると，軽量なグローバルインデックスに集約されるスコア値の差が縮まっていき，各ノードにおける最高スコア値以外に，閾値として与える，検索結果中の最低スコア値を超えるスコア値が存在しなくなっていくためであると考えられる．このとき，$N^x(w,k,m)$ および，$L^x(w,k,m)$ の結果は，各ノードにおける最高スコア値を含むタプル 1 つだけになる．以上のような状態のとき，$k = 100$ として，ST では，親ノードに w，k，m を送信するために 8 バイト，$S^x(w,k,m)$ の結果を送信するために

8×100 バイト，送信された結果に含まれる 100 ノードそれぞれについて，w，k，m を送信，$N^x(w,k,m)$ の結果を送信，送信された結果のノードに w，k，m を送信，$L^x(w,k,m)$ の結果を送信するために，$100 \times (8+8+8+8)$ バイト，まとめると $808 + 3200 = 4008$ バイトの通信量が発生する．同様に，DT で発生する通信量は，w の担当ノードを探索するために $4 \times \log N$ バイト，w の担当ノードに w，k，m を送信するために 8 バイト，$N^x(w,k,m)$ の結果を送信するために 8×100 バイト，送信された結果に含まれる 100 ノードそれぞれについて，w，k，m を送信，$L^x(w,k,m)$ の結果を送信するために，$100 \times (8+8)$ バイト，まとめると $4 \times \log N + 808 + 1600 = 4 \times \log N + 2408$ バイトの通信量が発生する．以上より，ノード数が 10,000 以上に増加しても，ST では 4,000 バイト，DT では $4 \times \log N + 2400$ バイトの通信量に収束することが予想される．

4.4　総合的な通信量の比較

各手法において，ノード数が 10,000 のとき，グローバルインデックス構築時に発生する通信量，グローバルインデックス更新時に発生する通信量，グローバルインデックスを用いた検索時に発生する通信量を足し合わせ，通信量が最小となる方式をプロットした結果を**図 6** に示す．横軸は，ネットワーク上のノードにファイルを追加した回数，縦軸は，$k = 100$ としたときの top-k 検索を行った回数である．ファイルの追加は毎回，ネットワーク上のノードをランダムに選び，1,000 ファイルを追加した．ST，DT において，ノードへファイルを追加した回数に対して発生する通信量は，図 4 のグラフを直線で近似した関数を用いた．

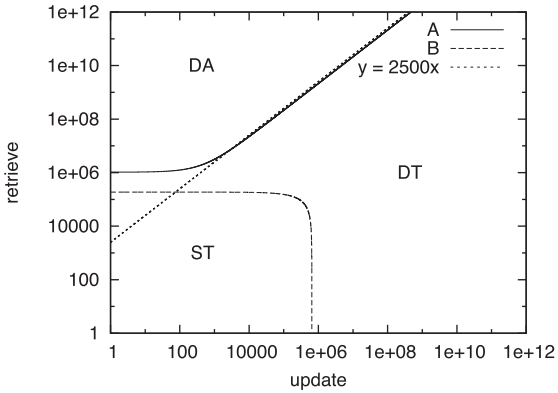

図 6　総合的に発生する通信量の比較
Fig. 6　Comparison of overall costs.

図より，曲線 A で囲まれた左上の領域では DA が，曲線 B で囲まれた左下の領域では ST が最も通信量が少なく，それ以外の領域では DT が最も通信量が少ない方式であることが分かる．ノードにファイルを追加した回数が 1,000 回あたりまでは，検索回数が 1,000,000 を超えない限り，提案手法である ST あるいは，DT により発生する通信量が最小となる．ノードにファイルを追加した回数が 1,000 回を超えたあたりから，DA と DT の境界が直線となり，ファイルの追加回数に対する検索回数の比率で通信量が最小となる手法が変化する．この直線と $y = Ax$ とのフィッティングを行った結果，$A = 2500$ で一致した．以上より，検索回数がファイル追加回数の 2,500 倍以下であるような状況ならば，提案手法である DT により通信量の削減が可能であることが分かる．

提案手法により発生する通信量が最小となるファイル追加回数と検索回数の比率である，2,500 という値は，計算により導くことが可能である．DA により，インデックス更新時に発生する通信量を U，検索時に発生する通信量を R とする．DT により，更新時に発生する通信量のうち，ファイル追加回数の増加にともない削減される割合を α，検索時に発生するオーバヘッドを β，DA と DT の境界の直線の傾きとなる，ファイル追加回数と検索回数の比率を γ とすると，式 (5) が成り立つ．

$$U + R\gamma = \alpha U + \beta R\gamma \tag{5}$$

式 (5) を変形することで得られる式 (6) の左辺に，

$$\frac{(1-\alpha)U}{(\beta-1)R} = \gamma \tag{6}$$

図 4，図 5 から読み取れるおおまかな値，$U = 10^9$，$R = 800$，$\alpha = 0.5$，$\beta = 2500/800 = 3.125$ を代入すると，$\gamma = 2900$ となり，2,500 にほぼ一致する．

5. ま と め

本研究では P2P ネットワーク上のファイルに対して top-k 検索を実現するために，グローバルインデックスを構築する手法について検討した．top-k 検索はローカルインデックスをすべて集約した完全なグローバルインデックスを構築することで実現可能であるが，構築時に多大な通信量が発生する，構築されるインデックスのサイズが膨大となる，という問題がある．この問題を解決するために，本研究では，ローカルインデックスごとに，検索語に対する最高スコア値を持つファイルの参照を含むタプルだけを集約した，軽量なグローバルインデックスを構築する手法を提案し，更新が頻繁に起こる状況で，本手法が有効であることをシミュレーションにより示した．

本研究では単純化のため，検索語を 1 つに限定して実験を行ったが，今後は一般的な top-k 検索同様，複数の検索語による検索を実現したい．

また，実験を行う際，ノードが保有する文書について，検索語に対するスコア値の分布傾向が一定であると仮定したが，実際の様々な状況を考慮した実験を行うことで，提案手法の有効性をさらに確かなものとしたい．

参 考 文 献

1) Chaudhuri, S. and Gravano, L.: Evaluating Top-k Selection Queries, *VLDB '99: Proc. 25th International Conference on Very Large Data Bases*, pp.397–410 (1999).
2) clip2: The Gnutella Protocol Specification v0.4. http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf（参照 2008-07-10）
3) Luu, T., Klemm, F., Podnar, I., Rajman, M. and Aberer, K.: ALVIS peers: A scalable full-text peer-to-peer retrieval engine, *P2PIR '06: Proc. International Workshop on Information Retrieval in Peer-to-peer Networks*, pp.41–48 (2006).
4) Matsunami, H., Terada, T. and Nishio, S.: A Query Processing Mechanism for Top-k Query in P2P Networks, *Proc. 21st International Conference on Data Engineering Workshops*, p.1240 (2005).
5) Maymounkov, P. and Mazières, D.: Kademlia: A Peer-to-peer Information System Based on the XOR Metric, *Proc. 1st International Workshop on Peer-to-peer Systems 2002*, pp.53–65 (2002).
6) Michel, S., Triantafillou, P. and Weikum, G.: KLEE: A framework for distributed top-k query algorithms, *VLDB '05: Proc. 31st International Conference on Very Large Data Bases*, pp.637–648 (2005).
7) Ratnasamy, S., Francis, P., Handley, M., Karp, R. and Schenker, S.: A Scalable Content Addressable Network, *Proc. 2001 SIGCOMM Conference*, Vol.31, pp.161–172 (2001).
8) Rowstron, A.I.T. and Druschel, P.: Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems, *Middleware '01: Proc. IFIP/ACM International Conference on Distributed Systems Platforms*, Heidelberg, pp.329–350 (2001).
9) Stoica, I., Morris, R., Karger, D., Kaashoek, F. and Balakrishnan, H.: Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications, *Proc. 2001 SIGCOMM Conference*, Vol.31, pp.149–160 (2001).
10) Vlachou, A., Doulkeridis, C., Nørvåg, K. and Vazirgiannis, M.: On Efficient Top-k Query Processing in Highly Distributed Environments, *ACM SIGMOD/PODS Conference: Vancouver, 2008*, pp.753–764 (2008).
11) Zennström, N. and Friis, J.: Skype. http://www.skype.com/（参照 2008-07-10）
12) Zennström, N., Friis, J. and Kasesalu, P.: Kazaa. http://www.kazaa.com/（参照 2008-07-10）
13) Zhang, J. and Suel, T.: Efficient search in large textual collections with redundancy, *WWW '07: Proc. 16th International Conference on World Wide Web*, pp.411–420 (2007).

(平成 20 年 7 月 10 日受付)
(平成 20 年 10 月 7 日採録)

成毛　源樹（学生会員）

2005 年千葉大学工学部情報画像工学科卒業．2008 年東京大学大学院総合文化研究科修士課程修了．現在，東京大学大学院総合文化研究科博士課程．

山口　和紀（正会員）

1956 年生．1979 年東京大学理学部数学科卒業．1981 年東京大学理学部助手．1985 年理学博士（東京大学）．1989 年筑波大学電子情報工学系講師．1992 年東京大学教養学部助教授．1999 年東京大学情報基盤センター教授．2007 年東京大学大学院総合文化研究科教授．コンピュータのためのモデリング全般に興味を持つ．情報処理学会，ACM 各会員．