複製による負荷分散を可能にしたP2Pプロトコルの提案

A P2P Protocol with Load Balancing using Replicas: A Proposal

佐治 和弘♥ 有次 正義♦

Kazuhiro SAJI Masayoshi ARITSUGI

P2P システムに対する効率的な負荷分散は重要な問題である．BATON では，各ノードが自律的に自身の持つ値の範囲を調整し，負荷分散を実現している．しかし，システム内に人気の高いオブジェクトが存在する場合，人気オブジェクトを持つノードのオブジェクト送信負荷が高くなってしまい，BATON の負荷分散だけでは対応しきれなくなってしまう恐れがある．本稿では，BATON に SCOPE で提供されている複製の分散管理手法を組み合わせ，人気オブジェクトを持つノードのオブジェクト送信負荷を，複製を持つノードに分散する P2P プロトコルを提案，検討し，今後考えなければならない課題を明らかにする．

It is important to realize efficient load balancing mechanisms in P2P systems. BATON, a P2P system, supports a mechanism in which each node changes the range of objects to be managed autonomously. However, if there are nodes managing objects with high request rates in a network, the mechanism of BATON may fail to achieve good load balance because of the high loads for serving such objects. In this paper, we propose a P2P protocol in which we integrate a method of replica management proposed by SCOPE into BATON and discuss issues for developing a system with good load balance mechanisms.

1. はじめに

Structured P2P システムの代表的な手法として，DHT を用いた手法がいくつか提案されており[1]，Chord[2]，Pastry[3]等のシステムが開発されている．これらの DHT を用いた手法は，効率的なデータの配置方法と探索方法を提供するが，データに識別子を割り当てるために用いるハッシュ関数が似ているデータ同士にも全く異なる識別子を割り当て，ばらばらにデータを配置するため，レンジクエリをサポートするためのデータの配置には不適切である．また，ハッシュ関数を用いない場合，Chord のようなリング型オーバーレイではデータの偏りがデータの分布に大きく影響してしまうため多くのデータを保持するノードが出現してしまう恐れがある．

レンジクエリをサポート可能なオーバーレイネットワークとして，BATON[4]が挙げられる．BATON では P2P のオーバーレイネットワークの構造に平衡 2 分木を用いており，DHT を用いた手法ではサポートできなかったレンジクエリのサポートが可能となっている．また，BATON では各ノードが，自律的に自身が受け持つ値の範囲を調節し，システム内の各ノードに掛かるクエリ数，アクセス数，管理するデータ数等の負荷を均一に保つ機能を持つ．

しかし，多くのアクセスを受けるような人気オブジェクトが存在する場合，人気オブジェクトを管理するノードは過負荷状態となってしまい，値の範囲調節だけでは対応しきれなくなってしまう恐れがある．この問題の解決策の 1 つとして，人気のあるオブジェクトの複製を生成して他のノードに持たせ，負荷を分散する方法が考えられる．しかし複製が多数存在する場合を考えると，複製の管理は容易ではない．

これらの問題を解決する手法として，本稿では BATON に SCOPE[5]で提案されている複製の分散管理手法を組み合わせることで，効率的な複製の分散管理の手法と，複製を用いて人気オブジェクトの送信負荷を分散する手法を提案する．

実際に eDonkey のトラフィックを計測した文献[6]によると，download ストリームの平均サイズが 2.48M bytes なのに対し，non-download ストリームの平均サイズが 16.7K bytes と計測されており，データダウンロードに使われるデータのサイズはそれ以外のメッセージデータのサイズに比べて非常に大きく，総送信コストの大半がオブジェクトの送信コストであることがわかる．本稿では，オブジェクト送信量の多いノードが過負荷状態であると考え，過負荷状態であるノードで生じたオブジェクト送信作業のいくつかを複製を持つノードに任せることで，過負荷状態の解消を目指す．さらに，提案手法では，複製の位置情報を分散管理することで，1 つのオブジェクトの複製が多数生成された場合も，1 ノードに複製の管理コストが集中することを避けることができる．

2. 提案手法

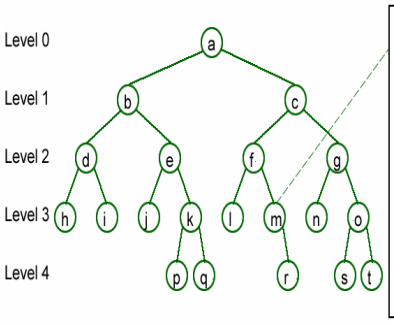
2.1 基本構造

本手法では，オーバーレイネットワークに BATON [4]の平衡 2 分木の構造を用いる．図 1 に BATON の木構造の例とノード m が持つルーティングテーブルを示す．各ノードには level と number が割り当てられる．level は木構造内でのそのノードの高さ，number は同一 level 内でのノードに左から番号を割り振ったものである．ノードの左隣，右隣のノードをそれぞれ left adjacent ノード，right adjacent ノードと呼ぶ．各ノードは親ノード，子ノード，左右の adjacent ノードへのリンクを持つ．また，各ノードは，右ルーティングテーブルと左ルーティングテーブルを持ち，level L のノードはそれぞれのテーブルに同じ level のノードを最大で L 個管理できる．

number が N のノードの右（左）ルーティングテーブルの j 番目のエントリには，number $N+2^{j-1}$（$N-2^{j-1}$）のノードへのリンクが置かれる．該当するノードがない時は，ルーティングテーブルのエントリを空にする．例えば図中のノード m の左ルーティングテーブルには，ノード m と同じlevel 3 で，number が $6-2^0=5$, $6-2^1=4$, $6-2^2=2$ のノード l, k, i へのリンクを，右ルーティングテーブルにはlevel 3 で，number が $6+2^0=7$, $6+2^1=8$ のノード n, o へのリンクを登録する．

♥ 学生会員 群馬大学大学院工学研究科情報工学専攻博士前期課程 kazuhiro@dbms.cs.gunma-u.ac.jp

♦正会員 群馬大学工学部情報工学科 aritsugi@cs.gunma-u. ac.jp



Node m: level=3, number=6
parent=f, leftchild=null, rightchild=r
leftadjacent=f, rightadjacent=r

Left routing table:

	Node	Left child	Right child	Lower bound	Upper bound
0	l	null	null	l_{lower}	l_{upper}
1	k	p	q	k_{lower}	k_{upper}
2	i	null	null	i_{lower}	i_{upper}

Right routing table:

	Node	Left child	Right child	Lower bound	Upper bound
0	n	null	null	n_{lower}	n_{upper}
1	o	s	t	o_{lower}	o_{upper}

図 1　BATON の木構造とルーティングテーブル [4]

Fig.1 An example of tree structure and routing table of BATON [4]

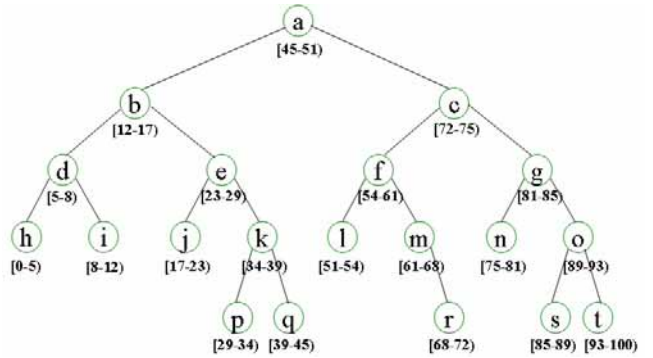


図 2　各ノードの値の範囲の例 [4]

Fig.2 An example of tree with value ranges of nodes [4]

木構造内の全ての葉ノードと内部ノードには値の範囲を割り当てる．各ノードに割り当てられた値の範囲の例を図 2 に示す．各ノードは left adjacent ノードの最大値以上から right adjacent ノードの最小値未満の値の範囲を管理する．例えば図中のノード e は，ノード j の最大値 23 以上，ノード p の最小値 29 未満の値の範囲を管理している．

2.2　SCOPE の分散管理手法

先に説明した BATON の平衡 2 分木の他に, 提案手法では複製の分散管理を行うために SCOPE[5]で用いられている RPT(replica partition tree)を使用する．3 ビットの識別子空間において，オブジェクト 101 の複製が識別子 000 のノードと識別子 111 のノードに配置された時の RPT の例を図 3 に示す．RPT は各オブジェクトに対して生成される．RPT の各ノードを代表ノード, 各代表ノードが持っている 2 ビットの値をパーティションベクトルと呼ぶ．複製の分散管理手法を，RPT の説明と共に述べる．

BATONのノードが，あるオブジェクトを自身の値の範囲に含んでいる場合，ノードはそのオブジェクトのPrimaryノードであると言う．SCOPEでは，Primaryノードが過負荷状態となることを避けるためにRPTを構築し，複製の情報を各ノードに分散して記録する． SCOPEでは識別子空間をパーティションに分割し，パーティションごとに代表ノードを決める．代表ノードは，自身が代表するパーティションをさらに分割して得られる各サブパーティションにおける複製の有無をパーティションベクトルに記録する．下位のサブパーティションも同様に分割され，最も小さなパーティションの代表ノードのパーティションベクトルは複製を持つノードの識別子を直接示す．1 回の分割で得られるパーティションの数とパーティションベクトルの長さは等しくなり，その長さは 2^nで設定される．各代表ノードが管理するパーティションの範囲を図中のSpaceに示す．代表ノードの決定方法を図 3 のオブジェクト 5=101 の代表ノードの決定方法を例として説明する．ここで，パーティションベクトルの長さは 2^1=2 である．

- Original identifier space：識別子空間の 101 にオブジェクト 5 が割り当てられる．RPT のルートノードが決まる．
- First level partition：下位 2(=3-*n*)ビットを固定し，上位 1 ビットを可変にする．この時, 001 を値の範囲に含む BATON のノードと, 101 を値の範囲に含む BATON のノードが first level partition の代表ノードとなり，RPT の中間ノードとなる．
- Second level partition：下位 1 ビットを固定し，上位 2 ビットを可変にする．この時，001，011，101，111 を値の範囲に含む BATON のノードが second level partition の代表ノードとなり, RPT の葉ノードとなる．

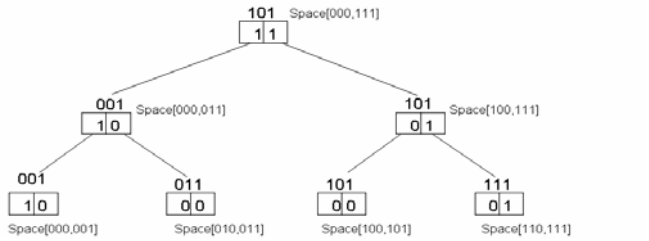


図 3　RPT の例 [5]

Fig.3 An example of RPT [5]

RPT の各代表ノードは, 下位のサブパーティションの複製の有無を，パーティションベクトルを使用して示す．下位のパーティションに複製が存在する時はパーティションベクトルの対応するビットに 1 を立て, 存在しないときはパーティションベクトルの対応するビットを 0 とする．RPT の葉ノードのパーティションベクトルは各識別子における複製の有無を直接示している．図 3 において，オブジェクトの Primary ノードであるルートノード 101 は, 下位サブパーティション 000～011 に複製が存在することをパーティションベクトルの左側のビットに 1 を立てることで示し，サブパーティション 100～111 に複製が存在することをパーティションベクトルの右側のビットに 1 を立てることで示している．

2.3　オペレーション

2.3.1　Subscribe

複製を持つノードをそのオブジェクトの Subscriber と呼ぶ．Subscribe オペレーションは，Subscriber が複製の位置を Primary ノードに知らせるために行う処理である．本稿ではオブジェクトを受信した際に，検索要求のクエリを発行したノードに複製を配置する．この手法はオーナ複製法と呼ばれており[7]，Gnutella でも用いられている方式である．Subscriber は自身の識別子の下位ビットをオリジナルデータの識別子と一致させることで, RPT の葉ノードとなる代表ノードを探す．図 3 において，ノード 010 がオブジェクト 101 の複製を持って Subscriber となった時，ノード 010 はまず, 下位 1 ビットをオブジェクト 101 の識別子と一致させて，RPT の葉ノードである代表ノードが 011 であると判断し， Subscribe オペレーションを送る．オペレーションを受け取った代表ノード 011 は, パーティションベクトルの左側

のビットに1を立てる.同様の処理をRPTのルートノード，またはノード001のような既に1の立っているパーティションベクトルを持つ代表ノードに辿り着くまで続ける．ノード010が複製を消去する時はUnsubscribeオペレーションを実行する．ノード010は代表ノード011を探索し，ノード011はパーティションベクトルの左側のビットを0にする．同様の処理をRPTのルートノード，またはパーティションベクトルに2つ以上1が立っている代表ノードに辿り着くまで続ける．

BATONでは各ノードが値の範囲を持ち，範囲は動的に変化する．本システムでは複製を持つノードの値の範囲が変化してもRPTを維持し，複製の位置を代表ノードを辿って把握できるように,複製をノードの値の1つにマッピングする．例えば図2のノードdが複製を生成した時，複製は5，6，7のいずれかの値にマッピングされる．値の範囲が変化し，ノードdの値の範囲が5〜6となり，代わりにノードiの値の範囲が8〜11から7〜11に変化した時,値7の管理はノードdからノードiに移るので，値7にマッピングされていた複製とパーティションベクトルはノードiに移動する.複製のマッピングにはSubscriberが管理する値の範囲の内からランダムに1つの値を選ぶ.

2.3.2 Exact Match Query

データvを探索するExact match queryを発行，または受け取ったノードはBATONのSearch exactアルゴリズムに従ってデータvの探索を行う．探索は，vの値を範囲内に含むノード，もしくはvの複製を持つノードに辿り着くまで行われる．探索に必要なステップ数は$O(logN)$である．

探索後，ノードはオブジェクト送信要求を発行したノードに，要求されたオブジェクトを返す．ここでPrimaryノードが過負荷状態になっている時は，オブジェクトの送信作業をSubscriberに頼む．Primaryノードはまずシステム内におけるオブジェクトの複製の有無をパーティションベクトルから確認し，複製が存在すれば下位の代表ノードにオブジェクト送信要求を転送する．要求を受けた代表ノードは，パーティションベクトルからSubscriberの位置を判断し，下位の代表ノードにオブジェクト送信要求を送る．オブジェクト送信要求はRPTを辿りSubscriberまで届けられる．代表ノードを持つBATONのノードが図3の識別子011〜101を値の範囲に持つ場合，ノードは，自身の持つ4つのパーティションベクトルの中から，ビット1が立っていて，最も下位にある，Space[100,111]の代表ノード101のパーティションベクトルを参照し，下位の代表ノード111にオブジェクト送信要求を送る．

各代表ノード間のオブジェクト送信要求の送信にはSearch exactを使って，$O(logN)$ノードを経由して送られる．RPTを辿ってSubscriberを見つけるために，Search Exactを送信する回数は平均で$O(logN)$なので，Subscriber発見のために必要となるホップ数は，平均でO（log^2N）となる．Exact match queryによる探索が，オブジェクトの複製を見つけないままPrimaryノードに辿り着いた場合，Primaryノードはオブジェクト送信要求をSubscriberに送るか，自身が要求されたオブジェクトの送信を行うか判断する．Subscriberに送信要求を送る場合，複製発見までに多くのノードを経由する必要があるので，Primaryノードが過負荷状態である場合以外では，Primaryノードがオブジェクトの送信を行うのが望ましい．本稿では，単位時間あたりのオブジェクト送信回数が閾値Aを越えたPrimaryノードは，A+1回目以降のオブジェクト送信処理のa%をSubscriberに行わせることで負荷を分散する．

2.3.3 Range Query

Range queryもExact match queryと同様の方法で処理される．まず検索する範囲を値の範囲に含むノードをExact match queryで1つ探し，そこから左右のadjacentノードを辿り，残りの検索範囲を持つノードを探す．検索範囲がX個のノードでカバーされている時，検索に必要なホップ数は$O(logN) + O(X)$である．

2.3.4 Update

更新処理は，Primaryノードがオブジェクトを更新した時，またはPrimaryノードがSubscriberからオブジェクトの更新要求を受け取った時に始まる．更新の通知はPrimaryノードからRPTを辿り，全てのSubscriberに送られる．RPTを使うことによりPrimaryノードは複製の位置の管理コストをパーティションベクトル維持のみに抑えることができ，更新情報も1レベル下位の代表ノードに送るだけで済むため，Primaryノードにかかる更新情報データの送信負荷を分散することができる．

2.3.5 Network Restructuring

ネットワークの再構築はオーバーレイネットワークの平衡木のバランスが崩れた時に行われるadjacent linkを経由してノードをシフトしていく作業であり，BATONで提供されている機能である．ネットワーク再構築の際には，ノード間のオブジェクトの移動は起きず，変更されるのはノードのlevel，number，ルーティングテーブルであり，RPTにも影響はない．

2.3.6 Join and Departure

ノードの参加・離脱はBATONのオペレーションを使用する．また，ノードの参加・離脱時にノード間で値の範囲の移動が起きた際には，移動が起こった識別子にマッピングされているオブジェクト，複製，パーティションベクトルも共に移動する．

2.3.7 Data Insertion and Deletion

オブジェクトの挿入と削除するオブジェクトの発見には，BATONのオペレーションを使用する．オブジェクトの削除が行われた時，Primaryノードはネットワーク内に存在するオブジェクトの複製を削除する必要があるので，削除命令を送信する.Primaryノードは RPTを辿り,全てのSubscriberに通知を送る．通知を受け取った各レベルの代表ノードはパーティションベクトルの値から複製の位置を判断し，下位レベルの代表ノードに削除命令を送信する．削除命令を送信後，代表ノードは削除されたオブジェクトに関するパーティションベクトルを削除する．

2.4 Node failure

ノードがFailした際には，BATON，SCOPEの修復作業を実行し，オーバーレイネットワークとRPTの修復処理を行う．Failしたノードのオブジェクトの複製が他ノードに存在する場合は，修復されたノードはRPTを辿り，オブジェクトをSubscriberノードから修復することが可能である．

3. 評価実験

提案手法の有効性を示すためにシミュレータを用いて評価実験を行い，提案手法で示した複製を用いた手法と，BATONで提案されている複製を用いない手法を比較した．0〜1023の識別子空間中に256個のノードと1000個のオブジェクトを配置した.各オブジェクトには0〜1023のいずれかの値をランダムに割り当て，オブジェクトの識別子に重複

はないものとする．以上の状態で 3000 回のSearch Exactを発生させる．オブジェクト k に対するSearch Exactオペレーションの発生確率Q_kはZipf法則[8]に従い，以下とした．

$$Q_k = \frac{k^{-\alpha}}{\sum_{m=1}^{1000} m^{-\alpha}}$$

実験ではα=1 とし，Primary ノードは 5 回目以降のオブジェクト送信処理の 75%を Subscriber に任せることとした．

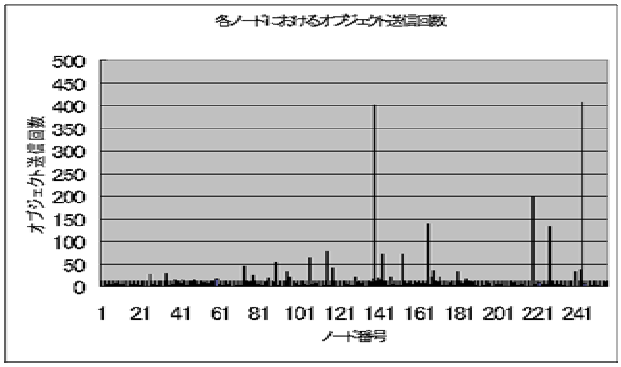


図 4　複製未使用時の各ノードのオブジェクト送信回数

Fig.4 Number of object sending in BATON

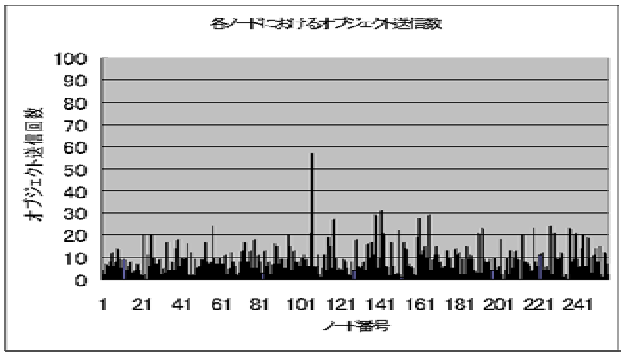


図 5　提案手法における各ノードのオブジェクト送信回数

Fig.5 Number of object sending in our proposal

シミュレーション時に各ノードが送信したオブジェクトの総数を図 4，5 に示す．図 4 は複製未使用時のオブジェクト送信回数である．人気オブジェクトを持つノードの送信回数が他に比べ大きくなっていることがわかる．これに対して本手法によるオブジェクト送信回数を示した図 5 では，ノード間のオブジェクト送信回数の偏りが解消されていることがわかる．

複製を用いることによるシステムの変化を表 1 にまとめる．提案手法で複製を利用する場合は，Primary ノードを探索してからさらに Subscriber を探しに行くため，ホップ数の最大値が増大してしまう．ただし，ホップ数の平均値は小さくなった．これは，探索の途中で複製を持つノードを偶然見つけるケースが提案手法では増えたためと思われる．複製を管理するために増加するコストを計測するために，Subscribe オペレーションのメッセージ数と，1 ノードが持つパーティションベクトル数を計測した．いうまでもないが，これらのコストは複製を利用することにより発生してしまったものである．これらの値をなるべく小さく抑えつつ，複製の分散管理を実現して，複製を利用した P2P の効果的な負荷分散を実現することは，今後の課題の 1 つである．

表 1　実験結果

Table 1 Experimental results

	提案手法	複製未使用
ホップ数平均値	4.52	4.96
ホップ数最大値	22	12
Subscribe メッセージ数合計	15645	0
Subscribe メッセージ数平均	61.11	0
パーティションベクトル数平均	24.87	0

4. おわりに

本稿では，BATON に SCOPE で提供されている複製の分散管理手法を組み合わせることで，人気オブジェクトを持つノードのオブジェクト送信負荷を，複製を持つノードに分散する P2P プロトコルを提案し，シミュレーションにより提案手法の有効性と問題点を示した．

今後の課題として，複製の配置方法の改善が挙げられる．提案手法では RPT を用いて複製の分散管理を実現できるが，RPT を使用しての複製の探索や，レンジクエリの複製によるサポートを行う時，複製を発見するまでに多くのノードを経由しなければならないことが問題として挙げられる．また，単純な複製の配置方法を含めたより詳細な比較，評価実験を行い，複製の管理に SCOPE の分散管理手法を用いることの有効性をさらに明確にすることも必要である．

[文献]

[1]H. Balakrishnan, M. F. Kaashoek, D. Karger, R. Morris and I. Stoica: “Looking up data in P2P systems”, CACM, 46, 2, pp. 43-48 (2003).

[2]I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek and H. Balakrishnan: “Chord: A scalable peer-to-peer lookup protocol for internet applications”, IEEE/ACM Trans. Networking, 11, 1, pp.17-32 (2003).

[3]A. I. T. Rowstron and P. Druschel: “Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems”, Proc. Middleware, LNCS 2218, pp.329-350 (2001).

[4]H. V. Jagadish, B. C. Ooi and Q. H. Vu: “BATON: A balanced tree structure for peer-to-peer Networks”, Proc. 31st VLDB, pp.661-672 (2005).

[5]X. Chen, S. Ren, H. Wang and X. Zhang: “SCOPE: Scalable consistency maintenance in structured P2P system”, Proc. IEEE INFOCOM, pp.1502-1513 (2005).

[6]K. Tutschku: “A measurement-based traffic profile of the eDonkey filesharing service”, Proc. PAM2004, LNCS 3015, pp.12-21 (2004).

[7]Q. Lv, P. Cao, E. Cohen, K. Li and S. Shenker: “Search and replication in unstructured peer-to-peer networks”, Proc. Supercomputing, pp.84-95 (2002).

[8]Zipf, G.K.: Human Behavior and the Principle of Least Effort, Addison-Wesley (1949).

佐治 和弘　Kazuhiro SAJI

群馬大学大学院工学研究科博士前期課程在学中．2005 群馬大学工学部情報工学科卒．日本データベース学会学生会員．

有次 正義　Masayoshi ARITSUGI

群馬大学工学部情報工学科助教授．1991九州大学工学部情報工学科卒．1996同大大学院博士後期課程了．博士（工学）．データベースシステム，分散並列データ処理等に興味を持つ．