L-012

DHT におけるノードテーブルサイズを削減するための仮想ノード配置手法

An Allocation Method of Virtual Node to Reduce Node Table Size in DHT

金子 豊† 黄 民錫† 竹内 真也† 和泉 吉則†
Yutaka KANEKO Minsok HWANG Shinya TAKEUCHI Yoshinori IZUMI

1． はじめに

大規模なキーバリューストアを実現する方法として複数のノードでキーを分散管理する DHT(Distributed Hash Table)がある．DHT の実現手段の 1 つとしてコンシステントハッシュ[1]が使われている．コンシステントハッシュでは，ノードのキー管理数を均等化するために，全ノードが固定数の仮想ノードを生成し配置する手法[2]が用いられる．しかし，全ノードが固定数の仮想ノードを生成すると，キーを検索するためのノードテーブルが大きくなる問題がある．

本報告では，ノード毎に異なる数の仮想ノードを生成し，ハッシュ空間内に配置することで，従来手法に比較してノードテーブルサイズを削減できる仮想ノード配置手法を提案する．

2．コンシステントハッシュ

コンシステントハッシュでは，円状のハッシュ空間内のノード ID の値の位置にノードを配置する．ノード ID はノードアドレスなどからハッシュ関数を使って生成する．キーバリューストアとして用いる場合には，登録するキーからハッシュ関数を使ってキーID を生成し，そのキーID がハッシュ空間内で時計回りに近いノード ID のノードがそのキーを管理する．ノードの検索にはハッシュ空間内の配置順でノード情報が格納されたノードテーブルを用いる．

コンシステントハッシュではハッシュ空間内でノードが均等に配置されることが期待されるが，実際にはノードが均等に配置されることはほとんどない．図 1 は 10 個のノードを生成した場合の実例であるが，ノードが不均等に配置されていることがわかる．このように不均等にノードが配置されると，ノードが管理するキー数が不均等になる．図 1 の場合では，ノード 9 には多くのキーが割り当てられ，ノード 1 にはほとんどキーが割り当てられない状態となる．

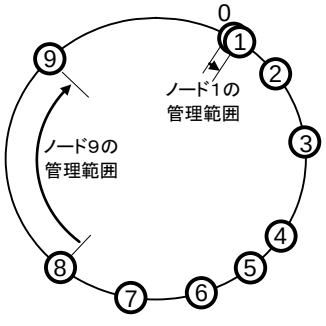


図 1 ハッシュ空間内のノードの配置例

ノード数を N，ノード ID のビット長を m とすると，ノード i の管理長 L_i は次式で表すことができる．なお，この管理長 L_i は，ノード数 N の影響を除くため，ノードがハッシュ空間内で均等に割り当てられた場合の管理長で正規化している．

$$L_i = \frac{ID_i - ID_{i-1}}{2^m / N}$$

3．従来の仮想ノード配置法

ノードの管理長を均等化する方法として，仮想ノードによる手法が用いられている[2]．各ノードはあらかじめ決められた V 個の異なるノード ID（仮想ノード）を生成し，ハッシュ空間内に配置する．各ノードが管理する範囲は，自ノードが生成した仮想ノードの管理長の合計となる．

図 2 は N＝10, 100, 1000 のときに，V=0, 10, 100 とした場合の，各ノードの管理長の標準偏差を実験により測定した結果である．実験はノード ID を変えて 10 回試行し，図ではその平均値と，最大値と最小値の範囲を示した．この実験結果から仮想ノードを増やすことで標準偏差は約 1, 0.3, 0.1 と減少し，ノードに割り当てられる管理長が均等に近づくことがわかる．また，標準偏差はノード数に関係なく仮想ノード数によって決まる傾向があることがわかる．

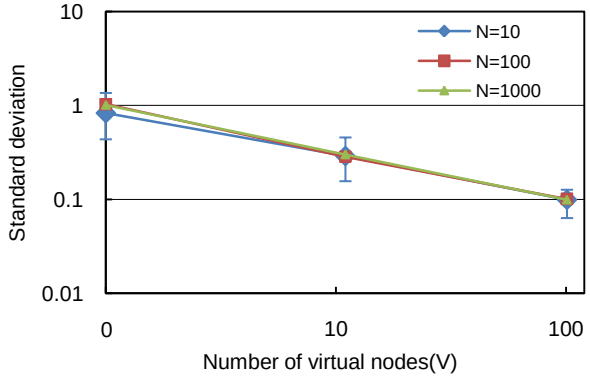


図 2 従来手法における管理長の標準偏差

4．提案手法

このように，仮想ノードの利用は管理長の均等化に有効な手段である．しかし，ノードテーブルサイズは N×(V+1)に比例することから，仮想ノードの追加によりそのサイズが増加する．ノードテーブルサイズはキー検索速度やメモリ消費量に影響するため，むやみに仮想ノードを増やすことはできない．仮想ノード数として，Chord[2]では $N=10^4 \sim 10^6$ に対して $V=\log_2 N$ が提案され，ddd[3]では N=10 程度のシステムに V=60 の運用が報告されている．

本報告では，各ノードに対して異なる数の仮想ノードを割り当てることで，従来手法に比べノードテーブルサイズを削減できる仮想ノード配置手法を提案する．提案手法のアルゴリズムを図 3 に示す．提案手法では管理長が最短のノードを求め，そのノードの仮想ノードを 1 つ増やすこと

†日本放送協会 放送技術研究所， NHK

を繰り返し行うことで仮想ノードを配置していく．仮想ノードを追加するこの処理ループを，最大管理長 L_{max} と最小管理長 L_{min} の割合 L_{max}/L_{min} がしきい値 Th 以下になるまで繰り返す．また，Th 以下に収束しない場合のために，全仮想ノード数があらかじめ決められた数 V_{max} に達した場合には処理ループを終了する．

```
loop=0
while((Lmax/Lmin>Th) && (loop < Vmax)){
   管理長最小のノードを取得
   そのノードの仮想ノードを 1 つ追加
   Lmax, Lmin を取得
   loop++
}
```

図 3　提案手法による仮想ノードの配置アルゴリズム

5. 実験結果と考察

提案手法の効果を調べるため実験を行った．ハッシュ関数に SHA-1(160bit)を用い，V_{max} は 100,000 とした．管理長 L の計算には，160bit のノード ID の上位 32bit を用い m=32 とした．各実験はノードアドレスを変えて 10 回行った．実験結果を図 4 ～ 6 に示す．各グラフは，10 回の試行の平均値と，最大値と最小値の範囲を示している．なお，今回の実験の範囲では処理ループが収束せずに仮想ノード数が V_{max} に達する場合は無かった．また，1 回の実験の計算時間は Xeon(3GHz)環境で 1 秒以内であった．

図 4 は Th=1.5, 2.0, 4.0 とした場合の管理長の標準偏差，図 5 はノードテーブルのサイズである．また，表 1 は Th=1.5 で N=10, 100, 1000 の場合の標準偏差とノードテーブルサイズである．Th=1.5 でおおよそ標準偏差が 0.1 となる．このときのテーブルサイズは従来の仮想ノード数を 10 と固定した場合のテーブルサイズとほぼ同程度のサイズとなっている．この結果から提案手法は，従来の固定的な仮想ノードの配置手法に比較して，同程度の均等化を 1/10 のノードテーブルサイズで実現できることが期待できる．

表 1　標準偏差とテーブルサイズ（平均値）

N	Std. dev.	Node table size
10	0.122	61
100	0.086	1095
1000	0.067	14348

図 6 は従来法と Th=1.5 の場合の提案手法の L_{max} と L_{min} を示している．仮想ノードを使わない場合，ノード数が増加するほど，L_{min} が 0 に近づく．これは，キーをほとんど管理しないノードが存在することを示している．一方，仮想ノードを使うことで，L_{max}, L_{min} は 1 に近づき，管理長が均等化されることがわかる．特に，従来手法ではノード数の増加に伴い，L_{min} が小さくなり均等化の性能が悪化しているのに比べ，提案手法では，ノード数が増加しても L_{min} の減少が生じず，ノード数に関係なく安定して均等化され

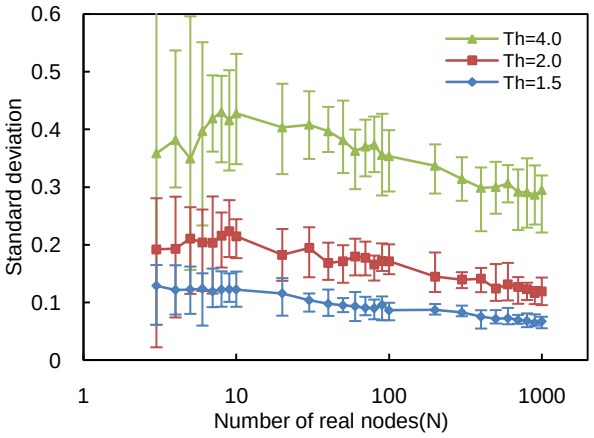


図 4　提案手法における管理長の標準偏差

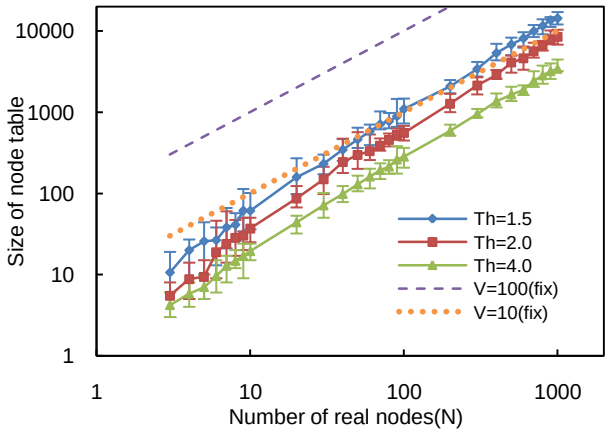


図 5　提案手法のノードテーブルサイズ

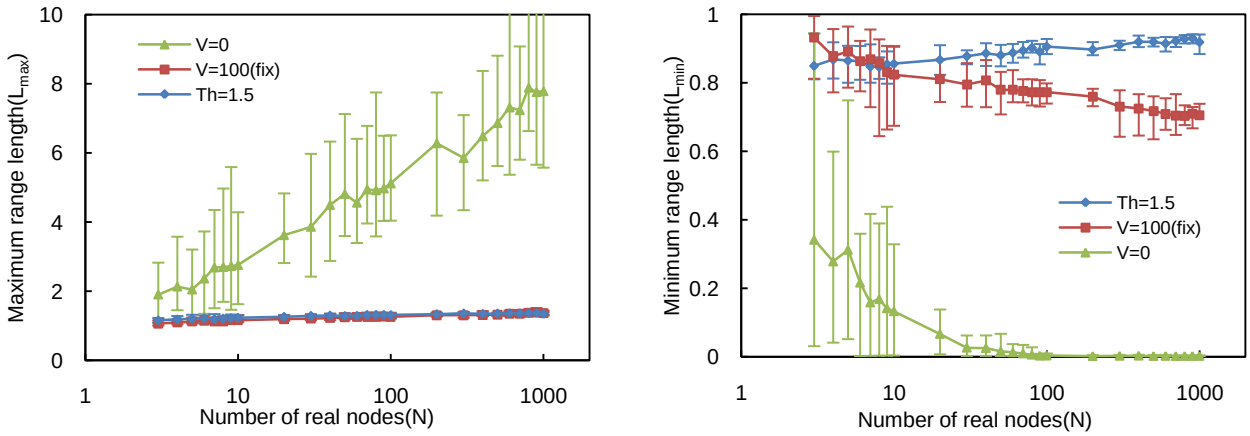


図 6　最大管理長 L_{max}（左）と最小管理長 L_{min}（右）

ていることがわかる．

図 7 は Th=1.5 で N=100, 1000 の場合のノード毎の仮想ノード数の分布について，それぞれ 10 回の実験結果を示している．N=100 の場合，90%以上のノードで仮想ノード数が 12～22 以下であり，最大でも 27 であった．N=1000 の場合には，90%以上のノードで仮想ノード数が 17～23 以下であり，最大でも 35 であった．ノード数の増加により仮想ノード数がやや増加傾向にあるが，これはノード数が増えることで所望のしきい値 Th に収束するまでにループ回数が増えるためと考えられる．表 1 から同一の Th であっても，ノード数 N が増加すると標準偏差が小さくなっていることから，同じ標準偏差で比較した場合には，仮想ノード数の分布は同程度となることが予想され，提案手法における仮想ノード数の分布は N の影響が少ないと考えられる．

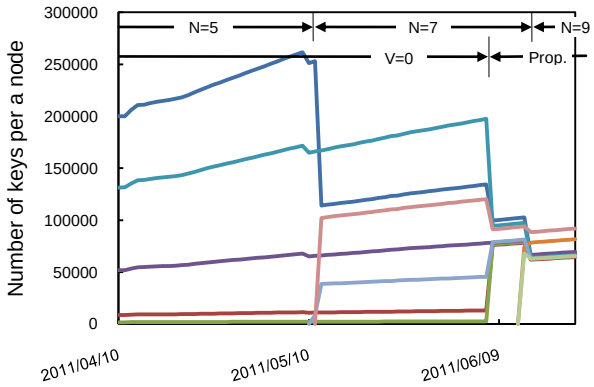
6. 分散ファイルシステムへの適用

実験的に運用している開発中の分散ファイルシステム[4]に提案手法を導入し，ノードが管理するキー数を測定した．この分散ファイルシステムは，分散して保管しているファイルの保管先アドレスの管理にコンシステントハッシュによるキーバリューストアを用いている．このシステムでは，システムに参加している全ノードを含むノードテーブルを，全ノードが共通に保持するように動作する．

図 8 に各ノードが管理しているキー数の時間による推移と代表的な時点の数値を示した．提案手法の Th は 1.5 とし，6 月 8 日に導入した．仮想ノードを使用していない期間では，キーの管理数に大きなばらつきが生じていた．5 月 1 日の時点では 243,719 個と全キーの半数近くを管理するノードと，2,160 個とほとんどキーを管理していないノードとが混在する状態となっていた．提案手法の導入後は，設定した管理長の比 Th=1.5 に対応して，ノードが管理するキー数の最大値と最小値の比も 1.5 以下の値となり，ノードの管理長に応じてキー管理数が均等化されていることがわかる．ノード数を 7 台から 9 台に変えてもこの状態が維持された．

7. まとめ

コンシステントハッシュを用いた DHT を対象に，従来手法に比べノードテーブルのサイズを減少することができる仮想ノード配置手法を提案した．実験の結果，従来の全ノードが固定的な数の仮想ノードを生成，配置する方法に比べ，同等の均等化の性能を約 1/10 のノードテーブルのサイズで得ることができることがわかった．また，提案手法を開発中の分散ファイルシステムに適用することで，キー管理の均等化に効果があることを示した．



Date	2011/05/01	2011/06/01	2011/06/10	2011/06/20
N	5	7	7	9
V	0	0	Proposed	Proposed
Node table size	5	7	25	58
Total key num.	479,901	572,134	602,095	638,362
Max key num.	243,719	190,956	100,737	91,520
Min key num.	2,160	2,579	76,868	64,157
max/min	112.8	74.0	1.3	1.4

図 8 分散ファイルシステムにおけるキー管理数の推移

参考文献

[1] D. Karger, A. Sherman, A. Berkheimer, B. Bogstad, R. Dhanidina, K. Iwamoto, B. Kim, L. Matkins, Y. Yerushalmi, “Web caching with consistent hashing,” Proceedings of the Eighth World-Wide Web Conference, 1999

[2] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, H. Balakrishnan, “Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications,” SIGCOMM’01, 2001

[3] 前橋孝広, 大原重樹, “大規模データ処理のための分散システムの実装とその応用,” 信学論 D, Vol.J93-D, No.7, pp.1072-1081, 2010

[4] 金子豊, 黄民錫, 竹内真也, 和泉吉則, “構造型 P2P を使った分散ファイルシステムにおける分散ディレクトリ管理手法,” FIT2009, L-033, pp.213-216, 2009

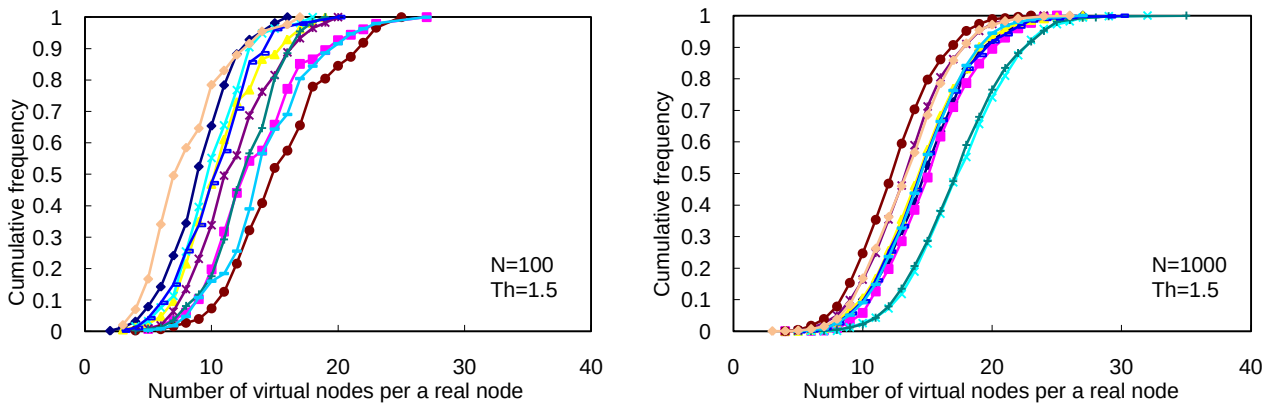


図 7 提案手法における仮想ノード数の分布