コンシステントハッシングにおける仮想ノードを用いた動的なデータ再配置方法の提案

村上 大河† 丸山 広† 高嶋 章雄† 中村 太一†

東京工科大学大学院バイオ・情報メディア研究科†

1 まえがき

Amazon Web Services などのクラウドコンピューティングベンダによって提供されるクラウドコンピューティングプラットフォームを利用することにより，利用者はトラフィックに応じてサーバの台数を増加可能なシステムを構築できるようになった．

このプラットフォームにおいて，従来より使用されているマスタースレーブ型のデータベースシステムではサーバ台数の増加に伴い，参照性能を向上させることはできるが，追加・更新性能は向上しない．そのためサーバの増加に伴いシステム追加・更新性能が向上するデータベースシステムが求められる．

この要求を満たすデータベースシステムとして，コンシステントハッシング[1]を用いた分散データベースシステムが提案されている．しかし，このデータベースシステム では各サーバに保存されるデータ数にばらつきがあるとシステム全体のスループットが向上しない．本研究では，ばらつきを抑える方法として， 仮想ノードを任意の区間に配置することによってこの問題を解決する．

2 関連技術

2.1 コンシステントハッシング

ハッシュ関数を用いて，データとサーバを同一空間にマッピングすることにより，区間ごとに担当するサーバを設定する方法である．この方法を用いると参照や挿入要求によって処理を行うサーバが特定されているため，サーバ数に応じて追加・更新性能を向上させることができる．

2.2 仮想ノード

コンシステントハッシングでは各サーバに保存されるデータ数にばらつきが出る[2]．各データに均等にアクセスがある場合，担当サーバにトラフィックの偏りが生じる．この問題を解決するために，サーバにユニークな ID を複数追加する．この追加した ID を仮想ノードと言う．仮想ノードを設定することによって，ひとつのサーバにリングの複数区間を担当させ，各サーバに保存されるデータ数のばらつきを抑えることができる．

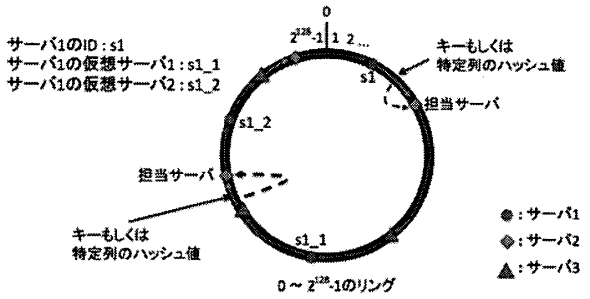

図 1：ハッシュ関数として MD5 を用いた際のコンシステントハッシングのリング

3 想定するシステム

本研究ではリクエストを受けたサーバが担当サーバの検索を行い，担当サーバがリクエストを受けたサーバ自身である場合は自分自身でリクエストを処理し，担当サーバが他のサーバである場合，リクエストを転送するシステムを想定する．また，各データに対して均等にアクセスがあると仮定する．

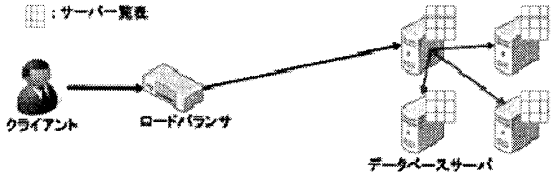

図 2：想定するシステム

3.1 サーバ 1 台当たりの 1 データベースアクセス要求の応答時間

想定したシステムにおける，1 データベースアクセスの応答時間の式を以下に示す．

$$T_{direct} = T_{HTTP1} + T_{Wait1} + T_{Search} + T_{DB} \quad \cdots(1)$$

$$T_{rdirect} = T_{HTTP1} + T_{Wait1} + T_{Search} + T_{HTTP2} + T_{Wait2} + T_{DB} \quad \cdots(2)$$

Dynamic data reallocation method using virtual-node in consistent hashing

† Taiga Murakami ・Hiroshi Maruyama・Akio Takashima ・Taichi Nakamura

Graduate School of Bionics , Computer and Media Sciences , Tokyo University of Technology

T_{HTTP1}=クライアントとデータベースサーバ間の通信
T_{HTTP2}= データベースサーバ間の通信
T_{Wait1}= リクエストを受けたサーバの待ち時間
T_{Wait2}= リクエストを転送先のサーバの待ち時間
T_{Search}= 担当サーバの検索に要する時間
T_{DB}= データベース処理に要する時間

T_{direct}は要求されたデータがリクエストを受け取ったサーバ自身であった場合，$T_{rdirect}$はデータが他のサーバにあり転送を行った場合の 1 データベースアクセス要求の応答時間である．

3.2 問題点

3.1 の(1),(2)式において，T_{DB}は一定であり，T_{HTTP1}およびT_{HTTP2}も一定であると仮定すると応答時間はT_{Search}とT_{Wait}に影響を受けることが分かる．T_{Search}は担当サーバを検索する時間なので，リングに存在するサーバ数や仮想ノード数が少ない検索時間は短く，T_{Wait1}はリクエストを受けたサーバのキューに入っているジョブ数，T_{Wait2}は担当サーバのキューに入っているジョブ数に依存する．$T_{Wait1.2}$を短くするためにはデータのばらつきを抑え，アクセス対象となる担当サーバを均等にする必要がある．このためばらつきを抑えるために仮想ノードを大量に設定すると，$T_{Wait1.2}$は減少するが，T_{Search}が増加してしまう．よって設定する仮想ノードの数を最小限にし，なおかつデータのばらつきを抑える必要がある．

表 1:仮想ノード数によるスループットの変化

物理サーバ数	仮想サーバ数	データ件数	標準偏差	実行時間(ms)	スループット(req/sec)
4	0	2,685,535	385,468	3,654,593	735
	5		60,659	3,385,209	793

4 提案手法

仮想ノードの数を最小限にし，なおかつデータのばらつきを抑える方法として，データのばらつきに応じて動的に仮想ノードをデータの集中が起こっている区間に追加しデータの再配置を行う方法について述べる．

4.1 仮想ノード ID とハッシュ値の対応表

従来の方法では仮想ノード ID は物理サーバの ID と同様にハッシュ関数によって求められていた．しかしこの方法では仮想ノード ID を任意の区間に配置することができない．そのため本研究では仮想ノードの ID とハッシュ値を対応付ける表を作成し，仮想ノードを任意の区間に配置できるようにした．

4.2 データ数リストの交換

本研究では各サーバが自分の保有しているデータ数および各区間に保存されているデータ数を記述したデータ数リストを作成し，サーバ間で交換する．

4.3 データ再配置の判断

サーバで予測されるトラフィックを処理できない場合はデータ数リストを基にデータ再配置を行いシステム全体のスループットを向上させる．データのばらつきがすでに解消されている場合は，物理的にサーバを追加する．

5 評価

予測されるトラフィックが各サーバのスループットを超えないようにするために必要な仮想ノード数を本提案手法と，従来方式である事前に仮想ノードを配置する方法とで比較を行った．

5.1 評価環境

MD5 をハッシュ関数として使用し，物理サーバを 10 台，保存するデータ 2,685,535 件のユニークな URL，各サーバのスループットを 400req/sec，予測されるトラフィックを 3600req/sec，判断を行う間隔を 30 秒としてシミュレーションを行った．

5.2 結果

図 3 に仮想ノード数と、データを最も多く保持しているサーバに予測されるトラフィックがサーバ 1 台のスループットの関係を示す．

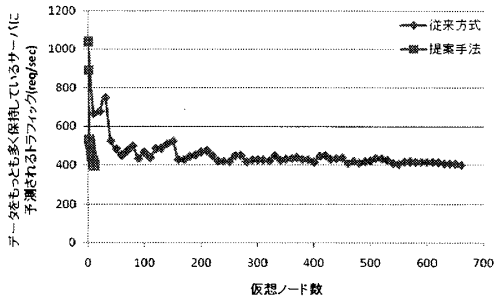

図 3:評価結果

5.3 考察

従来の方式では，事前に各物理サーバに66個の仮想ノードを設定し，全体では660個の仮想ノードがリングに存在する状態で最も多くデータを保持するサーバへのトラフィックが各サーバのスループットを下回ったのに比べ，本提案手法では仮想ノード11個で達成できた．

参考文献

[1]Karger, D., Lehman, E., Leighton, T., Panigrahy, R., Levine, M., and Lewin, D. 1997. Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the World Wide Web. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on theory of Computing*
[2]Tom White. Consistent Hashing http://weblogs.java.net/blog/2007/11/27/consistent-hashing