情報処理学会論文誌 Vol.58 No.12 1977–1992 (Dec. 2017)

多様なデータサイズ分布を持つZipf分布型処理要求に対する負荷分散とインメモリデータ・サイズ近似的最小化

山下 高生[1,a)] 栗田 弘之[1] 高田 直樹[1]

受付日 2017年1月12日, 採録日 2017年9月5日

概要：本論文では，ネットワーク装置の制御等，一定のリアルタイム性が求められる Key-Value 型データ処理における負荷分散方法を提案する．提案方法は，Key-Value 型データにおけるキーの処理要求頻度が Zipf 分布型の特性を持つこと，および，キーの処理に必要なデータが処理要求頻度の違いにより多様なサイズ分布を持つ条件での利用を前提としている．また，一定のリアルタイム性が必要な処理を実現するためのインメモリ・データベースの利用を前提としている．提案方法では，サーバ負荷の平準化，および，インメモリデータ・サイズの近似的最小化を実現する．本論文での提案方法は，これまで提案してきた Zipf 分布型要求処理のサーバ負荷平準化とインメモリデータ・サイズの近似的最小化を実現する負荷分散方法に対して，要求頻度の各ランクのデータサイズ条件を汎用化する理論的拡張を行うことで実現した．本理論的拡張では，処理要求頻度の各ランクの処理に必要なデータのサイズが，ランクの違いに対して任意の分布である条件について，既存提案方法が利用可能であることを示す．提案方法の適用評価として VPN サービスを前提とし，処理要求頻度に比例したサイズのデータを持つ条件でのシミュレーションによる評価を行った．本評価の観点は，インメモリデータ・サイズの近似的最小化，および，サーバ間のインメモリデータ・サイズの平準化である．評価の結果，広範な条件下で，負荷分散に加え，大幅なインメモリデータ・サイズの削減が可能であること，および，インメモリデータ・サイズのサーバ間のばらつきについても，サーバに必要な平均的インメモリデータ・サイズに対して十分に小さい値を実現できることを明らかにした．

キーワード：Zipf 分布，スケールアウト，ラウンドロビン，コンシステントハッシング，負荷分散

Approximate In-Memory Data Size Minimization in Load Balancing to Process Requests with Zipf Distribution of Arrival Rate and Various Data-Size Distribution

Takao Yamashita[1,a)] Hiroyuki Kurita[1] Naoki Takada[1]

Received: January 12, 2017, Accepted: September 5, 2017

Abstract: In this paper, we propose a load balancing method to process requests with Zipf distributions of request arrival rate and with various data-size distributions, which accomplishes the equalization of server load and the approximate size minimization of in-memory data. In our previous work, we proposed a load balancing method that achieves server load equalization and approximate in-memory data size minimization under the condition where the size of data item of each rank is equal. In this paper, we achieve generalizing data size distribution through further theoretical investigation of the load balancing method that we previously proposed. We evaluated the proposed method by simulation in terms of the mean and the variance of in-memory data size among servers in the case where the proposed method is applied to VPN services. The simulation showed that it can significantly decrease the size of in-memory data as well as achieve the equalization of in-memory data sizes among servers.

Keywords: Zipf distribution, scale out, round robin, consistent hashing, load balancing

1. はじめに

インターネットの活用の進展により，WWW（World Wide Web）[1]，Web 三層モデル等で用いられるデータ

[1] NTT ネットワークサービスシステム研究所
NTT Network Service Systems Laboratories, NTT Corporation, Musashino, Tokyo 180–8585, Japan
[a)] yamashita.takao@lab.ntt.co.jp

ベースシステム [2]，DNS（Domain Name Service）[3] サーバ，認証サーバや [4], [5]，一定のリアルタイム性要件を満たす必要のある VoD（Video on Demand）サーバ [6]，SIP（Session Initiation Protocol）[7] サーバ，ネットワーク機器の制御サーバ等，様々なサービスや機能がインターネット上のサーバから提供されるようになってきている．

このように様々な機能を汎用の計算機によって実現する際に，必要に応じて各機能に計算機資源を割り当てていくことでコスト効果を高めることのできる，クラウド・コンピューティング技術 [8] や NFV（Network Function Virtualization）技術 [9] の検討や商用利用が近年進みつつある．これらの技術では，機能ごとに計算機資源を準備するのではなく，様々な機能のための資源を全体として準備し，それをユーザ数や利用状況の変動に応じて割り当てる．このようなサーバの処理能力を柔軟に変化させることができるスケール柔軟性は，サービスのコスト効果の観点から重要である．スケール柔軟性を実現するための要素技術としては，1 つの計算機の資源を分割し OS レベル等で複数の計算機に見せる仮想化技術 [10] や，複数のサーバがフラットな関係で協調し，並列分散処理により大規模な処理能力を実現するスケールアウト技術 [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22] があり，研究開発や普及が進んでいる．

一方で，インターネット上で提供されるアプリケーションとしてのサービスや，ネットワークサービスを提供するためのサーバ処理に関して，その処理要求特性は様々である．その中で Zipf 分布 [23] と呼ばれる一部のデータを用いた処理要求が大半を占める一方で，この一部のデータの量と比較して，処理要求頻度の低い処理に用いられるデータが大量に存在する処理要求特性が存在する．Zipf 分布は，Web キャッシュやマルチメディアサーバのキャッシュ処理等，広範囲の利用領域において報告され [24], [25], [26]，重要な処理要求特性となっている．そのことにより，Zipf 分布の処理要求に対する分散処理に関しては，様々なシステム構成において検討されてきている [17], [27], [28], [29], [30]．

これらの中で，クラウド・コンピューティング環境での Zipf 分布に従う要求処理については，データがアクセスされる頻度のランクを基にしたキャッシュ置換方法 [27] やコスト・性能の制約の中でサーバ資源の管理を行うキャッシュ方法 [17] 等が提案されている．これらの方法はキャッシュ方式を用いているため，全サーバが同一のデータを利用可能であること，サーバ起動時とキャッシュに十分なデータが格納されたときの性能差の存在が，サービスや運用に大きな影響を及ぼさない条件で利用可能である．一方で，これらの既存技術を用いた場合，故障対応や OS/アプリケーション更新等のサーバ停止・再起動によってキャッシュヒット率が変化する．そのことにより，サーバの応答時間の増加，および，性能低下が発生する．そのため，上述のようにリアルタイム性が求められる場合や，性能低下によって，サービス品質・運用に大きな影響を与える利用条件では問題が発生してしまう．ここで指摘した問題については，2 章で説明する提案技術の適用例をもとに 7 章で詳細に説明する．

既存技術のようなキャッシュ方式を用いた場合の問題を解決し，一定のリアルタイム性のある処理を実現するためには，インメモリ・データベース等，高速の記憶デバイスを用いることが効果的な場合がある．その場合のサーバ台数を考えると，たとえば商用のネットワークサービスの場合，ネットワーク装置の制御に必要なデータがメモリ等，非常に高速の記憶装置に格納されていてもユーザ利用のログ蓄積等，一定程度の処理負荷がサーバにかかる．このことにより，商用利用においては，高速な記憶デバイスを用いても，一定以上台数のサーバが必要となってくる．そのため，このようなケースでもサーバ間の負荷分散を行う分散処理技術は不可欠である．

我々は，これまで Key-Value 型データのキーの要求頻度が Zipf 分布に従う条件下における，参照操作が中心のデータ処理に適した負荷分散方法を提案してきた [32], [33]．本既存提案では，サーバの処理に一定のリアルタイム性が求められる要件を満たすため，Key-Value 型データの管理にインメモリ・データベースを利用することを前提としている．さらに，大量のデータが存在する条件下での，メモリの有効利用のため，コンシステントハッシング技術を用いることを前提としてきた．これは，大量のデータをサーバ全体として保持できるよう全データを複数のサーバに分割して保持しつつ処理要求の負荷分散を行うことを実現できるためである．コンシステントハッシングでは，各キーに対する要求頻度が極端に変わらない場合には，大小様々な要求頻度のキーをサーバが受け持つことで平均的にサーバ負荷が分散される．一方で，極端に要求頻度の大きいキーが存在すると，そのキーを担当するサーバが少数のキーの処理で性能を使い切る状況や，あるキーの処理を単体のサーバでは処理しきれない状況が発生する．これらの状況の場合，コンシステントハッシングでのメモリの有効利用ができないことや，処理できないキーが発生することになる．この要求頻度の高いキーの処理の存在に対し，既存提案方法では，まず，要求頻度を 3 つの領域に分割した．そして，3 つの領域の処理について，データ複製，データ複製とコンシステントハッシングの組合せ，コンシステントハッシングを使い分ける負荷分散方式を提案した．このことにより，要求頻度の高いキーの処理の負荷分散とメモリの有効利用を実現した．さらに，上記 3 つの要求頻度の領域の境界を決定する方法を提案することで，サーバ間の負荷の平準化に加えインメモリデータ・サイズの近似的最小化を実現した．この既存提案方法は，Key-Value 型データのキーごとの各データのサイズの分布特性としては，デー

タのサイズが，平均的には一定として扱えることを前提としている．

一方で，キーごとの各データのサイズについては，多様な分布が存在する．たとえば，**図 1** のように，ユーザを様々な組織のプライベート・ネットワーク（PN: Private Network）につなぐための Remote Access VPN（Virtual Private Network）サービスを大規模通信事業者が行う場合を考える．この例では，通信事業者は，ユーザの少ない小規模の PN からユーザ数のきわめて多い大規模の PN まで，様々な規模の PN に対してサービスを行う．その中で通信事業者は，ユーザが指定する PN の識別子をキーとして，ユーザが行う通信を指定された PN に接続する制御を行わなければならない．本例では，PN のユーザ数が増加するほど，その PN に接続するための要求の頻度が増加すると考えられる．それに加え，通常 PN へのアクセス制御を行う VPN Gateway が収容可能なユーザ数は，ハードウェアの性能等により上限がある．そのため，ユーザ数が増加するとユーザを収容するための VPN Gateway の数が増える．VPN Gateway の増加は，通信事業者が PN ごとに管理するデータのサイズの増加につながる．これに対し我々がこれまで提案してきた負荷分散方法は，上述のようにキーごとのデータのサイズが，平均的には大よそ一定として扱えることを前提にしているため，本例のような条件下では利用が困難である．本例については，より詳細に 2 章で説明する．

本論文では，これまで提案してきた，Zipf 分布型要求処理のサーバ負荷平準化とインメモリデータ・サイズの近似的最小化を実現する負荷分散方法について，適用可能な条件の拡張を行う．本拡張は，各キーに割り当てられたデータのサイズの分布に関する条件の拡張である．既存提案方法の適用条件は，各キーに割り当てられたデータのサイズが平均的には一定であることであった．これに対して，既存提案方法の理論的拡張を行うことにより，任意のデータサイズ分布で既存提案方法が利用可能であることを示す．そのことにより，既存提案方法が，多様なデータサイズ分布を持つ Zipf 分布型処理要求に対する負荷分散方法として利用可能であることになる．さらに，提案方法に対し，

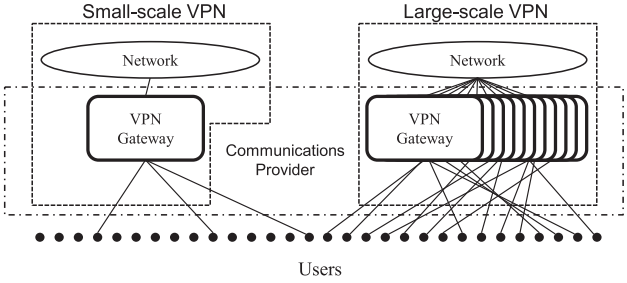


図 1 通信事業者により提供される VPN サービス例

Fig. 1 Example of VPN service provided by communications provider.

前パラグラフで説明した大規模通信事業者の行う Remote Access VPN サービスを例とし，キーの処理要求頻度に応じて増加するサイズのデータを持つ条件でシミュレーションによる評価を行う．本評価の中では，インメモリデータ・サイズの近似的最小化，および，サーバ間のインメモリデータ・サイズの平準化ついて評価結果を議論する．その中で，最初に，広範な条件下で，負荷の平準化，および，大幅なインメモリデータ・サイズの削減が可能であることを示す．さらに，インメモリデータ・サイズのサーバ間のばらつきについても，十分に小さなレベルを実現できることを明らかにする．

本論文の構成について述べる．2 章では，本章で導入した本論文の提案方法の応用対象について詳細に説明する．3 章では，既存提案方法である，Zipf 分布型処理要求のための負荷分散方法について，その概要を説明する．4 章では，既存提案方法における負荷分散を実現するための条件について説明する．5 章では，まず，4 章で説明した負荷分散を実現するための条件を満たす範囲の中で，任意のデータサイズ分布におけるインメモリデータ・サイズの変化の理論的解析を行う．次に，本解析をもとに，既存提案方法によって，任意のデータサイズ分布でインメモリデータ・サイズの近似的最小化方法が可能であることを理論的に示す．6 章では，提案方法に対して，シミュレーションによる評価と議論を行う．評価項目は，インメモリデータ・サイズの近似的最小化，および，サーバ間のインメモリデータ・サイズの平準化である．7 章では，関連研究について説明する．8 章では，本論文のまとめについて述べる．

2. 応用対象例と計算資源平準化の必要性

2.1 本研究の応用対象例

VPN は，インターネットや閉域網等の共用されたネットワークを経由してプライベート・ネットワーク（PN: Private Network）を実現する技術である．VPN の利用形態としては，**図 2** に示すように，(a) Site-to-Site VPN と (b) User-to-LAN（Remote Access）VPN の二形態が広く利用されている．Site-to-Site VPN の例としては，企業の複数の拠点を接続して PN を構築するため等に用いられる．また，User-to-LAN VPN の例としては，企業が社員に対

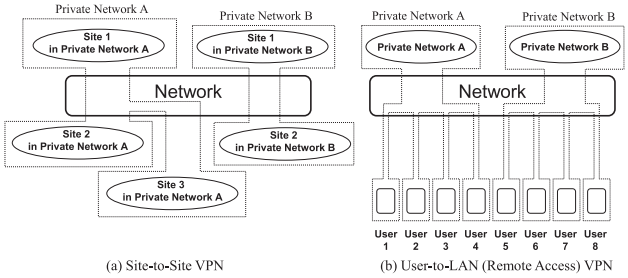


図 2 VPN の利用形態

Fig. 2 Implementation of VPN networks.

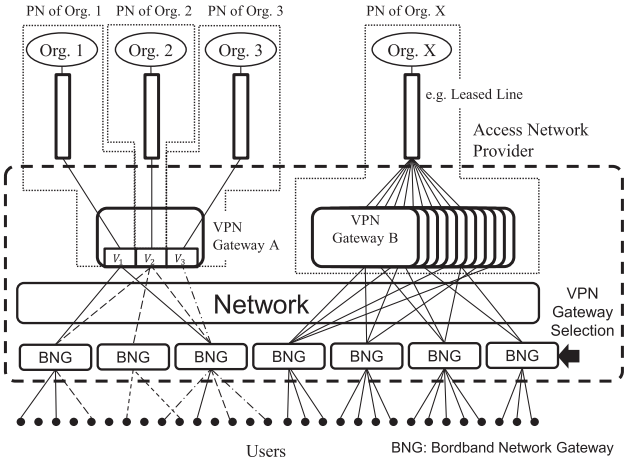


図 3 VPN サービスの構成例

Fig. 3 Network architecture example of VPN service provided by communications provider.

して，遠隔で業務を行うためのリモート・アクセスの手段を提供するため等に用いられる．このような目的の場合，企業は IPsec コンセントレータ等の VPN Gateway を設置し，社員は遠隔から，この VPN Gateway に接続して企業の PN にアクセスする．

本研究の応用対象例として，User-to-LAN VPN において通信事業者が VPN Gateway を運用し，多数の PN を持つ組織が，これを利用する形態を想定している．ここで，ブロードバンド・サービスを提供するための標準化を行っている Broadband Forum [31] で策定されたネットワーク・モデルの中で，アクセス網を提供する通信事業者が上記の User-to-LAN VPN のサービスを提供する形態の例を，**図 3** を用いて説明する．

Broadband Forum のモデルでは，Broadband Network Gateway（BNG）と呼ばれる装置で，ユーザを収容する．この装置でブロードバンド・サービスとしてのユーザが特定される．このようなネットワークを用いて上記の User-to-LAN VPN を提供する場合，図 3 で示したように，PN ごとに VPN Gateway を利用して PN の利用者としてのユーザを認証を行う必要がある．その後，認証に成功したユーザのパケットは，まとめて PN に転送される．

通常，VPN Gateway と呼ばれる装置には，パケットの転送性能，暗号化機能の性能，認証の処理性能等によって，収容可能なユーザ数の上限が存在する．そのため，各 VPN について，図 3 に示したように，PN に収容するユーザ数に比例した数の VPN Gateway を設置する必要がある．また，ユーザ数が小規模であるプライベート・ネットワークを効率的に利用可能とするには，図 3 に示したように，1 つの VPN Gateway によって，複数の PN を収容することが必要となる．

今，ブロードバンド・サービスとしてのユーザが，ある PN に接続する場合，通信事業者が提供するブロードバンド・ネットワーク側に，ユーザを適切な VPN Gateway に導く VPN Gateway 選択の機能（以降，VPN Gateway 選択機能と呼ぶ）が必要となる．VPN Gateway 選択機能は，ユーザが接続しようとしている PN を特定する識別子（以降，PN-ID と呼ぶ）を元に，適切な VPN Gateway を選択して，BNG 等のネットワーク装置やユーザ端末に指示する機能である．本機能の実現には，PN-ID をキーとして，そのキーに関連付けられた VPN Gateway の IP アドレス等の識別情報や VPN Gateway 等の通信装置を制御するためのデータを管理する必要がある．

ここで，本研究の応用先として上記の VPN Gateway 選択機能への適用を考える．VPN Gateway 選択機能は，ユーザが，ある PN-ID で識別される PN に接続要求を行うときに利用される．今，1 人のユーザが PN への接続要求を行う頻度を，PN によらず一定と仮定する．この場合，各プライベート・ネットワークのユーザ数に比例した頻度で，PN-ID をキーとした VPN Gateway 選択機能の処理が発生する．また，上述のように，各 PN について，PN に収容するユーザ数に比例した数の VPN Gateway が必要となることから，VPN Gateway 選択機能の処理で用いられる PN ごとのデータ量は，ユーザ数に比例して増加することになる．このような場合，VPN Gateway 選択機能の処理が発生する頻度，および，VPN 選択機能の処理で用いられる PN ごとのデータ量の双方が，該当する PN のユーザ数に比例して増加することになる．本研究の応用対象として，本 VPN Gateway 選択機能におけるデータ処理を想定し，その条件のもとに 6 章において評価を行う．

2.2 計算機資源平準化の必要性

2.1 節で説明した VPN Gateway 選択機能のように，大量の処理要求を汎用計算機上で動作するサーバによって行う場合，複数のサーバで処理を行うことによりスケール性を確保する必要がある．さらに，複数のサーバ処理において，個々のサーバの計算性能やストレージ量の計算資源を有効活用することは，コスト低減のために重要である．

複数のサーバで処理のスケール性を確保する場合，それらのサーバにおいて実際に使われる計算性能の平準化を行う負荷分散方法があるとする．N_s 台のサーバを準備したとき，その負荷分散方法によって 1 台のサーバに最大で L_u の負荷がかかると想定する．また，負荷分散方法が，サーバで使われる計算性能を完全には平準化できないとし，1 台のサーバの負荷が $L_u(1-q)$ から L_u の範囲にあるとする．ここで，q は負荷が変動する割合とする．次に，1 台のサーバで一定の応答時間で処理可能な負荷の上限を L_s とする．各サーバの負荷は，L_s 以下でなければならないため，$L_u \leq L_s$ となる．一方で，実際に負荷分散方法のもとで各サーバにかかる負荷の範囲は $L_u(1-q)$ から L_u であるため，確実に利用される計算性能に起因する負荷は，全サー

バのトータルで $N_sL_u(1-q)$ となる．すなわち，$L_u = L_s$ の状態で使用したとしても最大で qN_sL_s の負荷が計算性能として使用されていない可能性があり，q が大きいほど，無駄が発生する．そのことで，結果的に，サーバ数の増加が必要となる．

VPN Gateway 選択機能の処理は，ネットワーク接続を行うためのものであり，一定以下の短時間で完了する必要がある．そのため，本研究の前提として，VPN Gateway 選択機能の処理に必要なデータをメモリ上に配置することを前提としている．この場合，前パラグラフの議論の各サーバの計算性能と同様に，サーバごとのメモリ量にも上限がある．そのため，計算性能の平準化の議論と同様の理由で，サーバ数を削減するためには，利用されるメモリ量の平準化が必要である．

以上の議論から，サーバ台数を減少させるため，計算性能，および，メモリによるストレージ量の双方において，計算資源の平準化が不可欠である．

3. Zipf 分布型処理要求のための負荷分散方法の概要

3.1 ランクの分割と負荷分散方法

本論文では，前章で述べたように Zipf 分布の処理要求を扱う．Zipf 分布とは，発生頻度の高い事象から低い事象に発生頻度と事象の組を並べ，発生頻度の順位をランクと呼んだとき，ランク r の発生頻度 $f(r)$ が下記の式に従う分布である．

$$f(r) = \frac{A}{r^p} \tag{1}$$

ここで，A と p は定数であり，本論文では，p を Zipf 定数と呼ぶこととする．本分布は，p の値が大きいほど，r の値の小さいランク（上位のランク）への処理要求の偏りが大きくなる特徴を持つ．

今，発生する事象を，あるキーを含む処理要求が，クライアントからサーバに到着することと定義したとき，各キーの到着頻度は**図 4** に示した曲線のようになる（以降，キーの到着頻度を，キーの要求頻度と呼ぶ）．本図において，横軸が，あるキーを含む処理要求がサーバに到着する事象が発生するランク（以降，キーのランクと呼ぶ）であり，縦軸が，キーの要求頻度である．以降，キーの要求頻度の値の大きいランクを，高いランクと表現する．

既存提案方法 [32], [33] では，キーの要求頻度のランク範囲を，3 つの範囲に分割する．これらの範囲は，要求頻度の高い領域，中間の領域，低い領域の 3 つの領域であり，それぞれ，図 4 に示すように，ホットゾーン（Hot zone），ノーマルゾーン（Normal zone），および，コールドゾーン（Cold zone）と呼ぶ．ここで，ノーマルゾーン，および，コールドゾーンの最も要求頻度の高いランクを，それぞれ，R_f (≥ 1)，および，R_l (≥ 1) と表すこととする．

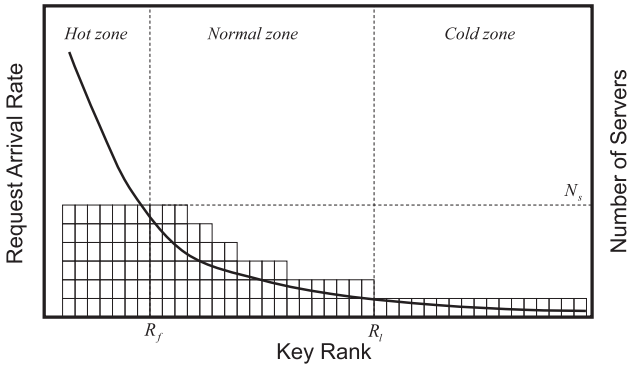


図 4 Zipf 分布と負荷分散方法のためのゾーン分割
Fig. 4 Zipf distribution and three zones for selecting one of three load balancing methods.

上記のランク全体を 3 つに分割した各ゾーンについて，ホットゾーンに属するランクのキーに対する処理要求に対しては，全サーバによるラウンドロビン処理によって負荷分散を実現する．このとき，ホットゾーンを処理するデータは，全サーバにデータ複製される．また，コールドゾーンに属するランクのキーに対する処理要求に対しては，全サーバを用いたコンシステントハッシングによる処理を行うことで負荷分散を実現する．ノーマルゾーンに属するランクのキーに対する処理要求については，ラウンドロビンとコンシステントハッシングの組合せによって処理を行う．この中で，各ランクを処理するサーバの選択はコンシステントハッシングをベースにした方法で行う．このことにより，それぞれのサーバが処理を担当するキーを全サーバに分散させる．選択されるサーバの台数は，全サーバ台数未満かつ 2 台以上である．選択された複数サーバに対して，そのランクを処理するためのデータが複製される．各キーに対する処理は，選択された複数のサーバよってラウンドロビン処理を行うことで負荷分散を実現する．

3.2 負荷分散方法に関する定義

次節以降の議論で必要な負荷分散の定義について説明する．まず，N_s，および，N_r を，それぞれ，全サーバの台数，および，全キー数とする．各キーを処理するために用いるサーバ台数を，分散度と呼ぶものとすると，ランク r の分散度 $d(r)$ を，N_s，および，N_r を用いて下記のように定義する．下記の式において，一行目の $1 \leq r \leq R_f - 1$ の範囲，二行目の $R_f \leq r \leq R_l - 1$ の範囲，および，三行目の $R_l \leq r \leq N_r$ の範囲は，それぞれ，ホットゾーン，ノーマルゾーン，および，コールドゾーンの範囲である．

$$d(r) = \begin{cases} N_s & (1 \leq r \leq R_f - 1) \\ g(r) = \lceil \frac{f(r)}{f(R_l)} \rceil & (R_f \leq r \leq R_l - 1) \\ 1 & (R_l \leq r \leq N_r) \end{cases} \tag{2}$$

ここで，ランク R_l のキーの要求頻度を，基本要求頻度と呼び，ノーマルゾーンに含まれるランクのキーの分散度は，

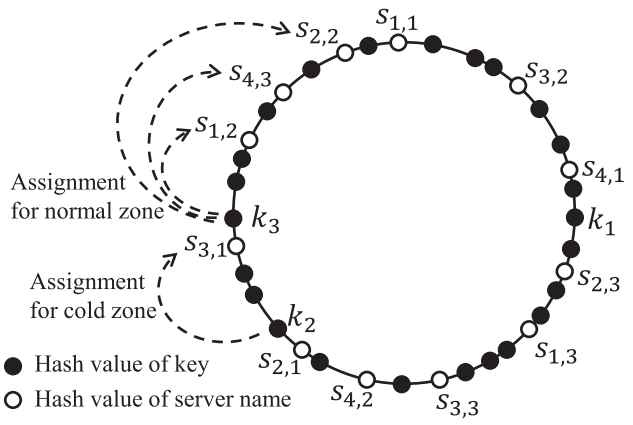


図 5 コンシステントハッシングによるサーバ決定方法
Fig. 5 Server assignment to keys in consistent hashing.

この基本要求頻度をもとに決定する．上式の中で用いている $\lceil x \rceil$ は，x 以上の最小の整数を表す．また，ノーマルゾーンの中のホットゾーンとの境界である R_f も，基本要求頻度 R_l によって一意に決定され，R_f は，$g(R_f - 1) \geq N_s$，かつ，$g(R_f) < N_s$ となる整数として定める（$g(r)$ は式 (2) に示すように R_l をパラメータとして含むことから，R_f は R_l によって決まることになる）．

図 4 の中に描いた四角形の 1 つ 1 つは，それぞれが，各ランクのキーを処理するサーバが 1 台存在することを意味しており，各ランクの四角形の数が分散度を表す．また，この四角形の数を一定のランク範囲にまたがって合計した数を，サーバ割当て数と呼ぶこととする．本図は，ホットゾーンとコールドゾーンのサーバ台数が一定であり，ノーマルゾーンでは，ホットゾーンとの境界からコールドゾーンとの境界に向けて階段状に減少する様子を表している．

次に，コールドゾーン，および，ノーマルゾーンの範囲にある，キー k を処理するサーバの決定方法について説明する．まず，**図 5** を用いてコンシステントハッシングのサーバの決定方法を説明する．本図は，キーのハッシュ値の範囲を，上限と下限を接続することでリング状に表している．黒丸，および，白丸は，それぞれ，キー，および，サーバ名のハッシュ値を表している．サーバは，複数の名前を持っており，$s_{x,y}$ は，サーバ x の y 番目の名前のハッシュ値である．あるキーを処理するサーバは，そのキーのハッシュ値を表す黒丸から時計方向に移動していき最初の白丸に該当するサーバが処理を行う．コールドゾーンの範囲にあるキーを処理するサーバの決定は，このコンシステントハッシングのサーバの決定方法を用いる．図 5 におけるキー k_2 の例の場合，処理を行うサーバは $s_{3,1}$ となる．

一方，提案方法のノーマルゾーンの処理では，あるキーに対して複数のサーバを割り当てる．そのため，コンシステントハッシングのサーバ決定方法を拡張して次のようにサーバを決定する．ランク r のキーのハッシュ値を表す黒丸から，時計方向に移動していき，1 番目の白丸から $d(r)$ 番目の白丸に該当するサーバをランク r のキーの処理に割り当てる．図 5 におけるキー k_3 の例の場合，キー k_3 に該当するランクの分散度を 3 とすると，キー k_3 を処理するサーバは $s_{1,2}$，$s_{4,3}$，および，$s_{2,2}$ となる．

4. 負荷分散条件

4.1 概要

前章で説明したように，既存提案方法においては，Zipf 分布である $f(r)$ が式 (1) で与えられたとき，ノーマルゾーン，および，コールドゾーンの最も要求頻度の高いランクである，R_f，および，R_l に応じて，各ランクを処理するためのサーバ台数である分散度が，式 (2) のように決定される．このことにより，サーバ間の負荷分散のレベル，および，各サーバに必要な，インメモリデータのサイズが変化する．ここで，負荷分散のレベルは，サーバへの要求処理に起因して発生するサーバ負荷の平均値に対してのばらつきを意味する．より正確な表現としては，以降の議論で，サーバ負荷のサーバ全体での平均値に対する標準偏差の割合（変動係数）として評価・議論する．

各ゾーンにおける負荷分散のレベルについて考えると，ホットゾーンに含まれるキーに対する処理要求は，全サーバを用いたラウンドロビン処理により，すべてのサーバの間で負荷分散が実現される．ホットゾーンの各キーの処理による負荷は，このラウンドロビンによって，全サーバに対し均等に振り分けられる．このとき，一定時間以上の各サーバの負荷の平均は，理想的には同一となる．

コールドゾーンについては，Zipf 分布はランクが低くなるにつれて，ランク変化に対して要求頻度の変化が小さくなり，ランク間の要求頻度のばらつきが少なくなる．よって，一定のランク以上にコールドゾーンを設定することで，サーバ負荷のばらつきを表す負荷分散のレベルを確率的に改善できる．本条件について，4.2 節で説明する．

ノーマルゾーンについては，比較的要求頻度が高いため，サーバ全体にどのように負荷を与えるかは，ノーマルゾーンの領域をどのように決めるかによって，すなわち，R_f と R_l をどのように決めるかによって変化することになる．ノーマルゾーンの負荷分散を実現可能な条件については，4.3 節で述べる．

4.2 コールドゾーンの負荷分散の条件

コールドゾーンに含まれるランクのキーは，3 章で説明したように，キーのハッシュ値をもとに，確率的に N_s 台のサーバに割り振られる．コールドゾーンに含まれるランクは，R_l 以上，N_r 以下の $N_r - R_l + 1$ 個のランクである．全ランクの要求頻度の平均 M_a，および，コールドゾーンに含まれるランクのキーの要求頻度のばらつきを表す分散 $\sigma_c^2(R_l)$ は，以下の式で表される．

$$M_a = \frac{1}{N_r}\sum_{r=1}^{N_r} f(r) \tag{3}$$

$$\sigma_c^2(R_l) = \frac{\sum_{r=R_l}^{N_r} f^2(r)}{N_r - R_l + 1} - \left(\frac{\sum_{r=R_l}^{N_r} f(r)}{N_r - R_l + 1}\right)^2 \tag{4}$$

コールドゾーンに含まれるランクのキーは，平均的に各サーバに対して，$(N_r - R_l + 1)/N_s$ 個割り当てられることになる．ここで，$N_c = (N_r - R_l + 1)/N_s$ 個とすると，仮にランクが無限標本と仮定すると，中心極限定理 [34] により，各サーバに割り当てられた要求頻度の和の平均値は，$N_c M_a$ であり，コールドゾーンによって発生する分散（ホットゾーンとノーマルゾーンの分散がないと見なしたときの分散）は，$N_c\sigma_c^2(R_l)$ となる．よって，コールドゾーンのランクをサーバに割り当てたとき，標準偏差である $\sqrt{N_c}\sigma_c(R_l)$ が，平均値 $N_c M_a$ と比較して十分小さければ，負荷分散を実現できることになる．実際には，ランクは有限標本であるため，厳密に中心極限定理には従わない．しかし，サーバ台数 N_s に対して，ランクの数が十分に大きな値であれば，実質的に，標本が有限であることの影響が相対的に弱くなると考えられる．そのため，平均値 $N_c M_a$ に対する標準偏差 $\sqrt{N_c}\sigma_c(R_l)$ の割合は，サーバ負荷の平均値に対するコールドゾーンに起因するサーバ負荷の標準偏差の割合に近くなると考えられる．4.1 節の第 1 パラグラフで述べたように，本論文では，負荷分散のレベルを，サーバ負荷のサーバ全体での平均値に対する標準偏差の割合として評価・議論する．これまでの議論により，平均値 $N_c M_a$ に対する標準偏差 $\sqrt{N_c}\sigma_c(R_l)$ の割合は，負荷分散のレベルに近似的に近くなると考えられることから，負荷分散のレベルを表す指標として用いることとする．より具体的には，この指標値が一定以下になることをコールドゾーンの負荷分散の条件として採用し，R_l が負荷分散を満たす条件として下記を用いることとする．

$$R_l \geq \hat{R}_l \tag{5}$$

ここで，$\hat{R}_l$ は，下記の条件を満たす最小のランク R_l であり，ε は，1 と比較して十分に小さい値を用いる．

$$\sqrt{N_c}\sigma_c(R_l)/N_c M_a = \sigma_c(R_l)/\sqrt{N_c}M_a \leq \varepsilon \tag{6}$$

ここまでに使用した記号，および，以下で使用する記号のうち，主要なものを**表 1** に示す．表中の式のカラムは，記号が最初に使われている式を表している．

4.3 ノーマルゾーンの負荷分散の条件

4.1 節で説明したように，ノーマルゾーンの領域の決め方により，ノーマルゾーンによる負荷分散への影響が決まる．ノーマルゾーンに属するキーの処理の負荷分散は，3 章で説明した図 4 に描かれている要求処理のサーバ割当てを表す四角形が，全サーバにどのように分散されるかによって決まる．負荷分散を実現するためには，この四角形が全サーバに均等に割り当てられることが必要となり，そのためには，全サーバ台数に対して十分に大きな数のサーバ割当て数となる必要がある．いい換えると，ノーマルゾーンに属する各キーを処理するためのサーバ台数である分散度の総和であるサーバ割当て数を，全サーバ台数と比較して十分に大きくする必要がある．

表 1 主要な記号の説明
Table 1 Description of important symbols.

記号	説明	式
$f(r)$	ランク r のキーの要求（発生）頻度	(1)
p	Zipf 定数	(1)
R_l	コールドゾーンの最も要求頻度の高いランク	(2)
R_f	ノーマルゾーンの最も要求頻度の高いランク．R_f は，$g(R_f - 1) \geq N_s$，かつ，$g(R_f) < N_s$ となる整数であり，基本要求頻度 R_l によって一意に決定される．	(2)
$d(r)$	ランク r の分散度	(2)
N_s	全サーバ台数	(2)
N_r	全キー数	(2)
$g(r)$	ノーマルゾーンの分散度	(2)
$\hat{R}_l$	コールドゾーンの負荷分散のレベルを一定以上にするための R_l の下限値	(5), (6)
R_e	単一のサーバで処理できる要求頻度に該当するランク．ここで，R_e は，0 を超える実数とし，単一のサーバで処理できる要求頻度は，$f(R_e)$ で表される．	(21)

本条件として，ノーマルゾーンのサーバ割当て数の近似的な下限が，サーバ数 N_s の K 倍となる条件を求めると，下記となる．

$$\sum_{r=R_f}^{R_l-1}\lceil g(r)\rceil = \sum_{r=R_f}^{R_l-1}\left\lceil\frac{f(r)}{f(R_l)}\right\rceil = \sum_{r=R_f}^{R_l-1}\left\lceil\frac{R_l^p}{r^p}\right\rceil$$
$$\geq \int_{r=R_f}^{R_l}\frac{R_l^p}{r^p}dr = \begin{cases}\frac{R_l - R_l^p R_f^{1-p}}{1-p} \geq KN_s & (p \neq 1)\\ R_l \log\frac{R_l}{R_f} \geq KN_s & (p = 1)\end{cases} \tag{7}$$

ここで，K が 1 と比較して十分に大きな値となることで，ノーマルゾーンのサーバ割当て数が，全サーバ数と比較して十分に大きな値となる．上記の条件を，R_f を R_l で表すようにまとめると次のようになる．

$$R_f \leq \begin{cases}\left[R_l^{1-p} - (1-p)KN_s R_l^{-p}\right]^{\frac{1}{1-p}} & (p \neq 1)\\ R_l \exp(-KN_s/R_l) & (p = 1)\end{cases} \tag{8}$$

式 (8) の条件に加え，ノーマルゾーンにおける分散度は，N_s を超えないことから，R_f と R_l は，次の条件を満たさなければならない．

$$N_s > \lceil R_l^p/R_f^p\rceil \geq R_l^p/R_f^p \tag{9}$$

式 (9) について，R_f を R_l で表すようにまとめると次のようになる．

$$R_f > N_s^{-\frac{1}{p}} R_l \tag{10}$$

以上の議論から，ノーマルゾーンに属するキーの処理の負荷分散を実現するためには，式 (8) と (10) の条件を，$R_f \geq 1$，および，$R_l \geq 1$ と同時に満足する必要がある．

本条件は，ノーマルゾーンの範囲のみを考慮した条件であり，すべてのランクの範囲の負荷分散を考えた場合，4.2 節で説明したように，コールドゾーンの負荷分散によって決まる条件である式 (5) を満たす必要がある．よって，負荷分散のための，ノーマルゾーン，分割条件は下記となる．

負荷分散のためのノーマルゾーン分割条件：

式 (5)，(8)，(10)，$R_f \geq 1$，および，$R_l \geq 1$ の条件を同時に満たすこと．

本条件の中で，式 (8) と (10) の条件について，R_f，および，R_l を，それぞれ，縦軸，および，横軸とするグラフにプロットするために，式 (8) と (10) の左辺と右辺の間に等号が成立する条件どうしの交点となる R_l，および，R_f を，それぞれ，R_l^*，および，R_f^* とし，これを求めると，以下の式で表される．

$$R_l^* = \begin{cases} \dfrac{(1-p)KN_s}{1-N_s^{\frac{p-1}{p}}} & (p \neq 1) \\ \dfrac{KN_s}{\log N_s} & (p = 1) \end{cases} \tag{11}$$

$$R_f^* = \begin{cases} \dfrac{(1-p)K}{N_s^{\frac{1-p}{p}} - 1} & (p \neq 1) \\ \dfrac{K}{\log N_s} & (p = 1) \end{cases} \tag{12}$$

5. 任意のデータサイズ変化におけるインメモリデータ・サイズ近似的最小化

4.3 節の議論をもとに，サーバ間の負荷分散を実現するための条件を，グラフにプロットすると，**図 6** となる．本図の縦軸，および，横軸は，それぞれ，R_f，および，R_l である．本図は，$p \geq 1$，$R_f^* \geq 1$，かつ，$R_l^* \geq \hat{R}_l$ の場合である．同図の斜線の領域は，負荷分散のためのノーマルゾーン分割条件を満足する領域である．また，斜線内の黒丸付きの折れ線は，式 (2) 直後の記述，および，表 1 に示した R_l と R_f の関係を表している．既存提案技術では，各ランクのデータのサイズに大きなばらつきがない場合，上記の負荷分散のためのノーマルゾーン分割条件が満たされる範囲内において，R_l をできる限り減少させることで，インメモリデータ・サイズを最小化することができることを明らかにした．本論文では，各ランクのデータのサイズの分布が任意の分布の場合でも，負荷分散のためのノーマルゾーン分割条件が満たされる範囲内で，R_l をできる限り減少させることでインメモリデータ・サイズを最小化可能であることを証明する．そのため，図 6 で示した，$p \geq 1$，$R_f^* \geq 1$，かつ，$R_l^* \geq \hat{R}_l$ 以外の条件についても，R_l を，できる限り減少させることは，既存提案技術で得た結論と同じであり，以降の議論には影響を与えないため，代表的なグラフのみ示すこととする．

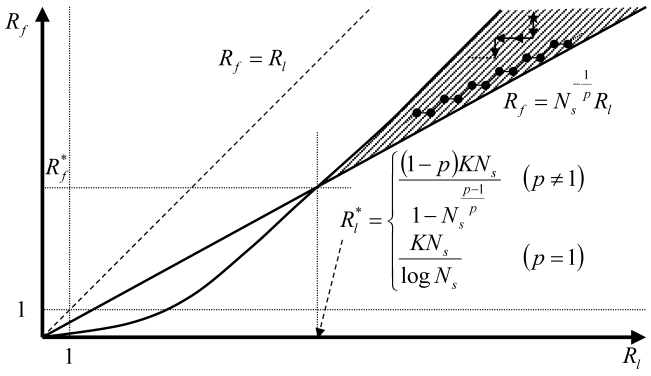


図 6 $R_f^* \geq 1$ における負荷分散を実現する条件

Fig. 6 Condition to accomplish load balancing when R_f^* is equal to or greater than 1.

各ランクのデータのサイズが任意の分布の場合，図 6 の斜線の領域の中で，インメモリデータのサイズの最小化を行うために，R_f，および，R_l を，どのように決めるかを説明する．ここで，各ランク r のキーを処理するために必要なデータのサイズを $e(r)$ (> 0) と表すこととする．

図 6 の斜線の領域を例として，負荷分散のためのノーマルゾーン分割条件を満足する領域の中のある点 (R_f, R_l) から開始して，その点をどのように移動することがインメモリデータのサイズを減少させる方向になるかを考える．その方向を明らかにするため，点 (R_f, R_l) のときのインメモリデータのサイズを，$m(R_f, R_l)$ と表すこととする．そして，$m(R_f, R_l)$ の値を，R_f を $R_f + 1$ $(= R_{f+})$ に変化させたときのインメモリデータのサイズ $m(R_{f+}, R_l)$，および，R_l を $R_l + 1$ $(= R_{l+})$ に変化させたときのインメモリデータのサイズ $m(R_f, R_{l+})$ と比較して，点 (R_f, R_l) を移動する方向を導く．まず，インメモリデータのサイズ $m(R_f, R_l)$ は，式 (2) の分散度の定義から，次式で表される．

$$m(R_f, R_l) = N_s \sum_{r=1}^{R_f - 1} e(r) + \sum_{r=R_f}^{R_l - 1} e(r) \left\lceil \frac{R_l^p}{r^p} \right\rceil + \sum_{r=R_l}^{N_r} e(r) \tag{13}$$

上式において，R_f を R_{f+} に変化させたときのインメモリデータのサイズ $m(R_{f+}, R_l)$ は，次式となる．

$$m(R_{f+}, R_l) = N_s \sum_{r=1}^{R_f} e(r) + \sum_{r=R_f+1}^{R_l - 1} e(r) \left\lceil \frac{R_l^p}{r^p} \right\rceil + \sum_{r=R_l}^{N_r} e(r) \tag{14}$$

よって，R_f を R_{f+} に変化させたときのインメモリデータのサイズの変化は，以下のようになる．

$$m(R_f, R_l) - m(R_{f+}, R_l)$$

$$= e(R_f)\left(\left\lceil \frac{R_l^p}{r^p} \right\rceil - N_s\right) \leq 0 \quad (15)$$

次に，R_l を R_{l+} に変化させたときのインメモリデータのサイズ $m(R_f, R_{l+})$ は，次式となる．

$$m(R_f, R_{l+}) = N_s \sum_{r=1}^{R_f-1} e(r) + \sum_{r=R_f}^{R_l} e(r) \left\lceil \frac{R_{l+}^p}{r^p} \right\rceil + \sum_{r=R_l+1}^{N_r} e(r) \quad (16)$$

よって，R_l を R_{l+} に変化させたときのインメモリデータのサイズの変化は，以下のようになる．

$$\begin{aligned} & m(R_f, R_l) - m(R_f, R_{l+}) \\ &= \sum_{r=R_f}^{R_l-1} e(r) \left\{ \left\lceil \frac{R_l^p}{r^p} \right\rceil - \left\lceil \frac{R_{l+}^p}{r^p} \right\rceil \right\} \\ &\quad + e(R_l)\left(1 - \left\lceil \frac{R_{l+}^p}{R_l^p} \right\rceil\right) \leq 0 \end{aligned} \quad (17)$$

以上の議論から，負荷分散のためのノーマルゾーン分割条件を満足する領域の中のある点 (R_f, R_l) を，R_f，および，R_l について減少させる方向に移動させることにより，$e(r)$ の定義によらず，インメモリデータのサイズを減少させることができることが分かる．図 6 の例で見てみると，同図の斜線の領域に示した星印の位置の点 (R_f, R_l) から開始し，矢印の方向に探索していくことでインメモリデータのサイズを減少させることができる．このことに加え，式 (2) 直後の記述，および，表 1 に示した R_f の説明のように，R_l の値によって，R_f は一意に決まる．また，R_f の決定方法から明らかなように，R_l を小さくするにつれて，R_f の値は変化しないか，減少する．よって，上記のインメモリデータのサイズを減少させるためには，R_f，および，R_l について減少させる方向に移動させることで可能となるという結論と合わせると，できる限り，R_l を減少させることでインメモリデータのサイズの削減が可能ということが分かる．

他方，文献 [32], [33] で我々は，インメモリデータのサイズを最小化する Zipf 分布型処理要求に適した負荷分散方式を提案し，$e(r)$ が大よそ一定と見なすことができる場合，負荷分散のためのノーマルゾーン分割条件を満たす R_f と R_l の領域の中で，できる限り R_l を小さな値とすることで，インメモリデータのサイズを最小化できることを明らかにしてきた．この R_l の決定方法は，本論文で明らかにした，任意の $e(r)$ を前提とした，R_l の決定方法と同じであることから，明らかに，文献 [32], [33] の方法は，図 6 で示した条件以外のすべての条件においても，ランクの処理に必要なインメモリデータ・サイズ $e(r)$ が，任意の分布において利用可能である．

6. 評価

6.1 評価項目

本論文の提案方法の目的は，ランク r の処理に必要なインメモリデータ・サイズ $e(r)$ が任意の分布となる Zipf 分布型処理要求において，以下を満たすことである．

(1) サーバ間の処理負荷の分散
(2) サーバのインメモリデータ・サイズの削減

このうち，上記の目的 (1) については，$e(r)$ の影響を受けないため，文献 [32], [33] による評価で明らかにしてきた結果と同じとなる．よって，本論文では，上記目的の (1) の評価については参考の範囲とし，目的 (2) の評価を中心に議論する．

6.2 評価に用いるインメモリデータ・サイズのモデル

評価においては，前章までの議論で任意としてきた，$e(r)$ を定める必要がある．本評価においては，1 章，および，2 章で述べた VPN サービスを例として，VPN Gateway 選択機能を実現するサーバに本提案方法を適用した場合の評価を行う．このとき，サーバが管理する Key-Value 型のデータの中で，キーは PN を示す識別子，Value は，PN ごとに必要な制御データと VPN Gateway ごとに必要な制御データである．ここで，PN ごとの制御データのサイズ，および，VPN Gateway ごとに必要な制御データのサイズを，それぞれ，D_v，および，D_g とする．さらに通常，VPN Gateway は，2.1 節で述べたように VPN Gateway の 1 台あたりに収容可能なユーザ数に上限があり，これを N_g と表す．また，1 ユーザあたり，単位時間に VPN 接続を要求する回数は，一定とし，これを s_v と表す．これらを用いて $e(r)$ として下記の式を用いる，下記の式を用いる理由については．付録 A.1 で説明する．

$$e(r) = \begin{cases} \frac{M-1}{R_b^p-1}\left(\frac{R_b^p}{r^p} - 1\right) + 1 & (1 \leq r < R_b) \\ 1 & (R_b \leq r < N_r) \end{cases} \quad (18)$$

ここで，R_b と M については，下記である．

$$s_v N_g = \frac{A}{R_b^p} \quad (19)$$

$$M = D_g(R_b^p - 1) + 1 \quad (20)$$

また，式 (18) において，データ・サイズを $D_v + D_g$ で規格化しており，$D_v + D_g = 1$ としている．

6.3 評価方法および条件

6.1 節で述べたように提案方式の目的は，サーバ間の処理負荷の分散，および，サーバのインメモリデータ・サイズの削減である．まず，最初にサーバ間の処理負荷の分散を評価するためのサーバ負荷の評価方法を説明する．各サーバは，3 章で述べたように，ホットゾーンに含まれるキーを処理するとともに，ノーマルゾーンとコールドゾーンの

キーのハッシュ値によって割り当てられたキーを処理する．また，各キーを含む処理要求は，そのキーのランクに応じて，式 (1) で決まる発生頻度で到着する．サーバ負荷は，キーを含む処理要求の処理によって発生する負荷であり，サーバが処理を担当するキーとキーの発生頻度によって決まる．サーバ負荷の評価では，1 回のキーの処理によって発生するサーバ負荷は，キーに依存せず同じであると仮定した．また，提案方法の特徴である，サーバによって処理を担当するキーが違うことによるサーバ負荷への影響を明らかにするため，ホットゾーンとノーマルゾーンの処理で行うラウンドロビン処理によって，複数のサーバ間で等しく要求処理が分配されることを前提とした．以上の前提をもとに，各サーバに割り当てられるキーを，3 章で説明した方法で行い，その違いによるサーバの負荷をシミュレーションにより評価した．

次に，インメモリデータ・サイズの評価方法について説明する．提案方法では，ホットゾーンはすべてのサーバが同じデータを持つ．そのため，ホットゾーンの処理に必要なインメモリデータ・サイズは全サーバで同一である．また，ランク $r \geq R_b$ では，各ランクに必要なインメモリデータ・サイズは，$D_v + D_g$ となり同一であるので，割り当てられるランクの数のみによってサーバのインメモリデータ・サイズの違いが発生する．ランク $r < R_b$ に含まれる，ノーマルゾーン，および，コールドゾーンについは，ランクにより異なるインメモリデータ・サイズになる．本評価においても，各サーバに割り当てられるキーを，3 章で説明した方法で行い，その違いによるインメモリデータ・サイズをシミュレーションにより評価した．

本章の評価のための条件としては，全キー数 N_r として，十分に大きな値として，1.0×10^8 を用いた．4.2 節，および，4.3 節で，それぞれ，説明した ε，および，K については，それぞれ，1 と比較して十分に，小さい値，および，大きい値として，1.0×10^{-2}，および，1.0×10^2 とした．サーバ単体の性能は，以下の評価において，下記の式の中の R_e として表す．

$$\frac{A}{R_e^p} \tag{21}$$

ここで，A，および，p は，式 (1) で用いているものと同じであり，上式は，ランク R_e（R_e は実数）の要求頻度と同じ処理要求を処理できる能力がサーバ単体の性能であることを表している．また，サーバ数 N_s は，下記の式で表すように，すべてのランク要求頻度の総和を，サーバ単体の性能で割った値を用いた．

$$N_s = \frac{1}{f(R_e)} \sum_{r=1}^{N_r} f(r) = \frac{R_e^p}{A} \sum_{r=1}^{N_r} \frac{A}{r^p} = R_e^p \sum_{r=1}^{N_r} \frac{1}{r^p} \tag{22}$$

さらに，R_b については，N_r を十分に大きな値にとっていること，Zipf 分布はランク r が大きくなるにつれて急激に要求頻度が低下することから N_r の 1%とした．

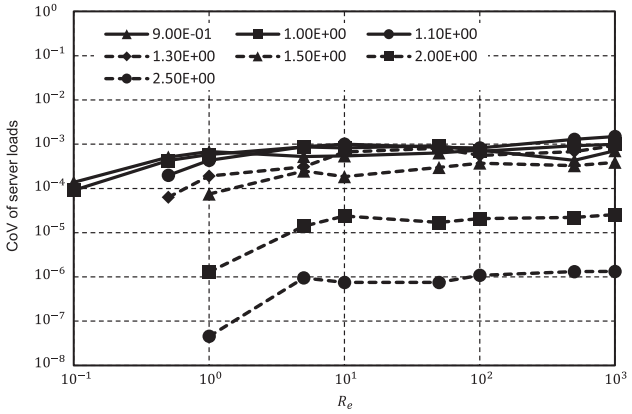


図 7 サーバ負荷の変動係数
Fig. 7 CoV of server loads.

6.4 評価結果および考察

目的 (1) のサーバ間の処理負荷の分散について，評価結果を，**図 7** に示す．本図の縦軸，および，横軸は，それぞれ，サーバ間の負荷の変動係数（Coefficient of Variation (CoV)：標準偏差の平均値に対する割合），および，R_e である．縦軸は，値が小さいほど，サーバ間の負荷のばらつきが小さいことを表す．また，横軸は，R_e が大きくなるほど，サーバ単体の性能が低くなり，その結果サーバ数が増加することを示している．本評価において，パラメータとして Zipf 定数 p を，9.0×10^{-1} から 2.5 の範囲で変化させた．本評価結果から分かるように，広い範囲のサーバ性能において，変動係数が 1.0×10^{-2} よりも小さくなっていることから，平均的な負荷から 1%未満の範囲にばらつきが収まり，十分に負荷分散が実現できていることが分かる．また，本図では，Zipf 定数 p が大きくなるにつれて，全体的な傾向として，変動係数が低下している．一般に，Zipf 定数 p が大きくなると，要求頻度が非常に高いランク範囲と非常に低いランク範囲に二極分化する．そのため，ノーマルゾーンの範囲が狭くなり，かつ，コールドゾーンの要求頻度が小さくなる．その結果，ノーマルゾーン，および，コールドゾーンによる負荷のばらつきの影響が小さくなるため，Zipf 定数 p が大きくなるにつれて，変動係数が低下していると考えられる．

目的 (2) の評価として，提案方法で実現できるインメモリデータ・サイズを，既存技術と比較する．既存技術としては，ラウンドロビンによる負荷分散，および，コンシステントハッシングという 2 つの方法の単純な組合せ（二分割法（two-zone method））を用いる．2 つの方法の使い分けについては，4.2 節で述べた $\hat{R}_l$ を用い，$1 \leq r < \hat{R}_l$ となるランク r については，ラウンドロビンによる負荷分散，$\hat{R}_l \leq r \leq N_r$ となるランク r については，コンシステントハッシングを用いる．

図 8 に，目的 (2) の評価として，提案方法によるインメモリデータ・サイズの，二分割法のインメモリデータ・

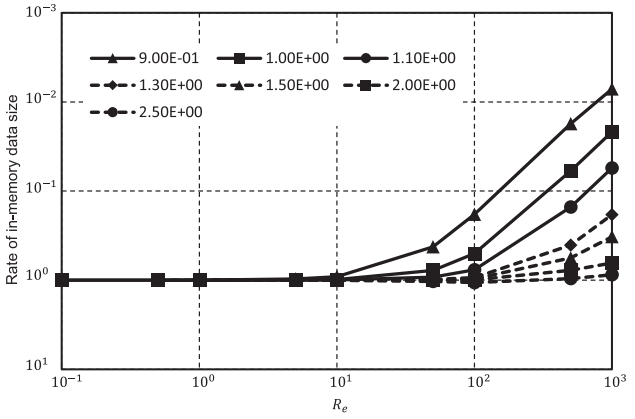


図 8 提案方法のインメモリデータ・サイズの二分割法に対する割合
Fig. 8 Rate of in-memory data size of proposed method to that of two-zone method.

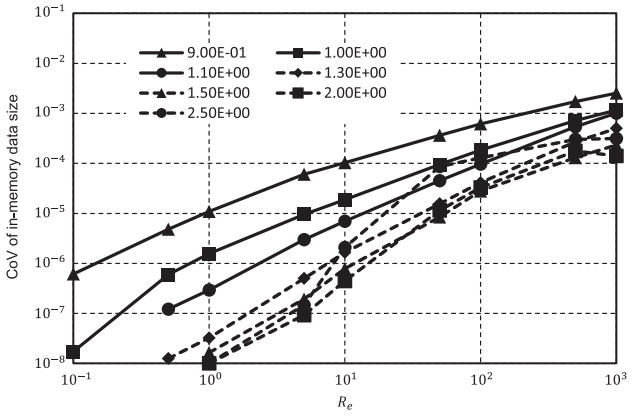


図 9 インメモリデータ・サイズの変動係数
Fig. 9 CoV of in-memory data size.

サイズに対する割合を示す．縦軸，および，横軸は，それぞれ，提案方法によるインメモリデータ・サイズの，二分割法のインメモリデータ・サイズに対する割合，および，R_e である．縦軸は，値が小さくなるほど（上にいくほど）提案方法のインメモリデータ・サイズの削減量が大きくなることを表している．横軸は，図 7 と同じように，R_e が大きくなるほど，サーバ単体の性能が低くなり，その結果サーバ数が増加することを示している．本評価において，パラメータとして Zipf 定数 p を，9.0×10^{-1} から 2.5 の範囲で変化させた．また，M について，最低のインメモリデータ・サイズである 1 と比較して十分大きな値として，1.0×10^3 を用いた．本図から分かるように，全体として，サーバ数が大きい（R_e が大きい）ほど，提案方法の効果が大きくなることが分かる．

Zipf 定数 p の変化による提案方法の効果に着目すると，Zipf 定数 p が小さいほど，提案方法の効果が高い．図 7 の考察で述べたように，一般に，Zipf 定数 p が大きくなると，要求頻度が非常に高いランク範囲と非常に低いランク範囲に二極分化する．本評価でのインメモリデータ・サイズは，式 (18) で表されるように，要求頻度に比例したモデルである．そのため，要求頻度と同じように，インメモリデータ・サイズの大きいランクと小さいランクに急峻に分割されると考えられる．さらに，コールドゾーンの大半は，インメモリデータ・サイズが一定の値になると考えられる．そのため，Zipf 定数 p が大きい場合，二分割法による評価結果に近くなり提案方法によるインメモリデータ・サイズが二分割法のインメモリデータ・サイズに近づいていると考えられる．

図 9 は，各サーバに必要なインメモリデータ・サイズの変動係数を表したグラフである．インメモリデータ・サイズの変動係数は，サーバ間でのインメモリデータ・サイズのばらつきを表している．本図の縦軸，および，横軸は，それぞれ，インメモリデータ・サイズの変動係数，および，R_e である．縦軸は，値が小さくなるほど，サーバ間でのインメモリデータ・サイズのばらつきが小さくなることを表している．横軸は，図 7 と同じように，R_e が大きくなるほど，サーバ単体の性能が低くなり，その結果サーバ数が増加することを示している．本評価において，パラメータとして Zipf 定数 p を，9.0×10^{-1} から 2.5 の範囲で変化させた．本図から分かるように，サーバ数が大きい（R_e が大きい）方向に向けて，変動係数は増加しているが，1.0×10^{-2} と比較して十分小さな値であり，サーバのインメモリデータ・サイズの平均値に対して 1%以下のばらつきになっている．このことから，インメモリデータ・サイズのサーバ間のばらつきも十分に小さい値に抑えられていることが分かる．

Zipf 定数 p の変化に着目すると，全体的な傾向として，Zipf 定数 p が大きいほど，変動係数が小さな値となっている．図 8 の評価結果の議論で述べたように，本評価で用いた各ランクのインメモリデータ・サイズのモデルの場合，Zipf 定数 p が大きくなると，インメモリデータ・サイズの大きいランクと小さいランクに急峻に分割されると考えられる．インメモリデータ・サイズの大きいランクは大半がホットゾーンと考えられ，この範囲はすべてのサーバが同じ量のデータを保持するため，インメモリデータ・サイズのばらつきへの影響は小さいと考えらえられる．さらに，コールドゾーンの大半は，インメモリデータ・サイズが一定の値になると考えられる．そのため，この範囲もインメモリデータ・サイズのばらつきへの影響は小さいと考えられる．その結果，Zipf 定数 p が大きいほど，中間のデータサイズを持つランクの範囲が減少していくことでインメモリデータ・サイズのサーバ間のばらつきが小さくなっていると考えられる．以上の評価で変化させたパラメータの範囲における全サーバ台数について，補足すると，本評価では，R_e を，1.0×10^{-1} から 1.0×10^3 まで，変化させている．この範囲のサーバ台数は，R_e に加え，Zipf 定数 p によっても影響を受けるが，Zipf 定数 p ごとに，数台前後か

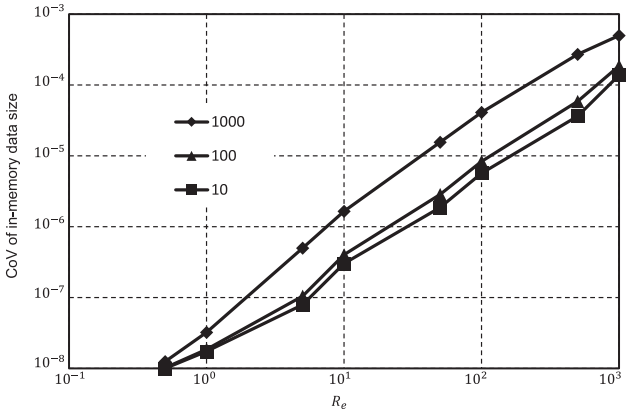


図 10 インメモリデータの変動係数の M 依存性
Fig. 10 CoV of in-memory data size depending on M.

ら，最大で一万数千台以上のサーバ数の範囲の評価を行っていることになる．

ここで，図 7，および，図 9 で評価したサーバ負荷，および，インメモリデータ・サイズの変動係数のサーバ台数への影響を考える．通常のサーバの運用において，CPU 使用率やメモリ使用率等の計算機資源について最大限使用しきるような運用は行わないことが多い．正常に動作している状態の計算機資源の使用量の範囲に一定のマージンを置くとともに，その範囲を超えた際に，異常を検知するためにさらに一定のマージンを置いた閾値を設け監視する運用を行う．このようなマージンは，応用分野によって異なるが，経験的には 1 桁台のパーセンテージになることは稀である．一方で，評価結果として得られたサーバ負荷，および，インメモリデータ・サイズの変動係数は，1%を十分に下回る値になっている．仮に 10%単位でのマージンを設定した場合，評価の結果で得られた変動係数は，マージンよりも十分に小さい値である．よって，上記の範囲のマージンから考えて，評価結果は，サーバ台数増加への影響が少ないことを示していると考えられる．

図 8，および，図 9 の評価において，上述のように，処理要求頻度にどの程度比例してインメモリデータ・サイズが増えるかを表す M を，1.0×10^3 とした．この M を変化させたときの，インメモリデータ・サイズのばらつきを，**図 10** に示す．本図の縦軸，および，横軸は，それぞれ，インメモリデータ・サイズの変動係数，および，R_e である．縦軸は，値が小さくなるほど，サーバ間でのインメモリデータ・サイズのばらつきが小さくなることを表している．横軸は，図 7 と同じように，R_e が大きくなるほど，サーバ単体の性能が低くなり，その結果サーバ数が増加することを示している．Zipf 定数 p は，1.3 とし，パラメータとして M を，1.0×10^1 から 1.0×10^3 の範囲で変化させた．本図から分かるように，サーバ数が大きい（R_e が大きい）ほど，インメモリデータ・サイズの変動係数が増加する．しかし，評価した範囲全般にわたり，変動係数は，1.0×10^{-2} と比較して十分に小さい値であり，サーバ間のばらつきは，十分に抑えられていることが分かる．

以上の評価により，Zipf 分布型処理要求に対して，各ランクを処理するためのインメモリデータ・サイズが処理要求頻度に比例するような場合において，広範な条件下で，負荷分散に加え，大幅なインメモリデータ・サイズの削減が可能であることが分かった．さらに，インメモリデータ・サイズのサーバ間のばらつきについても，サーバに必要なインメモリデータ・サイズの平均値と比較して十分に小さな範囲にあることを明らかにした．

7. 関連研究

データを用いた処理要求を，汎用計算機上で動作するサーバによって大量に処理する場合，複数のサーバで処理を行うことによりスケール性を確保する必要がある．複数のサーバが多数のクライアントからの要求を協調して処理する場合，各サーバは，クライアントからの要求を処理するためのデータを保持する必要がある．このデータのサーバへ配置方法には，配置を行うタイミングの観点から，次の 2 つの方法に分類される．1 つ目は，クライアントからサーバに到着した要求に応じてキャッシュ等の形態でサーバにデータ配置を行う方式（以下，受動配置型と呼ぶ）であり，2 つ目は，設定等によって計画的に複数のサーバにデータ配置を行う方式（以下，能動配置型と呼ぶ）である．1 章で述べたように，キャッシュを用いて受動配置型によって Zipf 分布の処理を行った場合 [17], [27]，サーバの保持するデータが時間により変化する．特に，起動時や起動してから十分に時間経過していない状況では，保持しているデータが少なく，十分な性能がでないうえに応答時間が長くなる．2 章で述べた応用を考えた場合，ユーザがネットワークへの接続要求を行ったときの短時間での接続確立が必要であるため，このような応用には望ましくない．また，通常，OS/アプリケーション更新等により一定の頻度でサーバ停止・再起動が必要となる．特に，多数のサーバでスケール性を確保している場合，更新作業はサーバの台数に比例した作業が必要となる．これに加え，キャッシュを用いた受動配置型を用いた場合，1 台のサーバの停止・再起動後，十分にキャッシュにデータが配置されることで許容範囲以上の処理性能に復帰するまで一定の時間が必要となる．サーバ台数の冗長性が高くない場合，処理性能が復帰するまで待った後，別のサーバの停止・更新・再起動を行うことになる．こういった時間のかかる作業を順次行っていくような運用は，オペレータの負担を増やすことから望ましくない．

本研究では，2.2 節で述べたように，一定以下の短時間で処理を完了する必要のある適用分野を想定していること，および，前パラグラフで述べたように運用上の観点から，受動配置型ではなく，能動配置型のデータ配置を対象とし

ている．さらに，配置されたデータの格納方法として，メモリ等高速な記憶デバイスに配置することを想定している．

能動配置型のデータ配置としては，複数のサーバに同一のデータを配置するデータ複製 [35], [36], [37], [38], [39], [40], [41]，および，データを分割して各サーバに配置するシャーディング [16], [18], [19], [20], [21], [22], [42] がある．2.2 節で説明したように，サーバ台数の削減には，計算性能とストレージ量の双方において計算資源の平準化が必要である．Zipf 分布の主な特徴は，キー全体の中で偏って非常に高い頻度で利用されるキーとそれに関連したデータが存在することである．また，Zipf 分布は，Web キャッシュ等，キーが大量に存在する分野で観測されている．まず，非常に高い頻度で利用されるキーが存在していることに対しては，複数のサーバで同一のキーの処理を行うことが必要である．このためには，データ複製 [35], [36], [37], [38], [39], [40], [41] が利用可能であり，これにより計算性能観点の計算資源の平準化を行うことができる．

次に，Zipf 分布が，キーおよびそれに関連付けられたデータが大量に存在する分野で観測されており，その分野への応用を考える．この場合，データを保存する記憶装置の有効利用が重要である．本研究で想定している応用としてメモリ上にデータを配置することを考えると，メモリ容量と比較して大量のデータが存在する場合も同じである．そのため，サーバ全体の処理に必要なデータを各サーバのメモリを有効利用するため複数のサーバ間で分担して保持することが必要である．このデータの分担保持には，シャーディング [16], [18], [19], [20], [21], [22], [42], [43], [44] が有効である．

以上の考えから，Zipf 分布の要求頻度特性を持つサーバ処理の負荷分散のために，本研究では，データ複製とシャーディングの組合せにより，計算資源の平準化を実現するアプローチをとってきた．特にシャーディングの手法の中で，Virtual Node の利用によりサーバ数の増減に対してデータ全体のサーバ全体対する再配置のコストが削減できることや Cassandra 等実利用の進んでいる実装での利用を考慮してコンシステントハッシングの利用を前提とした．

本論文の対象としている，Key-Value 型のデータを扱うことのできる実装には Redis [18], Cassandra [19], HBase [20], Oracle NoSQL Database [22] 等，様々なものが存在する．これらの実装では，シャーディングとデータ複製技術を併用しているものが多い．このデータ複製とシャーディングの併用という観点からは，本論文の提案方法と共通している．しかし，これらの実装を用いた場合，データ複製において，複製の数は，複製ノード，テーブル，カラムの組合せ等，粗い粒度で設定することになる．そのため，要求頻度の高いキーと低いキーの双方が同じ数だけ，複製されてしまう．たとえば，Cassandra の場合，replication factor というパラメータによってデータ複製が行われるサーバ数が決まる．この replication factor は，同様のデータ構造を持つ keysapce 単位での設定であるため，本研究で対象としている応用においてはすべてのデータが同じ数だけデータ複製されてしまう．そのため，ストレージ量が要求頻度の低いキーとそれに関連付けられたデータの蓄積のために大量に使用され，特にメモリ等の高速なデバイスを用いたインメモリ・データベースの利用を想定した場合，利用が困難である．

一方で，たとえば，Cassandra のデータ複製の機能において，現在実装されている keyspace 単位ではなく，1) データ複製の数を row 単位での設定とする，かつ，2) 各サーバが，メモリ等の高速な記憶デバイスにデータを能動配置するという変更を行えば，本論文での提案を，Cassandra の設定方法として実現することができると考えられ，本論文の提案技術を効率的に実利用できると考えられる．

8. おわりに

本論文では，Zipf 分布型の特性を持ち，かつ，Zipf 分布の各ランクの処理に必要なデータが，多様なサイズ分布を持つデータ処理について，サーバ負荷の平準化，および，インメモリデータ・サイズの近似的最小化を実現する負荷分散方法を提案した．提案方法の実現は，これまで提案してきた，Zipf 分布型処理要求のサーバ負荷平準化とインメモリデータ・サイズの近似的最小化を実現する負荷分散方法に対して，理論的拡張を行うことで達成した．本理論的拡張において，既存提案方法ではデータサイズがランクによらず一定であるという条件であったことに対し，任意のデータサイズ分布の条件で既存提案方法が利用可能であることを示した．提案方法に対して，処理要求の頻度に比例したサイズのデータを持つ条件での，シミュレーションによる評価を行った．その結果，広範な条件下で，負荷分散に加え，大幅なインメモリデータ・サイズの削減が可能であること，および，インメモリデータ・サイズのサーバ間のばらつきについても，サーバに必要なインメモリデータ・サイズの平均値と比較して十分に小さな範囲にあることを明らかにした．

参考文献

[1] The World Wide Web Consortium (W3C), available from ⟨https://www.w3.org⟩.
[2] Gray, J. and Reuter, A.: Transaction Processing: Concepts and Techniques, Morgan Kaufmann (1993).
[3] Mockapetris, P.: Domain Names - Concepts and Facilities, Internet Engineering Task Force (IETF) Request for Comments: 1034 (1987), available from ⟨https://www.ietf.org/rfc/rfc1034.txt⟩.
[4] Rigney, C., Willens, S., Rubens, A. and Simpson, W.: Remote Authentication Dial In User Service (RADIUS), Internet Engineering Task Force (IETF) Request for Comments: 2865 (2000), available from ⟨https://www.ietf.org/rfc/rfc2865.txt⟩.

[5] Wahl, M., Howes, T. and Kille, S.: Lightweight Directory Access Protocol (v3), Internet Engineering Task Force (IETF) Request for Comments: 2251 (1997), available from ⟨https://www.ietf.org/rfc/rfc2251.txt⟩.

[6] Schulzrinne, H., Rao, A. and Lanphier, R.: Real Time Streaming Protocol (RTSP), Internet Engineering Task Force (IETF) Request for Comments: 2326 (1998), available from ⟨https://www.ietf.org/rfc/rfc2326.txt⟩.

[7] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. and Schooler, E.: SIP: Session Initiation Protocol, Internet Engineering Task Force (IETF) Request for Comments: 3261 (2001), available from ⟨https://www.ietf.org/rfc/rfc3261.txt⟩.

[8] National Institute of Standards and Technologies: The NIST Definition of Cloud Computing (2011), available from ⟨http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf⟩.

[9] European Telecommunications Standards Institute: Network Functions Virtualisation (NFV); Architectural Framework, ETSI GS NFV 002 V1.2.1 (2014), available from ⟨http://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.02.01_60/gs_NFV002v010201p.pdf⟩.

[10] Chisnall, D.: *The Definitive Guide to the Xen Hypervisor*, Prentice Hall (2007).

[11] Wang, Z., Das, S.K., Kumar, M. and Shen, H.: Update Propagation through Replica Chain in Decentralized and Unstructured P2P Systems, *Proc. Int'l Conf. Peer-to-Peer Computing* (*P2P'04*), pp.64–71 (2004).

[12] Cuenca-Acuna, F.M., Martin, R.P. and Nguyen, T.D.: Autonomous Replication for High Availability in Unstructured P2P Systems, *Proc. 22nd IEEE Int'l Symp. Reliable Distributed Systems*, pp.99–108 (2003).

[13] Gopalakrishnan, V., Silaghi, B., Bhattacharjee, B. and Keleher, P.: Adaptive Replication in Peer-to-Peer Systems, *Proc. 24th IEEE Int'l Conf. Distributed Computing Systems*, pp.360–369 (2004).

[14] Yamashita, T.: Distributed View Divergence Control of Data Freshness in Replicated Database Systems, *IEEE Trans. Knowledge and Data Engineering*, Vol.21, No.10, pp.1403–1417 (2009).

[15] Yamashita, T.: Dynamic Replica Control Based on Fairly Assigned Variation of Data with Weak Consistency for Loosely Coupled Distributed Systems, *Proc. 22nd IEEE Int'l Conference on Distributed Computing Systems* (*ICDCS*), pp.280–289 (2002).

[16] Karger, D., Lehman, E., Leighton, T., Panigrahy, R., Levine, M. and Lewin, D.: Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web, *Proc. 29th Annual ACM Symposium on Theory of Computing*, pp.654–663 (1997).

[17] Kabir, F. and Chiu, D.: Reconciling Cost and Performance Objectives for Elastic Web Caches, *Proc. Int'l Conf. Cloud Computing and Service Computing*, pp.88–95 (2012).

[18] Redis, available from ⟨http://redis.io⟩.

[19] The Apache Cassandra database, available from ⟨http://cassandra.apache.org⟩.

[20] Apache HBase, available from ⟨https://hbase.apache.org⟩.

[21] MongoDB, available from ⟨https://www.mongodb.com⟩.

[22] Oracle NoSQL Database, available from ⟨https://www.oracle.com/database/nosql/index.html⟩.

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

[24] Breslau, L., Cao, P., Fan, L., Phillips, G. and Shenker, S.: Web Caching and Zipf-Like Distributions: Evidence and Implications, *Proc. IEEE INFOCOM*, pp.126–134 (1999).

[25] Yamakami, T.: A Zipf-Like Distribution of Popularity and Hits in the Mobile Web Pages with Short Life Time, *Proc. 7th Int'l Conf. Parallel and Distributed Computing*, Applications and Technologies (*PDCAT'06*), pp.240–243 (2006).

[26] Chen, J., Long, H. and Liang, L.: Distributed Hot Spots Caching Mechanism for Queries with Popular Distribution, *Proc. Int'l Symposium on Computer Science and Computational Technology*, pp.418–421 (2008).

[27] Nair, G. and Jayarekha, P.: A Rank Based Replacement Policy for Multimedia Server Cache Using Zipf-Like Law, *Journal of Computing*, Vol.2, No.3, pp.14–22 (2010).

[28] Fan, X., Cao, J., Mao, H., Wu, W., Zhao, Y. and Xu, C.: Web Access Patterns Enhancing Data Access Performance of Cooperative Caching in IMANETs, *Proc. 17th Int'l Conf. Mobile Data Management*, pp.50–59 (2016).

[29] Bianchi, S., Servu, S., Felber, P. and Kropf, P.: Adaptive Load Balancing for DHT Lookups, *Proc. 15th Int'l Conf. Computer Communications and Networks*, pp.411–418 (2006).

[30] Gu, Y., Chen, L. and Tang, K.: A Load Balancing Method under Zipf-Like Requests Distribution in DHT-based P2P Network Systems, *Proc. Int'l Conf. Web Information Systems and Mining*, pp.656–660 (2009).

[31] Broadband Forum, available from ⟨https://www.broadband-forum.org/⟩.

[32] 山下高生，栗田弘之，高田直樹，南　拓也，太田賢治：Zipf分布型の処理要求に適したスケールアウト手法における負荷分散と記憶域近似的最小化，電子情報通信学会 NS/IN合同ワークショップ，IN-2012-177, pp.137–142 (2013).

[33] 山下高生，栗田弘之，高田直樹：Zipf分布に適したスケールアウト型負荷分散とインメモリデータ・サイズの近似的最小化，電子情報通信学会論文誌 B, Vol.J100-B, No.9, pp.851–866 (2017).

[34] 松原　望，縄田和満，中井検裕：統計学入門，東京大学出版会 (1991).

[35] Bernstein, P.A., Hadzilacos, V. and Goodman, N.: *Concurrency Control and Recovery in Database Systems*, Addison-Wesley (1987).

[36] Helal, A., Heddaya, A. and Bhargava, B.: *Replication Techniques in Distributed Systems*, Kluwer Academic Publishers (1996).

[37] Ladin, R., Liskov, B. and Ghemawat, S.: Providing High Availability Using Lazy Replication, *ACM Trans. Computer Systems*, Vol.10, No.4, pp.360–391 (1992).

[38] Pu, C. and Leff, A.: Replica Control in Distributed Systems: An Asynchronous Approach, *Proc. ACM SIGMOD'91*, pp.377–386 (May 1991).

[39] Gray, J., Helland, P., O'Neil, P. and Shasha, D.: The Dangers of Replication and a Solution, *Proc. ACM SIGMOD'96*, pp.173–182 (June 1996).

[40] Fischer, J.J. and Michael, A.: Sacrificing Serializability to Attain High Availability of Data in an Unreliable Network, *Proc. 1st ACM Symp. Principles of Database Systems*, pp.70–75 (May 1982).

[41] Cox, P. and Noble, B.D.: Fast Reconciliations in Fluid Replication, *Proc. Int'l Conf. Distributed Computing Systems*, pp.449–458 (2001).

[42] PostgreSQL-XC, available from ⟨https://sourceforge.net/projects/postgres-xc⟩.

[43] Karger, D. et al.: Web caching with consistent hashing, *Computer Networks*, Vol.31, No.11-16, pp.1203–1213 (1999).
[44] 入江道生，岩佐絵里子，金子雅志，福元 健，上田清志：通信ノードにおけるコンシステント・ハッシュ法を用いた負荷分散とデータ複製方式，電子情報通信学会論文誌 B，Vol.J97-B, No.1, pp.31–40 (2014).

付 録

A.1 評価に用いたインメモリデータ・サイズのモデル

本論文の提案方法の評価において，6.2 節で述べたように Zipf 分布の各ランク r のデータサイズ $e(r)$ を，式 (18) とした．本付録では，$e(r)$ として式 (18) を用いた根拠について説明する．

ランク r の要求頻度は，式 (1) から $f(r) = A/r^p$ となる．また，ユーザあたり単位時間に PN 接続を要求する回数は，s_v であるから，ランク r のキーに該当する PN の全ユーザ数は，下記の式で表される．

$$\frac{f(r)}{s_v} = \frac{A}{s_v r^p} \tag{A.1}$$

今，$f(r)/s_v \geq N_g$ であるとき，必要な VPN Gateway の数は，以下の式となる．ここで，VPN Gateway の数は実際には 1 以上の整数であるが，データサイズ変化の影響評価を単純化するため，実数として扱う．

$$\frac{f(r)}{s_v N_g} = \frac{A}{s_v N_g r^p} \tag{A.2}$$

前述のように，PN ごとの制御データのサイズ，および，VPN Gateway ごとに必要な制御データのサイズが，それぞれ，D_v，および，D_g なので，ランク r の Value のデータサイズは，以下となる．

$$D_v + D_g \frac{f(r)}{s_v N_g} = D_v + D_g \frac{A}{s_v N_g r^p} \tag{A.3}$$

上式を以下のように変換する．

$$D_v + D_g \frac{A}{s_v N_g r^p} = D_v + D_g + D_g \left(\frac{A}{s_v N_g r^p} - 1 \right) \tag{A.4}$$

ランク R_b を 1 台の VPN Gateway で収容可能なユーザ数のランクの最小値とすると，ランク R_b において式 (A.2) が 1 になるため，以下の式が成り立つ．

$$\frac{f(R_b)}{s_v N_g} = \frac{A}{s_v N_g R_b^p} = 1 \tag{A.5}$$

この式を以下のように変形する．

$$s_v N_g = \frac{A}{R_b^p} \tag{A.6}$$

これを式 (A.4) に代入すると，下記のようになる．

$$D_v + D_g \frac{A}{s_v N_g r^p} = D_v + D_g + D_g \left(\frac{R_b^p}{r^p} - 1 \right) \tag{A.7}$$

データの単位を，$D_v + D_g$ とし，$D_v + D_g = 1$ とするとともに，M を以下のように定義する．

$$M = D_g(R_b^p - 1) + 1 \tag{A.8}$$

これを，式 (A.7) に代入すると，

$$D_v + D_g \frac{A}{s_v N_g r^p} = \frac{M-1}{R_b^p - 1} \left(\frac{R_b^p}{r^p} - 1 \right) + 1 \tag{A.9}$$

となる．ランク $r > R_b$ に関しては，各ランクの VPN のユーザ数が 1 台未満の VPN Gateway で収容できることから，一台の VPN Gateway で複数の VPN を収容することが妥当である．この場合，VPN ごとの VPN Gateway 数は 1 であり，データサイズは $D_v + D_g$ となる．以上の議論から，データの単位を，$D_v + D_g$ とし，ランク r のデータサイズ $e(r)$ を，次のように定義する．

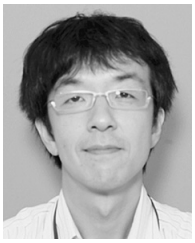
$$e(r) = \begin{cases} \frac{M-1}{R_b^p - 1} \left(\frac{R_b^p}{r^p} - 1 \right) + 1 & (1 \leq r < R_b) \\ 1 & (R_b \leq r < N_r) \end{cases} \tag{A.10}$$

山下 高生 （正会員）

1990 年京都大学工学部電気工学科卒業．1992 年同大学大学院工学研究科修士課程修了．同年日本電信電話株式会社入社．以来，NTT 情報流通プラットフォーム研究所等において，広域分散処理基盤，データ複製，大規模情報システム，認証技術，ネットワーク制御技術の研究開発に従事．現在，NTT ネットワークサービスシステム研究所主任研究員．博士（情報学）．IEEE，電子情報通信学会，APS 各会員．

栗田 弘之

2005 年東京大学工学部電子情報工学科卒業．2007 年同大学大学院情報理工学系研究科修士課程修了．同年日本電信電話株式会社入社．以来，NTT 情報流通プラットフォーム研究所等において，Wi-Fi，VPN 等のネットワークサービスにおける認証技術や，ネットワーク機能仮想化（NFV）に関する研究開発に従事．現在，東日本電信電話株式会社主査．電子情報通信学会会員．



高田 直樹

1996 年仙台電波高専情報システム専攻卒業．同年日本電信電話株式会社入社．以来，NTT グループ企業数社で VOD サーバ，モバイルアプリケーション，認証サーバの開発等，商用化開発に従事．2009 年より，NTT 情報流通プラットフォーム研究所等において，NGN 用認証サーバ，および，汎用サーバによるネットワーク仮想化技術の研究開発に従事．現在，NTT ネットワークサービスシステム研究所主任研究員．