範囲検索可能な分散クラウドストレージにおける電力消費を考慮した負荷分散機構の提案

木全 崇[1] 寺西 裕一[1] 原井 洋明[1]

概要：我々は，これまでにデータに付与された時間や値に基づく範囲検索が可能な分散ストレージを実現する手法として，分散されたノード全体を範囲検索可能なオーバーレイネットワークを用いて構成された分散クラウドストレージを効率的に管理・制御する手法を提案してきた．しかしながら，これまでの手法では，システムにかかる負荷を分散ストレージを構成するノード全体に平準化することを目的としており，ストレージリソースの使用量が小さい状況であっても，多くのノードを利用し続ける問題が存在していた．本研究では，これまで提案してきた分散クラウドストレージの管理・制御手法に対し，システム全体の状況を考慮して，必要なリソース量が小さい状況のもとでは，システム全体が消費するリソースそのものを削減可能とする拡張を施したアルゴリズム (VNLB-ES) を提案する．本稿ではシミュレーション評価によって，VNLB-ES が，従来の手法と同等のコストで，稼働する物理計算機の数を削減し，電力消費量を削減できることを確認した．

A Load Balancing Mechanism Considering Electric Power Consumptions for Range Queriable Distributed Cloud Storages

TAKASHI KIMATA[1] YUUICHI TERANISHI[1] HIROAKI HARAI[1]

1. はじめに

スマートフォン，センサー，家電などの多種多様なモノがインターネットを介して通信を行ない，現実世界の状況を把握・分析して実社会の活動を支援する IoT(Internet of Things) に基づくサービスが注目されている．IoT に基づくサービスでは，これまでインターネットに参加することがなかった多様なセンサー等から得られる無数のデータ (ビッグデータ) を組み合わせ，相関性や新規性を見い出すことが重要であり，多種多様なセンサーが生成するデータを，高い可用性のもと効率的に保存・処理可能なクラウド基盤の必要性が高まっている．上記を実現するクラウド基盤技術としては，キーバリューストア型の分散ストレージやそれに基づく分散処理技術が数多く用いられている [1] [2] [3] [4] [5] [6] ．この構成では，ネットワーク接続された多数の計算機 (以下ノード) を用いてデータを分散して保存・処理することができる．

キーバリューストア型の分散ストレージやそれに基づく分散処理技術は，ノードにおけるデータの分担量に偏りがあると，分担量が多いノードが処理のボトルネックとなりシステム全体の性能が低下し得るため，ノード間でデータの分担量を均等化する負荷分散が重要である．一方，クラウド運用においては，多数のノードが同時稼働する場合の電力消費量も課題であり，必要な計算リソースの量が少ない状況のもとでは，できるだけ稼働するノード数を減らせる必要がある．

既存の DHT による分散ストレージ [1] [2] [3] [4] は，コンシステントハッシュを用いることで，高い負荷分散性能を実現可能する．しかし，データが持つキーの値の順序は，コンシステントハッシュによって保たれず，キーの値が近い場合であっても，異なるノードに保存される．このため，キーの値に基づく範囲検索を行うことができず，例えば，「6 月 28 日から 30 日のセンサーデータ」や「東経 141 度，北緯 43 度の地点から 1km 以内のセンサデータ」といった時間や空間に関する範囲の検索が重要となる IoT には適し

[1] 国立研究開発法人 情報通信研究機構
National Institute of Information and Communications Technology

ていない．

キーの順序を保存し，範囲検索を可能とする分散ストレージの実現方法 [5] [6] も提案されている．これらの方法では，キーの順序を保存した上でノード間の負荷分散を行なうため，担当するキーの値が近い隣接ノード間で分担量の調整を行なう．しかし，ノードが担当するキー空間を縮小，拡張させる際，キー空間の調整がさらに別のノードとの間で必要になる場合があるなど，負荷分散のためのオーバーヘッドが非常に大きくなる課題がある．

我々の研究グループでは，この負荷分散にかかるオーバヘッドを軽減させる手法として，負荷分散が必要なノード (物理ノード) を複数の仮想的なノード (仮想ノード) に分割し，仮想ノード全体を範囲検索可能なオーバーレイネットワーク (以下，オーバレイ) によって管理する手法を提案してきた [7] ．しかし，提案手法を含む既存の手法では，負荷を分散ストレージを構成するノード全体に平準化するため，リソースの使用量が小さい状況であっても，多くのノードを利用し続けてしまい，消費電力量が削減できない問題がある．

そこで本研究では，上記既存手法を拡張し，物理ノード間で負荷を平準化させつつ，仮想ノードの配置を物理ノード上にまとめることで，負荷分散と消費電力量の削減を両立させる方法を提案する．

2. 既存手法: VNLB

まず，提案手法がベースとして用いる既存手法 [7] について述べる．以下，この既存手法を VNLB (Virtual Nodes-based Load-balancing) と表記する．

2.1 VNLB の概要

VNLB では，図 1 にあるように，物理ノード (Physical nodes) に複数の仮想ノードを稼働する．VNLB は，この仮想ノード単位で後述の負荷分散を行なう．仮想ノードが担当するデータ数はシステム全体で同一数に定められ，物理ノードが保持する仮想ノードの数は，物理ノードの性能に応じて決められる．

各仮想ノードはキー空間中のある範囲を担当し，その範囲内のキーの値に対応するデータを蓄積，保持する．データの保存は，< キー，データ > のペアを指定して行ない，データの取得は，キー，または，キーの範囲を指定して行なう．あるデータに対する処理を実行させたい場合は，そのデータのキーを担当する仮想ノードが処理を行なう．また，データが保存されていない空ノード (empty virtual node) は，キー空間上の (-1, 0) のキーを持つこととする．各空ノードのキーは，乱数により決定する．以下，(-1, 0) 範囲の空ノードが使用するキーの範囲を EVN (Empty virtual node) 領域と呼ぶ．

VNLB は，キーの範囲を指定した検索を分散環境でスケーラブルに実行するため，どの仮想ノードがどの範囲を担当するかを，キーの順序を保存することができる構造化オーバレイアルゴリズムを用いて管理する．VNLB では，構造化オーバレイアルゴリズムとして，Chord# を用いた．

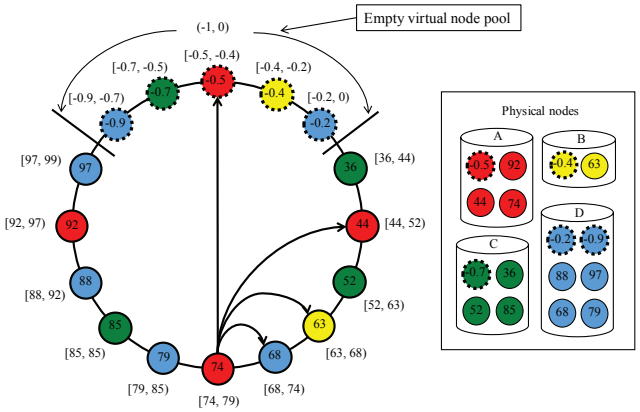

図 1 VNLB-ES の構造例

Chord# はリングベースの構造化オーバレイであり，各ノードは自ノードの次にキーが大きいノードへのポインタ (successor と呼ぶ) と，キーが次に小さいノードへのポインタ（predecessor と呼ぶ）を持つ（ただし，最大キーのノードの successor は最小キーのノード，最小キーのノードの predecessor は最大キーのノードを指す）．また，各ノードは複数のレベルを持つ経路表を保持し，レベル i に 2^i 個離れたノードへのポインタを格納するスキップ構造を構築する．この経路表を用いて，検索キーと経路表中のポインタが指すノードのキーに基づいた greedy ルーティングを行うことで，ノード数 n に対し，最大検索ホップ数は $\log_2 n$ となる．

VNLB では，各仮想ノードのキー空間上の担当範囲における範囲の開始値をキーとして保持してオーバレイに参加する．担当ノードの検索は，対象とするデータに対応するキー (または検索したいキーの範囲の最小値) を指定してオーバレイを検索する．

図において，同じ色の仮想ノードは，同じ物理ノードに配置されていることを示している．

2.2 物理ノード間の負荷調整

VNLB においては，物理ノード間の負荷分散は，負荷が過大となった物理ノード上で稼働中の仮想ノードを，他の過負荷ではない物理ノード上の仮想ノードと入れ替えることによって行なう．この操作は “swap” と呼ばれる．

物理ノードの負荷が他の物理ノードと比べて大きいかどうかは，物理ノード自身が判断する．物理ノード自身が過負荷かどうかを判断するため，VNLB では，各物理ノードが，仮想ノード上の経路表を用いたランダムサンプリングを行ない，空ノードの数とサンプリングを行なった物理ノードの数から平均負荷を推定し，それよりも一定以上負

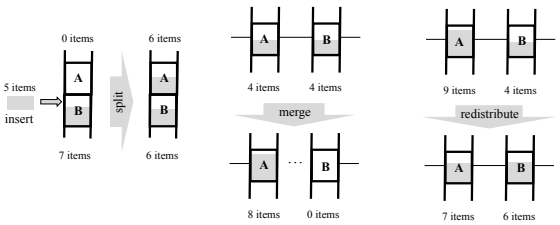

図 2 仮想ノード間の負荷調整

荷が大きければ過負荷と判断する．

swap において，仮想ノードの入れ替えを行なう際，仮想ノードが持つネットワークアドレスが変化することになるため，オーバレイの構造は変化する．

2.3 仮想ノード間の負荷調整

VNLB では，仮想ノードが保持できるデータ数には上限があり，その上限を越えないようにしなければならない．また，VNLB では仮想ノードが少しでもデータを保持していると稼働状態となる．上記物理ノード間の負荷調整では，物理ノード上で稼働中の仮想ノードの数によって負荷を計測するため，仮想ノード自体が保持するデータ量が少ないと，稼働中の仮想ノードが多いにもかかわらず物理ノードとしては負荷が小さいと判断してしまう．そこで，仮想ノードが保持するデータ数が一定以上とならないよう，また，小さすぎる状態とならないよう，VNLB は，“split”, “merge”, “redistribute” という仮想ノード間の負荷調整オペレーションを規定している (図 2)．

split は，ある仮想ノードへのデータ追加の際，追加によって一定数を越える場合に，その仮想ノードのデータを空ノードとの間で均等となるよう分割保持する動作をする．merge は，一定以下のデータ数しか持たない 2 つの隣接仮想ノードを，ひとつの仮想ノードに集約し，ひとつを空ノードとする動作をする．redistribute は，一定以上のデータ数を持つノードと隣接する仮想ノード間との間で，保持数が均等となるよう分割する動作をする．

VNLB はこれらの動作によって，範囲検索可能な分散ストレージにおける負荷分散を少ないオーバヘッドで実現する．しかし，前述のとおり，負荷を全ての物理ノード全体に平準化する動作をするため，全体のリソースの使用量が小さい状況であっても，多くの物理ノードが稼働状態となり，消費電力量を削減できない問題がある．

3. 提案手法

我々は，前章に示した既存手法を拡張し，物理ノード間で負荷を平準化させつつ，仮想ノードの配置を物理ノード上にまとめることで，負荷分散と消費電力量の削減を両立させる方法を提案する．

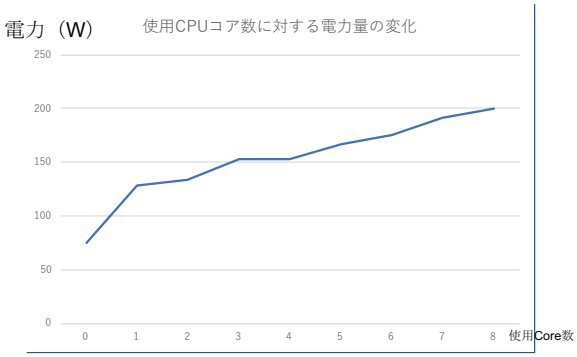

図 3 サーバにおける利用 CPU コア数の増加に伴う電力消費量の変化

3.1 基本方針

図 3 は，実際のサーバにおける電力使用量を計測した結果を示している．X 軸は，使用 CPU コア数，縦軸は消費電力量である．図に示している通り，使用している CPU コア数が 0 のときの電力量は，1 以上の場合と比べて小さい．これは，CPU 等を休眠状態とすることができるためである．したがって，全く使用されていない物理ノードの数を増やすことができれば，全体の消費電力量を減らすことができると考えられる．suspend や shutdown を行なうことができれば，消費電力はさらに削減される．

そこで，提案方式は，VNLB の動作を全体のリソース使用量が小さい状況のもとでは，空の物理ノードを多く確保し，消費電力量を削減するよう拡張する．以下，提案方式を VNLB-ES (VNLB Electricity Saving extension) と表記する．

VNLB-ES は，EVN 領域を，Active(利用可能), Sleep(休眠中), Busy(過負荷状態) の 3 種類の領域に分割する (図 4)．

Active 領域へは，各物理ノードから，仮想ノード数が過負荷状態となる数以下までの仮想ノードが参加する．Sleep 領域へは，物理ノードが休眠状態にある仮想ノードが参加する．Busy 領域へは，各物理ノードから，仮想ノード数が過負荷状態となる数以上の仮想ノードが参加する．

3.2 アルゴリズム

以下では，VNLB-ES のアルゴリズムについて述べる．VNLB-ES における仮想ノード間の負荷調整は，VNLB と同様の動作によって実現する．VNLB-ES の VNLB との主な違いは，物理ノード間の負荷調整の方法にある．

3.2.1 物理ノードの追加

物理ノード N が分散ストレージに追加されると，まず，オーバレイを用いてサンプリングを行ない，過負荷となる仮想ノード数 μ を得る．N が保持可能な仮想ノード数を M_N，N 上の n 番目の仮想ノードを $v_{N_n}(n = 1, 2, ...M_N)$ と表記するとき，v_{N_n} がオーバレイに参加する際，以下の値をキーとして保持する．

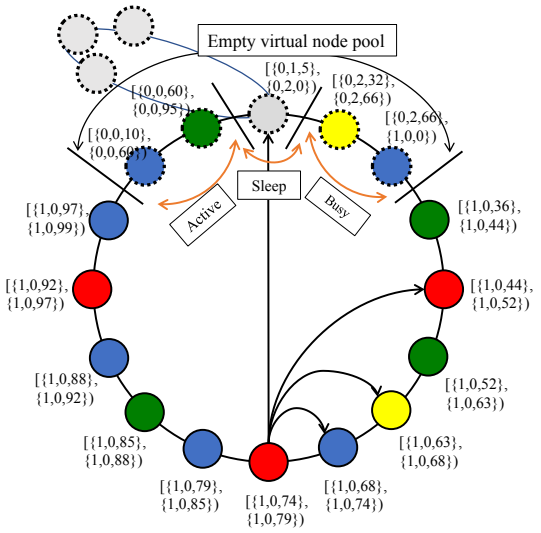

図 4 VNLB-ES の構造例

$$\{w, p, r\}$$

キーの順序は，lexicographical order である．すなわち，前の要素の順序が後の要素の順序よりも優先される．w は，以下の値を持つ．

$$w = \begin{cases} 0, & v_{N_n} \text{ is empty} \\ 1, & v_{N_n} \text{ is not empty} \end{cases}$$

すなわち，$w = 0$ が，VLNB における EVN 領域に相当する．また，p は，以下の値を持つ．

$$p = \begin{cases} 0, & v_{N_n} \text{ is not empty or } n \leq \mu \\ 1, & N \text{ is sleeping} \\ 2, & \mu > n \end{cases}$$

$w = 0$ のとき，すなわち，EVN 領域内では，$p = 0$ が Active 領域, $p = 1$ が Sleep 領域，$p = 2$ が Busy 領域である．物理ノードが休眠状態にあるとき，当該物理ノードが保持する全ての仮想ノードに $p = 1$ を指定する．$w = 1$ のときは，p の値は 0 となる．

r は，$w = 0$ のときは乱数値となり，$w = 1$ のときは，仮想ノードが担当するキーの範囲の最小値となる．$w = 0$ のときの乱数値は，初期状態で生成して各仮想ノードに割り当てる．

実際には，$p = 1$ の仮想ノードは，休眠状態にあるため，オーバレイに参加できない．このため，Active 領域の最後の仮想ノードを保持する物理ノードは，休眠状態にある仮想ノードのポインタ (ネットワークアドレス) 集合を保持する．ある仮想ノードが Active 領域の最後のノードとして参加する場合，参加後に predecessor ノード（参加処理以前は Active 領域の最終ノード）から Sleep 領域に属している仮想ノードのポインタ集合を引き継ぐ．

3.2.2 物理ノードの休眠状態への移行と復帰

同じ物理ノード上で動作する全ての仮想ノードが EVN 領域に属する状態となった物理ノードは，Active 領域の最後尾に参加している仮想ノードを検索し，自身が保持する仮想ノードのポインタ情報を転送した上で，休眠状態に入る．

VNLB の挙動は，各仮想ノードはオーバーレイネットワーク上の Active 領域に存在するノードを検索し，それらに負荷を分散させることで全体の負荷分散を計ることに相当する．一方，VNLB-ES においては，各仮想ノードは，まず最初に Active 領域上に存在するノードを検索する．見つからない場合は，Sleep 領域にある物理ノードの一つを選択して，μ の値に基づいて，当該ノードが保持する仮想ノードを，Active 領域と Busy 領域に参加させる．すべての Sleep 領域の仮想ノードが稼働状態となった場合は，次に Busy 領域に存在するノードを swap 対象に決定する．

この動作により，負荷が増大する状況のもとでは，物理ノードが過負荷状態に陥る前に，休眠状態にある物理ノードを逐次起動していくことで，負荷が分散される．

4. 評価

VNLB-ES の有効性を確認するため，シミュレーションによる評価を行なった．シミュレータは，提案アルゴリズム独自の挙動を確認するために自作したものを用いている．

4.1 シミュレーション設定

シミュレーションでは，初期状態で負荷が正規分布したがっている状態，すなわち，一部のノードの負荷が高い状態から，VNLB および VNLB-ES によって負荷が分散し，収束するまでの挙動をシミュレートした．本シミュレーションでは，サンプリングによって求める過負荷状態となる仮想ノード稼働数 μ はあらかじめ計測されているものとし，7 に設定した．また，許容稼働ノード数の下限 τ を 2 とした．

表 1 に，シミュレーションで用いたパラメータを示す．物理ノード数は，100，物理ノードあたりの仮想ノード数の上限を 10 としており，最大仮想ノード数は 1,000 である．

本評価では，負荷分散処理が収束するまでにかかった swap ステップ数の平均値，および，最終的に稼働状態となった物理ノード数の平均値を比較した．それぞれの結果は 10 回の試行の平均値である．

4.2 結果

図 5 は，負荷分散処理が収束するまでにかかったステップ数の平均である．VNLB が平均 10.2 ステップであるのに対し VNLB-ES は 11.9 であった．VNLB との差は 2 ステップ程度に収まっており，負荷分散にかかるステップ数には大きな差はないと言える．

表 1　シミュレーション設定

パラメータ	値
物理ノード数	100
最大仮想ノード数	1,000
μ	7
τ	2
初期負荷分布	正規分布
試行回数	10

図 6 は，最終的に稼働状態となった物理ノード数の平均値である．VNLB は全ての物理ノードが稼働状態となるのに対し，VNLB-ES における稼働物理ノード数は 61 であり，稼働数を 39 % 削減することができた．

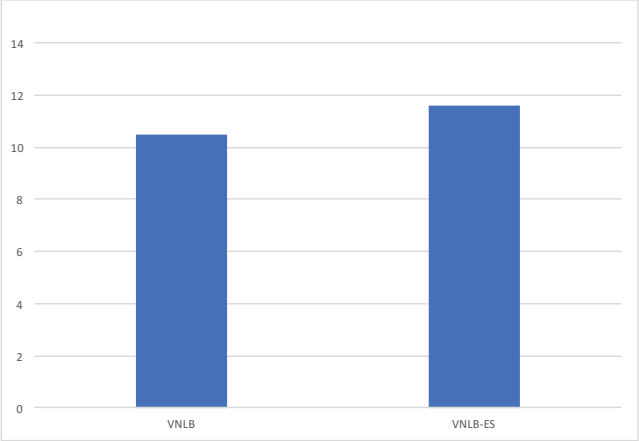

図 5　収束までにかかるステップ数の平均

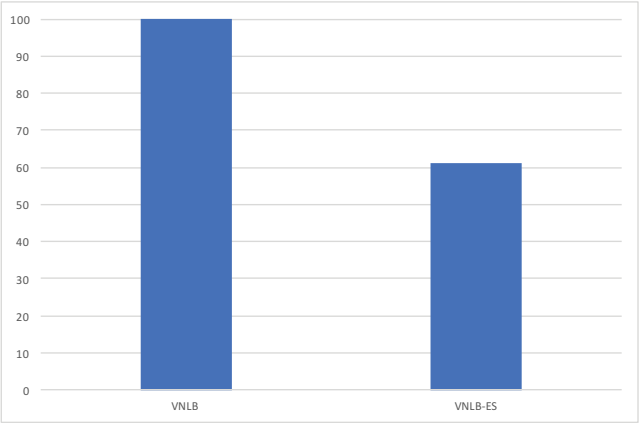

図 6　分散処理収束後の稼働物理ノード数

5. おわりに

本稿では，範囲検索可能な分散ストレージを構成するノード全体に平準化することを目的とする既存手法が持つ課題であった，ストレージリソースの使用量が小さい状況であっても，多くのノードを利用し続ける問題を解決する手法として，必要なリソース量が小さい状況のもとでは，システム全体が消費するリソースを削減可能とする新たなアルゴリズム (VNLB-ES) を提案した．また，シミュレーション評価によって，VNLB-ES が，従来の手法と同等のコストで，稼働する物理計算機の数を削減し，電力消費量を削減できることを確認した．

今後は，さらなる詳細な評価，アルゴリズムの実機上での実装等を進めるとともに，実際のクラウド上のストレージや長期稼働する実アプリケーションに適用した場合の効果等についても確認を進める．

参考文献

[1] P. Felber, P. Kropf, E. Schiller, S. Serbu, “Survey on load balancing in peer-to-peer distributed hash tables,” IEEE Communications Surveys & Tutorials 16.1, 2014.

[2] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, “Dynamo: Amazons Highly Available Key-value Store,” in Proc. of SOSP’07, 2007.

[3] “Hadoop Distributed File System,” [Online]. Available: http://hadoop.apache.org/hdfs

[4] I. Nakagawa, K. Nagami, “Jobcast - Parallel and distributed processing framework Data processing on a cloud style KVS database,” Journal of Information Processing, Vol.21, No.3, 2013.

[5] P. Ganesan, M. Bawa, H. G. Molina, “Online Balancing of Range-Partitioned Data with Applications to Peer-to-Peer Systems,” in Proc. of VLDB’04, 2004.

[6] I. Konstantinou, D. Tsoumakos, N. Koziris, “Fast and Cost-Effective Online Load-Balancing in Distributed Range-Queriable Systems,” IEEE Transations on Parallel and Distributed Systems Vol.22, No.8, 2011.

[7] X. Shao, M. Jibiki, Y. Teranishi, and N. Nishinaga, “Effective Load Balancing Mechanism for Heterogeneous Range Queriable Cloud Storage,” in Proc. of IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom 2015), pp. 405-412, 2015.