# 範囲検索可能な分散クラウドストレージ・処理基盤のための低電力消費の負荷分散手法とその評価

木全　崇[1,a)]　寺西　裕一[1,b)]　細川　貴史[1,c)]　原井　洋明[1,d)]

**概要**：本稿では，IoT(Internet of Things) において要求されるデータに付与された時刻などの連続した値に基づく範囲検索を，高いスケーラビリティで実現可能な分散クラウドストレージを低電力消費で管理・制御する新たな手法 VNLB-ES (Virtual Nodes-based Load-Balancing with Electricity Saving extension) を提案する．VNLB-ES は，範囲検索可能な分散クラウドストレージにおいて，システムにかかるストレージ負荷・検索負荷・処理負荷を効率的に複数の物理ノード（計算機）に分散させる機能に加え，システム全体の状況から，要求される処理に必要となる全体のリソース量が少ない状況では，稼働させる物理ノードそのものの数を削減し，停止/suspend 状態とする機能を提供し，システム全体の消費電力を削減可能とする．本稿では，シミュレーション評価によってシステム全体の処理量に応じて，動的に稼働する物理ノードの数を削減し，処理量が少ない状況において最大で電力消費量を 56%削減可能であることを確認した．

**キーワード**：IoT，P2P, 分散キーバリューストア，範囲検索，負荷分散

# Proposal and Evaluation of A Power-Saving Load Balancing Method of Range Quariable Cloud Storage and Processing Platform

Takashi Kimata[1,a)]　Yuuichi Teranishi[1,b)]　Takafumi Hosokawa[1,c)]　Hiroaki Harai[1,d)]

***Keywords:*** IoT，P2P, distributed key-value store, range query, load balancing

## 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といったいわゆるビッグデータ処理が重要である．ビッグデータ処理を効率的に実現する上では，多種多様なセンサーが生成するデータを，高い可用性のもと高速に所望のデータを検索可能な分散ストレージ，それら膨大なデータを効率的に分析処理可能な分散処理基盤が必要となる．

これら分散クラウド基盤を実現するに辺り，これまで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提案されている [1] [2] [3] [4] [5] [6]．しかしながら，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，サーバにおけるデータの分担量に偏りがあると，分担量が多いサーバ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，各サーバ間で担当するデータの分担量を効率的に均等化する負荷分散技術が重要となるため，DHT（Distributed Hash Table）を用いた既存キーバリューストアによる分散ストレージ構成技術 [1] [2] [3] [4] では，コンシステントハッ

1 情報通信研究機構
4-2-1, Nukui-Kitamachi, Koganei, Tokyo 184–8795, Japan
a) kimata@nict.go.jp
b) teranisi@nict.go.jp
c) takafumihosokawa@nict.go.jp
d) harai@nict.go.jp

ⓒ 2019 Information Processing Society of Japan

シュを用いることで，高い負荷分散性能を実現可能とする．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，異なるサーバ上に保存される．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 には適していない．

既存 DHT を拡張することにより，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サーバ間の負荷分散を行なうため，サーバにおけるデータの分担量に偏りが生じた際には，担当するキーの値が近い隣接サーバ間で分担量の調整を行なう．しかし，サーバ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サーバ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，負荷分散が必要なサーバ（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].

一方，クラウド運用においては，多数のサーバが同時稼働する場合におけるシステム全体の電力消費量が重要な課題である．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サーバ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，多くのサーバを利用し続け，消費電力量が削減できない問題がある．

そこで本稿では，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，処理を実施する仮想ノードの配置を特定の物理ノード上にあえて偏らせることで，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. 従来研究

範囲検索可能なキーバリューストアによる分散ストレージにおける負荷分散（ロードバランシング）は，データが持つキーの順序を維持する必要があるため，DHT と比べると単純ではない．最も初期の研究としては Aspnes らによるもの [8] がある．この研究では，ある閾値を基準に閾値以下の負荷を持つノード（サーバ）が新しいデータ（要求）を受け入れ，閾値以上を越える負荷を持つノードはデータの受け入れを制限することで Skip Graph やそれらに似たデータ構造を用いた分散ストレージにおける負荷分散を実現している．しかし，この方法では，適切な閾値の設定が困難となる問題がある．

Genesan [9] らは，NBRADJUST と REORDER と呼ばれる操作を組み合わせる範囲検索可能な分散ストレージの負荷分散手法を提供している．この方法は，ノードの担当範囲を変更すると，他の多くのノードに影響が及ぶなど，負荷分散処理としてオーバヘッドが大きいという課題がある．NBRADJUST と REORDER の組み合わせにおいてオーバーヘッドを減らすことで負荷分散に必要な時間を短くした手法に NIXMIG [11] がある．

統計分析を用いた負荷分散手法として HiGLOB [10] がある．HigLOB では，統計分析を用いることでシステム全体を俯瞰した負荷分散を実現する．しかしながら，HigLOB では，リアルタイムにノード毎の詳細な情報を取得する必要があり，動的に構成が変わる環境ではメンテナンスコストが非常に大きくなる．また，複雑な実装を必要とする．

こうした課題を解決する手法として，我々はこれまでに，物理ノード上に複数の仮想ノードを起動し，それらの間でオーバレイネットワーク構築して負荷を分散させることで，オーバヘッドを大幅に削減できる手法を提案してきた [7]. しかし，文献 [7] の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サーバ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，多くのサーバを利用し続けてしまい，消費電力が大きくなってしまうという課題が残されている．

## 3. 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．

### 3.1 VNLB の概要

VNLB では，図 1 にあるように，各物理ノード (Physical nodes) 上で複数の仮想ノード（以下，ノードと表記）が稼働する．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ノードの数は，物理ノードの性能に応じて決められ，VNLB は，このノード単位で後述の負荷分散を実施する．（図において，同じ色のノードは，同じ物理ノードに配置されていることを示している．）

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，乱数により決定し，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．また，VNLB では，それら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

ⓒ 2019 Information Processing Society of Japan

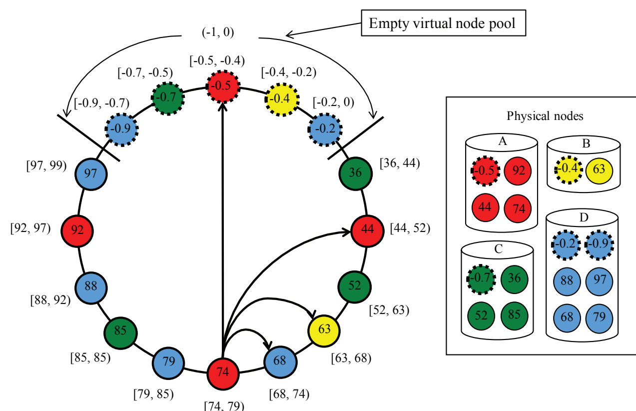


図 1　VNLB の構造例

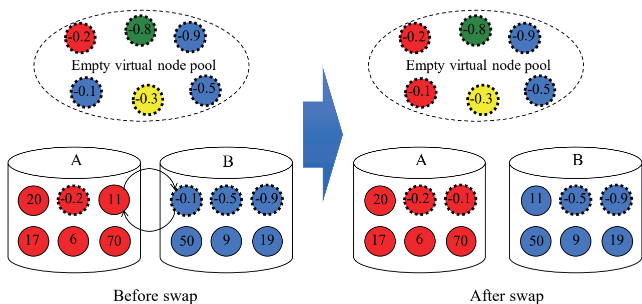


図 2　物理ノード間での swap 処理例

め，キー順序保存可能な構造化オーバレイアルゴリズム Chord# を用いて，ノードの担当範囲を管理している．

VNLB の検索に用いている 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していることから，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ことで実現する．

### 3.2　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ノードを，他の過負荷ではない物理ノード上のノードと入れ替えることによって行なう（図 2）．この操作は “swap” と呼ばれる．

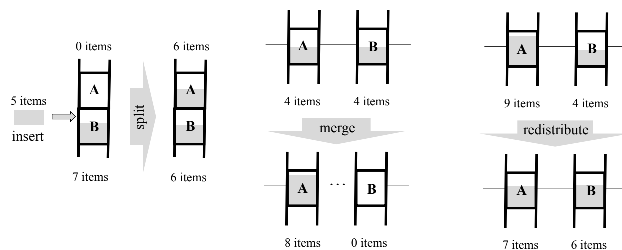


図 3　仮想ノード間の負荷調整

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，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，負荷判定に用いる閾値（$\mu$）を決定する．$\mu$ よりも一定以上負荷が大きければ過負荷と判断し，swap を実施する．swap において，ノードの入れ替えを行なう際，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．

### 3.3　ノード間の負荷調整

VNLB では，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 3)．

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．

ⓒ 2019 Information Processing Society of Japan

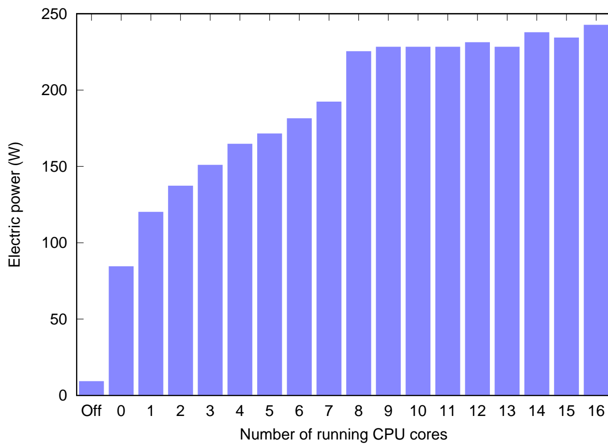


図 4　サーバにおける利用 CPU コア数の増加に伴う電力消費量の変化

表 1　電力計測を行ったサーバのスペック

| パラメータ | 値 |
|---|---|
| CPU | Xeon E5-2690 |
| コア数 | 8 |
| スレッド数 | 16 |
| HDD | 900G（ミラーリング） |
| OS | Ubuntu 16.04 |

## 4. VLNB-ES

我々は，VLNB を拡張し，物理ノード間で負荷を平準化させせつつ，処理を実施する仮想ノードの配置を特定の物理ノード上にあえて偏らせることで，負荷分散と消費電力量の削減を両立させせる方式 VNLB-ES (VNLB Electricity Saving extension) を提案する．

### 4.1　基本方針

図 4 は，本稿執筆時点で一般的なスペック（表 1）を持つサーバにおける使用電力を計測した結果を示している．

「Off」はサーバが停止/suspend 状態にある時の消費電力である. その他は，使用 CPU コア数が 0 コアから 16 コアに負荷を与えた状態で 3 秒間隔で 1 分間取得した消費電力の平均値となっている．図に示している通り，サーバ休眠時は，使用している CPU コア数が 0 以上のときの電力量と比べて比べて小さい．これは，サーバそのものを停止/suspend 状態にすることによって CPU 等を休眠状態とすることができ，管理用インタフェース (IPMI) の待ち受け電力のみ消費される状態となるためである．したがって，停止/suspend 状態の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．

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．

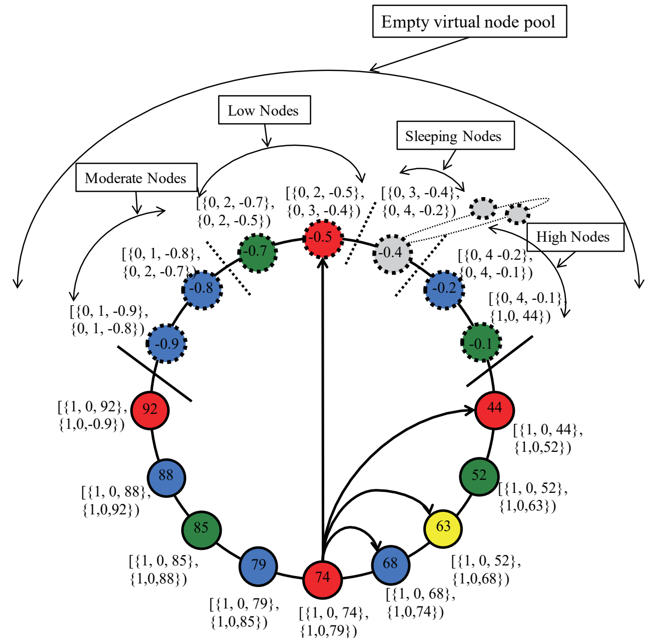


図 5　VNLB-ES の構造例

### 4.2　アルゴリズム

本項では，VNLB-ES のアルゴリズムの詳細について述べる．VNLB-ES における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．

#### 4.2.1　物理ノードの追加

VNLB-ES では，VNLB で定義された EVN 領域を，Low（利用可能で低負荷状態），Moderate（利用可能で中負荷状態）, Sleep（休眠中）, High（利用可能だが使用するとノードが過負荷状態に陥る）の 4 種類の領域に分割する（図 5）．

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，過負荷となるノード数 $\mu$，低負荷となるノード数 $\tau$ を得る．$N$ が保持可能なノード数を $M_N$，$N$ 上の $n$ 番目のノードを $v_{N_n}(n = 1, 2, ...M_N)$ 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．

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

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．

ⓒ 2019 Information Processing Society of Japan

$$p = \begin{cases} 0, & v_{N_n} \text{ is not empty or } n \le \mu \\ 1, & \tau < n \text{ and } \mu > n \\ 2, & \tau \\ 3, & N \text{ is sleeping} \\ 4, & \mu > n \end{cases}$$

$w = 0$ のとき，すなわち，EVN 領域内では，$p = 1$ が Moderate 領域, $p = 2$ が Low 領域，$p = 3$ が Sleep 領域，$p = 4$ が High 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ノードに $p = 4$ を指定する．$w = 1$ のときは，$p$ の値は 0 となる．

実際には，$p = 3$ のノードは，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，Low 領域の最後のノードを保持する物理ノードは，休眠状態にあるノードへのポインタ (ネットワークアドレス) 集合を保持する．あるノードが Low 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 Low 領域の最終ノード）から Sleep 領域に属しているノードのポインタ集合を引き継ぐ．

$r$ は，$w = 0$ のときは乱数値 (-1.0˜ 0.0) となり，$w = 1$ のときは，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ノードに割り当てる．

#### 4.2.2 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ノードが EVN 領域に属する状態となった物理ノードは，Low 領域の最後尾に参加しているノードを検索し，自身が保持するノードのポインタ情報を転送した上で，休眠状態に入る．

各物理ノードは，$\mu$，$\tau$ の閾値に基づいて，自身が保持するノードを，Low，Moderate，High 領域領域に参加させる．

#### 4.2.3 過負荷状態の物理ノードにおける負荷分散処理

自身が過負荷状態であると判定した物理ノードは，Algorithm 1 に従い負荷分散処理を実行する．Algorithm 1 において，`find_any` は，引数に指定された 2 つのキーの間にあるいずれかのノードを検索する機能を指す．この機能は Chord#のルーティングにおいて，第 2 引数をキーとした検索を行い，第 1 引数との間にあるキーにクエリが到達した時にそのキーを持つノードを得る操作によって実現可能である．また，`wake_up` は，引数に指定されたノードが属する物理ノード（Sleep 状態にある）を起動する操作である．すなわち，まず負荷分散対象となるノードを Moderate 領域，Low 領域，Sleep 領域，High 領域の順に検索する．Sleep 領域にあるノードが発見された場合は，物理ノードを起動する．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が逐次起動される．

負荷分散対象を発見できた場合，VNLB-ES は，VNLB と同様に ”swap” 処理によって負荷分散を実行する．

#### 4.2.4 低負荷状態の物理ノードにおける負荷分散処理

自身が低負荷であると判断したノードは，自身を休眠状態に移行するため負荷分散処理を実行する．分散対象の検索は，過負荷状態の負荷分散処理と同様に，まず Moderate 領域で検索し，次いで Low 領域に存在するノードを検索する．別の物理ノードから負荷分散要求を受けた場合，実行している負荷分散処理を停止する．

**Algorithm 2:** select a next node

```
1  function select
2  begin
3      q ← find_any({0, 1, −1.0}, {0, 1, 0});
4      if (q is empty) then
5          q ← find_any({0, 2, −1.0}, {0, 2, 0});
6      if (q is empty) then
7          q ← find_any({0, 3, −1.0}, {0, 3, 0});
8          if (q is not empty) then
9              wake_up(q);
10     if q is empty then
11         q ← find_any({0, 4, −1.0}, {0, 4, 0});
12     return q;
```

## 5. 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．

### 5.1 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，VNLB-ES それぞれのアルゴリズムによって処理負荷が収束するまでの挙動を再現した．初期値として与えた負荷は，システム全体として実行可能な処理量を 1 としたとき，0.1 から 0.7 まで変化させている．

表 2 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ノード数の上限を 10 としており，最大ノード数は 1,000 である．シミュレーションでは，単純化のため，物理ノードが持つ処理能力は均一であるものとし，過負荷・低負荷状態とみなすパラメータであるノード稼働数 $\mu$，$\tau$ は，あらかじめ設定されているものとした．値としては，$\mu = 8$，$\tau = 3$ を用いた．計測値としては，100 回の試行結果の平均値を採用している．

ⓒ 2019 Information Processing Society of Japan

表 2　シミュレーション設定

| パラメータ | 値 |
|---|---|
| 物理ノード数 | 100 |
| 仮想ノード数 | 10 |
| $\mu$ | 8 |
| $\tau$ | 3 |
| 初期負荷分布 | 正規分布 |
| 試行回数 | 100 |

### 5.2　結果

図 6 は，各システム負荷において，負荷分散処理を実施したあと稼働している物理ノードの台数を示している．図 7 は，同様に，負荷分散処理が収束後に各ノードに 1CPU コアが割り当てられる想定にて，図 1 で計測した消費電力を当てはめたものである．すなわち，Y 軸は消費電力を示している．それぞれの図において，「Optimal」は，全体負荷量があらかじめわかっている想定で対応する数だけ物理ノードを稼働させた，最適化状態を示している．

図 6 において，VNLB は，全ての物理ノードの性能が同一の時，物理ノード上で稼働するノード数が均一になるよう処理を分散させる．したがって，100 台の物理ノードが存在する状態において，稼働台数は常に 100 となる．一方，VNLB-ES では，物理ノード上で稼働するノード数が 0 ならば，停止/suspend 状態となり，稼働台数は減少する．停止できる台数は，負荷が上昇するにしたがって減っているが，特に負荷が小さい時に電力を削減できていることがわかる．最も負荷が低い条件である全体負荷量が 0.1 のとき，VNLB-ES の稼働台数は 20 となっており，80 台の物理サーバを停止できていることになる．

一方，図 7 では，VNLB においては稼働している物理ノード数が 100 であっても，物理ノード上で動作しているノード数が少なければ，稼働する CPU 数も少なくなるため，消費電力量は減少する．最も負荷が低い条件である全体負荷量が 0.1 のとき，VNLB の電力消費量は 11kW ととなっている．VNLB-ES では，停止ノードが存在することになるため，さらに電力量は減少し，全体負荷量が 0.1 の時の電力消費量は 4.8kW となった．すなわち，約 56% の電力を削減できていることがわかる．シミュレーションでは全体負荷量を 0.7 まで増加させて評価を実施しているが，いずれの状態においても既存手法である VNLB よりも少ない消費電力となることが確認できた．VNLB-ES は，Moderate 状態となったノード同士を対象に負荷分散が実行されず，収束状態となるためこのことが Optimal との間に差が生じる原因である．しかし，VNLB-ES と Optimal の差は最大でも数 kW 以内に抑えられており，一定の効果が得られていると言える．

## 6.　おわりに

本稿では，IoT において要求されるデータに付与された時刻などの連続した値に基づく範囲検索を，高いスケーラビリティで実現可能な分散クラウドストレージを低電力消費で管理・制御する新たな手法 VNLB-ES (Virtual Nodes-based Load-Balancing with Electricity Saving extension) を提案した．VNLB-ES は既存手法である VNLB が持つストレージ負荷・検索負荷・処理負荷を効率的に複数の物理ノード（計算機）に分散させる機能を維持したまま，システム全体に影響を与えずに必要のないノードを休止させることで電力消費の削減が可能である．それらをシミュレーション評価によって，システム全体の処理量に応じて，動的に稼働する物理ノードの数を削減し，処理量が少ない状況において最大で電力消費量を 56%削減可能であることを確認した．

しかしながら，現状の VNLB-ES にはいくつか改善すべき点も残されている．例えば，Sleep 領域に属するノードのポインタを単一のノードに保持させているため，このポインタを保持するノードがシステム全体の単一障害点となりうる．これらの課題を解決するためには，ポインタ情報を Low 領域に存在するノード間で冗長化させて保持するなどの工夫が必要となる．また，現状のアルゴリズムでは，起動する物理ノードは乱数によって決定するため，物

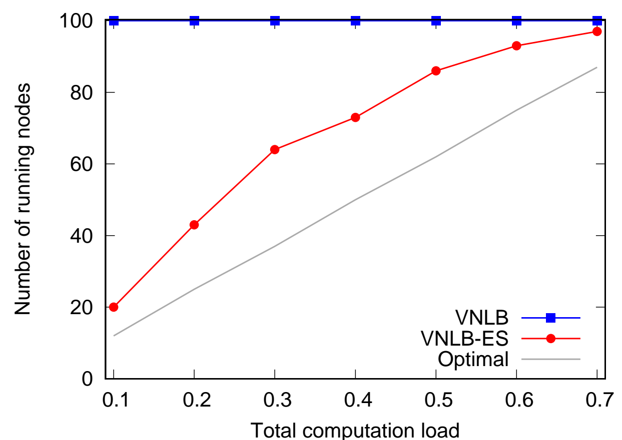


図 6　負荷分散処理収束後の稼働物理ノード数

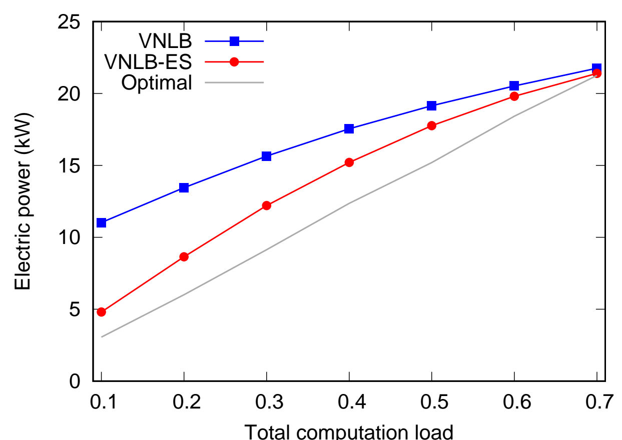


図 7　負荷分散処理収束後の消費電力

ⓒ 2019 Information Processing Society of Japan

理的に遠い位置に存在する物理ノードが起動されることがある．このような状況は，例えばエッジコンピューティングのような応答遅延が小さいことが要求される状況では望ましくない．この要求に応えるには，物理的な位置を考慮した物理ノードを選択できる機能が必要となるが，そのような機能の実現方法は今後の課題である．

今後は，さらなる詳細な評価や実機上でのアルゴリズム実装を想定した単一障害点への対応，範囲検索のボトルネックの解消などを実施し，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への適用等を行い，有用性を示していく．

## 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.
[8] J. Aspnes, J. Kirsch, A. Krishnamurthy, “Load balancing and locality in rangequeriable data structures,“ in: Proc. PODC’04, 2004.
[9] P. Ganesan, M. Bawa, H.G. Molina, “Online balancing of range-partitioned data with applications to peer-to-peer systems,“ in: Proc. VLDB’04, 2004.
[10] Q.H. Vu, B.C. Ooi, M. Rinard, K.L. Tan, “Histogram-based global load balancing in structured peer-to-peer systems,“ IEEE Trans. Knowl. Data Eng. 21 (4) (2009) 595608.
[11] I. Konstantinou, D. Tsoumakos, N. Koziris, “Fast and cost-effective online loadbalancing in distributed range-queriable systems,“ IEEE Trans. Parallel Distrib. Syst. 22 (8) (2011) 13501364.
[12] Schütt,, Thorsten and Schintke,, Florian and Reinefeld,, Alexander, “Range queries on structured overlay networks,“ Elsevier Science Publishers B. V. Computer Communications. 31 (2) (2008) 280-291.

ⓒ 2019 Information Processing Society of Japan