情報処理学会論文誌 Vol.61 No.2 339–350 (Feb. 2020)

分散クラウドストレージ・処理基盤における消費電力の削減を可能とする負荷分散手法

木全 崇[1,a)] 寺西 裕一[1,b)] 細川 貴史[1,c)] 原井 洋明[1,d)] 下條 真司[1,2,e)]

受付日 2019年5月9日, 採録日 2019年11月7日

概要：IoT（Internet of Things）において生成されるデータを，時刻などの連続値をキーとして保存・処理可能な分散クラウドストレージ上で消費電力の削減を実現する新たな管理・制御手法 VNLB-ES を提案する．VNLB-ES は，分散クラウドストレージシステムにかかるストレージ負荷・検索負荷・処理負荷を，仮想ノードを用いて効率的に複数の物理ノード（計算機）に分散させる機能を備える．また，要求される処理に必要となるリソース量が少ない状況では，仮想ノードが稼働しない物理ノードを確保して，停止/一時停止状態とする機能を提供する．これら 2 つの機能の組合せにより，物理ノード間の効率的な負荷分散を実現しつつ，システム全体の消費電力を大幅に削減する．シミュレーション評価，および，実機を用いた評価により，VNLB-ES が，既存の負荷分散方式と比べて最大で電力消費量を 70%程度削減できることを示した．

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

An Electricity Saving Load Balancing Method for Cloud Storage and Processing Platforms

Takashi Kimata[1,a)] Yuuichi Teranishi[1,b)] Takafumi Hosokawa[1,c)]
Hiroaki Harai[1,d)] Shinji Shimojo[1,2,e)]

Received: May 9, 2019, Accepted: November 7, 2019

Abstract: We propose a novel electricity saving distributed storage management scheme called VNLB-ES, that stores and processes data by continuous keys such as timestamps for IoT (Internet of Things). VNLB-ES provides an efficient load distribution function using virtual nodes. VNLB-ES efficiently balances the storage load, query processing load, and data processing load among physical nodes (computers) that consist the distributed cloud storages. In addition, VNLB-ES provides a physical node control function that stops or suspends the physical nodes which are not required for the total processing load. By the combination of these two functions, VNLB-ES achieves both efficient load balancing and significant reduction of the total electricity consumption. To show the effectiveness of the proposal, we conducted simulations and experiments and confirmed that the electricity consumption can be reduced about 70% in maximum.

Keywords: IoT, distributed key-value store, load balancing, electricity saving

1. はじめに

スマートフォン，センサ，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などから得られる無数のデータを組み合わせ，相関性や新規性を見出すといったいわゆる

[1] 情報通信研究機構
National Institute of Information and Communications Technology, Koganei, Tokyo 184–8795, Japan
[2] 大阪大学サイバーメディアセンター
Cybermedia Center, Osaka University, Ibaraki, Osaka 567–0047, Japan
[a)] kimata@nict.go.jp
[b)] teranisi@nict.go.jp
[c)] takafumihosokawa@nict.go.jp
[d)] harai@nict.go.jp
[e)] shimojo@cmc.osaka-u.ac.jp

ビッグデータ処理が重要な位置付けを占める．ビッグデータ処理を効率的に実現するうえでは，多種多様なセンサ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，それらの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．

上記分散クラウド基盤技術として，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きた [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する．このため，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．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シュを用いることで，高い負荷分散性能が実現されている．しかし，コンシステントハッシュではデータが持つキーの値の順序は保たれず，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，キーの値が近いデータの取得や処理の効率が低下する．したがって，たとえば，「6 月 28 日から 30 日のセンサデータ」や「東経 141 度，北緯 43 度の地点から 1 km 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，コンシステントハッシュは適さない．

既存 DHT を拡張することにより，キーの順序を保存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，ストレージ負荷・検索負荷・データ処理負荷をキーの順序を保存したうえで複数サーバ間で調整するため，保存・処理す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再配分を行う．しかし，キーの順序を保存するには，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り，負荷分散にかかるオーバヘッドが大きくなるという課題があった．

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，負荷分散が必要な計算機・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，複数の物理ノードにまたがる仮想ノード間の論理的オーバレイネットワーク（以下，オーバレイ）によって分散ストレージを実現する手法（以下，VNLB: Virtual Nodes-based Load-Balancing と呼ぶ）を提案してきた [7]．VNLB は仮想ノード間が自律動作する自律分散型のアルゴリズムであり，集中管理型と比べて高いスケーラビリティを担保できるという利点を持つ．

一方，クラウド運用において，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をいかに抑えるかは重要な課題である．しかし，上記提案手法を含む既存の手法は，分散ストレージを構成する全サーバを利用して負荷を分散させることを前提としており，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，消費電力量が大きくなるという問題があった．

上記をふまえた本稿の貢献は以下のとおりである．

- 既存手法である VNLB の自律分散アルゴリズムを拡張し，システム全体の消費電力量を削減させる負荷分散方法 VNLB-ES（Virtual Nodes-based Load-Balancing with Electricity Saving extension）を提案した．VNLB-ES は，全体負荷が小さくなったとき，仮想ノードが稼働する物理ノードの配置をあえて偏らせ，稼働していない物理ノードは停止/一時停止状態とすることで，システム全体の消費電力量を削減させる．
- 提案アルゴリズムについて，シミュレーションおよび実機サーバを用いた実験を行い，処理量が少ない状況において最大で 70%の消費電力量の削減が可能となることを確認した．また，360 秒（6 分）程度で負荷分散処理が完了することを確認した．
- 1,000 個の物理ノード，16,000 個の仮想ノードを対象とした大規模シミュレーション評価を行い，小規模環境と同様，最大で電力消費量を 70%削減可能となることを確認した．

2. 従来研究

データが持つキーの順序を維持する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，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サーバが新しいデータ（処理要求）を受け入れ，閾値を超える負荷を持つサーバ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，サーバに応じて適切に閾値を設定する必要があり，設定が困難となるという問題がある．

Genesan ら [9] 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，サーバの担当範囲を変更すると，他の多くのサーバに影響が及ぶなど，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 の組合せにおいて，負荷の移動先の選択を隣接サーバから選択するなどにより負荷分散に必要な時間を短くする手法として NIXMIG [11] がある．しかし，NBRADJUST と REORDER が，負荷の調整において多くのサーバに影響を及ぼすという本質的な課題を解決するものではない．

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，リアルタイムにサーバごと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を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させる手法 VNLB を提案してきた [7]．しかし，VNLB を含む既存の手法は，いずれも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．本研究は，この課題の解決に臨むものである．

3. VNLB

まず，提案手法がベースとして用いる既存手法である VNLB [7] について述べる．

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が稼働する．仮想ノードが担当する負荷（データ数，処理量）はシステム全体で同一の数に定める．また，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，VNLB は，この仮想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）．

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い，データの取得は，キー，または，キーの範囲を指定して行う．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う．また，データが保存されていない空の仮想ノード（empty virtual node; EVN）は，キー空間上の $(-1, 0)$ のキーを持つこととする．各 EVN のキーは，乱数により決定し，以下，$(-1, 0)$ 範囲の EVN が使用するキーの範囲を EVN 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，仮想ノードの担当範囲を管理している．

VNLB の検索に用いる Chord#はリングベースの構造化オーバレイネットワークであり，オーバレイネットワークに参加する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，仮想ノードの検索は，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う（**図 2**）．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て，各物理ノードに対し

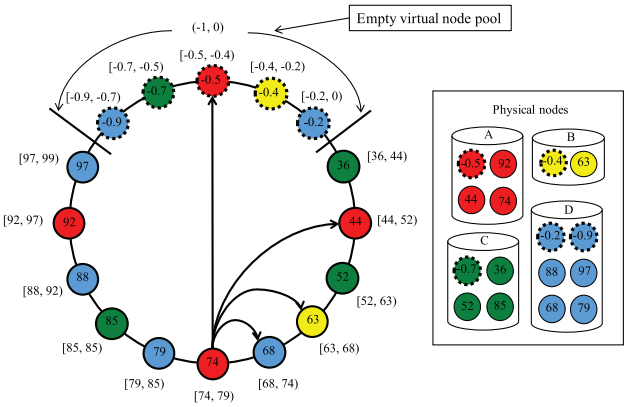

図 1 VNLB の構造例
Fig. 1 An example structure of VNLB.

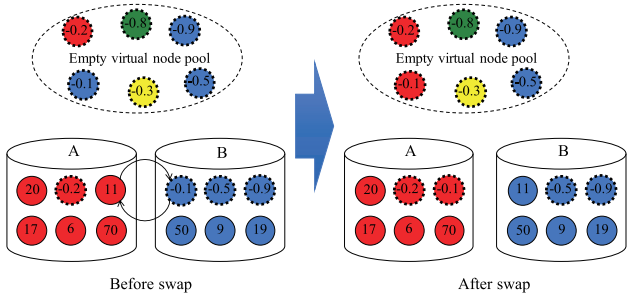

図 2 物理ノード間での swap 処理例
Fig. 2 An example of swap between physical nodes.

ランダムサンプリングを行い，EVN の数とサンプリングを行った物理ノードにおける稼働仮想ノードの数から，平均の稼働仮想ノード数を推定し，平均よりも一定以上稼働仮想ノード数の割合が多い物理ノードは過負荷状態にあると判断して swap を実施する．ここでは，過負荷と見なす仮想ノード数の割合の閾値を γ と表記する．swap において，仮想ノードの入れ替えを行う際，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．仮想ノードが分担実行している処理があるとき，処理の実行途中に入れ替えが行われると実行している処理が中断する．処理を途切れさせることなく仮想ノードを入れ替えるには，仮想ノードの複製を行ったあと，複製開始後に受け取った処理結果の複製を行うなどのいわゆる Live Migration [13] が必要となる．Live Migration としてどのような方法の適用が適切かは，処理内容に依存する部分があるため，本稿では対象外とする．

3.3 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，その上限を超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（処理を分担）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，仮想ノードが受け持つ処理・データ量が小さいと，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状態となる．そこで，仮想ノードが分担する処理・データ量が過大・過少にならないよう，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超える場合に，その仮想ノードのデータを EVN 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，1 つの仮想ノードに集約し，1 つを EVN とする動作をする．redistribute は，一定以上のデータ数を持つ仮想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．split および redistribute は，分担キーの範囲を分割することで負荷の分割や平準化を行うことが可能な場合に実行できるオペレーションである．

VNLB はこれらの動作によって，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，負荷をすべ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という問題がある．

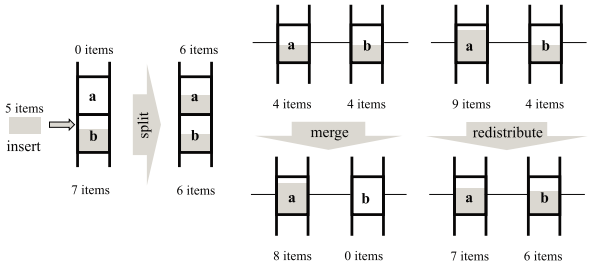

図 3 仮想ノード間の負荷調整
Fig. 3 Load balancing between virtual nodes.

4. VNLB-ES

本稿では，VNLB の課題を解決する拡張方式 VNLB-ES を提案する．VNLB-ES は，VNLB における稼働仮想ノードを収容するために必要な最小限の数の物理ノードに集約させることで負荷分散と消費電力量の削減を両立させる．

4.1 基本方針

図 4 は，本稿執筆時点で一般的な仕様（**表 1**）のサーバ単体の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/一時停止状態にあるときの消費電力である．そのほかは，使用 CPU コア数が 0 コアから 16 コアに，処理負荷の高いプロセス（映像処理プロセス）を割り当てた状態で 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．図に示しているとおり，サーバ停止時は，使用している CPU コア数が 0 以上のときの電力量と

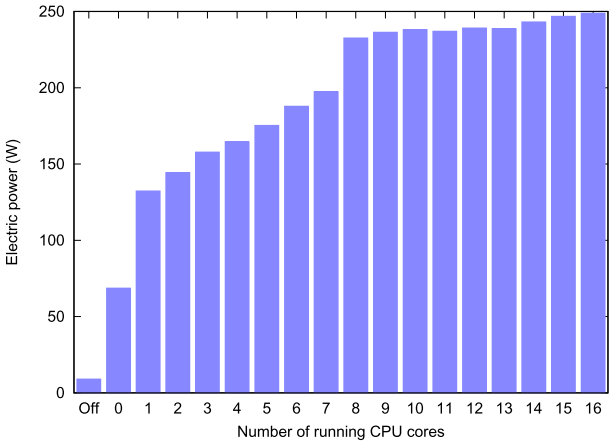

図 4 サーバにおける利用 CPU コア数の増加にともなう電力消費量の変化
Fig. 4 Electric power consumption changing the number of CPU cores on a server computer.

表 1 電力計測を行ったサーバのスペック
Table 1 The server specifications.

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

比べて小さい．これは，サーバそのものが停止/一時停止状態になることによって，多くの電力を消費する CPU などを休止させ，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/一時停止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 に対し，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挙動変更を施す．

4.2 アルゴリズム

本節では，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．

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（利用可能で通常の負荷状態），Sleep（停止/一時停止状態），High（利用可能だが過負荷状態）の 4 種類の領域に分割する（**図 5**）．

先に述べたとおり，VNLB では，物理ノード上で稼働可能な仮想ノード数に対し，稼働中の仮想ノード数が一定以上の割合であるとき，その物理ノードが過負荷状態にあると判断する．また，一定以下の割合であるとき，低負荷状態にあると判断する．ここでは，過負荷と見なす仮想ノード数の割合の閾値を γ，低負荷と見なす割合の閾値を τ と表記する．VNLB では，全物理ノードの負荷をできるだけ均一とするため，閾値 γ，τ に相当する閾値は，サンプリングによって全体の負荷を予測したうえで決定していた．これに対し，VNLB-ES は，負荷を均一化するのではなく，

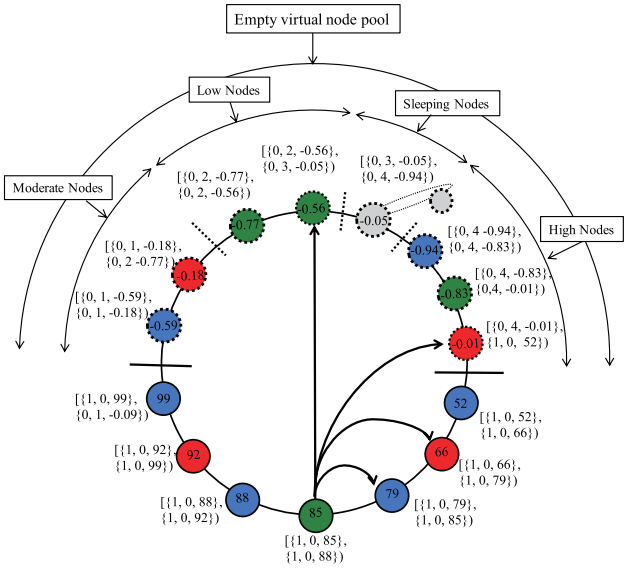

図 5 VNLB-ES の構造例
Fig. 5 An example structure of VNLB-ES.

あらかじめ決定された γ を負荷の上限とし，負荷が上限を超えない範囲で負荷を分担するよう動作する．この動作により，物理ノードが過負荷にならない範囲で，できるだけ仮想ノードが稼働しない物理ノード（空物理ノード）を増やす．VNLB-ES では，γ として OS の動作に支障が出ない程度の値を，τ として $\gamma/2$ 以下の値（2 つ以上の低負荷の物理ノードから 1 つの空物理ノードを生成できる値）を設定する．

n 個の仮想ノードを収容可能な物理ノード V 上の i 番目に稼働中の仮想ノードを V_i（$i = 1, 2, \ldots, n$）と表記するとき，仮想ノード V_i 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．

$$\{w, p, q\}$$

ただし，w は，以下の値を持つ．

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

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

$$p = \begin{cases} 0, & w = 1 \\ 1, & w = 0 \text{ and } \tau \le \frac{k}{n} \text{ and } \frac{k}{n} \le \gamma \\ 2, & w = 0 \text{ and } \frac{k}{n} < \tau \\ 3, & w = 0 \text{ and } V \text{ is sleeping} \\ 4, & w = 0 \text{ and } \gamma < \frac{k}{n} \end{cases}$$

k は，仮想ノードが属する物理ノードで稼働中の（EVN ではない）仮想ノードの数である．キーの順序は，lexicographical order とする．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．$w = 0$ 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．物理ノードが停止/一時停止状態にあるとき，当該物理ノードが保持するすべての仮想ノードに $p = 3$ を指定する．$w = 1$ のときは，p の値は 0 となる．

$p = 3$ の仮想ノードは，停止/一時停止状態にあるため，オーバレイに参加できない．このため，Low 領域の最後尾の仮想ノードは，停止/一時停止状態の物理ノードが保持する仮想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．ある仮想ノードが Low 領域の最後尾に参加する場合，参加前に最後尾であった仮想ノード（参加処理以前は Low 領域の最終ノード）から Sleep 領域に属している仮想ノードのポインタ集合を引き継ぐ．

q は，VNLB におけるキーに相当し，$w = 0$ のときは乱数値（-1.0〜0.0）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．

各仮想ノードは，γ, τ の閾値に基づいて，Low, Moderate, High 領域に参加する．

4.2.2 過負荷状態の物理ノードにおける負荷分散処理

過負荷状態にある物理ノードの負荷分散処理を，Algorithm 1 に示す．Algorithm 1 は，物理ノードにおいて稼働中の仮想ノード数の割合が γ を上回る場合に，稼働中の仮想ノードと，他の物理ノード上の EVN との間で swap を実行する動作に対応する．`recruit` は，仮想ノードが自律的に新たな EVN の取得を行う機能である．`recruit` の動作は Algorithm 2 に示すとおりである．Algorithm 2 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仮想ノードを検索する機能である．この機能は，q に乱数を指定したうえでオーバレイネットワークの検索機能を用いることで実現できる．Algorithm 2 に示したとおり，過負荷状態の仮想ノードが実行する `recruit` は，負荷分散対象となる EVN を Moderate 領域，Low 領域，Sleep 領域，High 領域の順に検索する．

Algorithm 1: a function called when overloaded

```
1  function on_overloaded
2  begin
3      for i ← γn to i ← n do
4          if V_i is not empty then
5              q ← V_i.recruit(true);
6              if q = φ then
7                  // the system is satulated.
8                  return;
9              if physical node of q is sleeping then
10                 wake_up(q.physical_node());
11             swap(V_i, q);
12             if swap was failed then
13                 go to 5;
```

Algorithm 2: `recruit` function

```
1  function recruit (flag)
2  begin
3      q ← find_any({0, 1, −1.0}, {0, 1, 0});
4      if q = φ then
5          q ← find_any({0, 2, −1.0}, {0, 2, 0});
6          if flag = false then
7              return q;
8      if q = φ then
9          q ← find_any({0, 3, −1.0}, {0, 3, 0});
10     if q = φ then
11         q ← find_any({0, 4, −1.0}, {0, 4, 0});
12     return q;
```

Sleep 領域から得られる EVN の物理ノードは停止/一時停止状態にある．Algorithm 1 における `wake_up` は，この状態にある物理ノードを起動する操作である．`recruit` により得られた EVN との間で swap 操作を行うことで，過負荷状態にある物理ノードにおいて，稼働中の仮想ノードの数は減り，負荷は軽減される．swap 操作は，対象となる仮想ノードの状態によっては失敗することがある．たとえば，swap 対象の仮想ノードが属する物理ノードが，次に述べる低負荷状態から停止/一時停止状態への移行状態にある場合などである．この場合は `recruit` 操作をやり直す．

swap 操作時，物理ノード上で稼働中の仮想ノードの数の割合が閾値 γ, τ を超える場合，キーの値が変化し，オーバレイ構造も変化する．

システム全体の負荷が高い状態となっているとき，EVN は High 領域にしか存在しない状態となる．EVN が High 領域のみとなる状態は，VNLB-ES が想定する負荷分散制御の対象外であり，`recruit` は，乱数によっていずれかの High 領域の仮想ノードを得るのみとなる．これは VNLB と同様の動作である．ただし，VNLB では，負荷変動などによる偏りを補正する動作をするのに対し，VNLB-ES はそのような動作をしない点が異なる．High 領域に参加する仮想ノードが使用されるのは，システム全体の負荷が高い状況のときのみである．このため，$i > \gamma n$ を満たす V_i は，オーバレイ上，つねに $p = 4$，すなわち，High 領域に参加していたとしても動作は変わらない．したがって，$i > \gamma n$ を満たす V_i は，つねに $p = 4$ としてオーバレイに参加することで，稼働中の仮想ノードの数の変化にともないキーの値を変化させるオーバヘッドを減らすことができる．図 5 では，この動作に従っていくつかの仮想ノードが High 領域に参加している．

以上の動作により，負荷が増大する状況のもとでは，物理ノードの負荷が上限に到達する前に，停止/一時停止状態にある物理ノードを逐次起動し，負荷を分散させる．

4.2.3 低負荷状態の物理ノードにおける負荷分散処理

低負荷状態にある物理ノードの負荷分散処理を，Algorithm 3 に示す．Algorithm 3 は，稼働中の仮想ノード数の割合が τ を下回る場合に，物理ノード上のすべての稼働中の仮想ノードと，他の物理ノード上の EVN との間で swap を実行する動作に対応する．Algorithm 3 から実行される `recruit` は，Algorithm 1 の場合と異なり，Moderate 領域を検索し，ついで Low 領域に存在する仮想ノードを検索するが，Sleep 領域と High 領域は検索対象としない．

Low 領域にある EVN が見つかった場合，見つかった EVN が属する物理ノードも低負荷状態にあるため，処理を継続すると稼働仮想ノード数の増減が繰り返される可能性がある．このため，Algorithm 3 を実行中の物理ノードと，見つかった EVN が属する物理ノードのいずれか一方が処理を継続しないよう動作する．本稿で後述するシミュ

Algorithm 3: a function called when underloaded

```
1  function on_underloaded
2  begin
3      for i ← 1 to i ← τn do
4          if V_i is not empty then
5              q ← V_i.recruit(false);
6              if q = φ then
7                  return;
8              if physical node of q is underloaded then
9                  decide continue or abandon;
10             swap(V_i, q);
11             if swap was failed then
12                 go to 5;
13             q ← find({0, 3, 1.0});
14             register pointer of V_i to q;
15     sleep();
```

レータ，および，実機実装では，物理ノードが低負荷状態と判定した時刻を検索時に取得し，その大小をもとに継続するか否かを決定した．

同じ物理ノード上で動作するすべての仮想ノードがEVN領域に属する状態となった物理ノードは，Low 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うえで，停止/一時停止状態に入る．

4.2.4 消費電力の見積り

以下では，アルゴリズムの動作収束後，VNLB-ES と VNLB の消費電力の見積り，および，それらの比較を考える．ここでは，簡単のため，物理ノードの性能が均一であり，物理ノードあたりの仮想ノード数が一定となると想定する．

物理ノード数を N とする．また，システム全体で実行可能な処理量を 1 としたときの，システム全体にかかる負荷量（全体負荷）を L（$0 < L < 1.0$）とする．全体負荷としては，VNLB において過負荷に陥らない範囲で収容可能な量である場合を考える．すなわち，$L < \gamma$ が満たされるとする．1 つの物理ノード上に k 個の仮想ノードが稼働状態となっているときの消費電力を $p(k)$，停止/一時停止状態の消費電力を $p(0) = 0$ とする．このとき，VNLB における全体消費電力 P_{VNLB} は，収束後の理想状態においては次のとおり見積もることができる．

$$P_{\mathrm{VNLB}} = Np(Ln) \tag{1}$$

一方，VNLB-ES における全体消費電力 $P_{\mathrm{VNLB\_ES}}$ は，収束後の理想状態においては次のとおり見積もることができる．

$$P_{\mathrm{VNLB\_ES}} = \frac{LN}{\gamma} p(\gamma n) \tag{2}$$

したがって，VNLB と比べた VNLB-ES の消費電力の比

表 2 シミュレーション・実験設定
Table 2 Parameters in the simulation and the experiment.

パラメータ	値
仮想ノード数/物理ノード	16
γ（VNLB）	$L \times 1.25$
γ（VNLB-ES）	0.75
τ（VNLB-ES）	0.25
初期負荷分布	一様分布

R は次のとおり表される．

$$R = \frac{P_{\mathrm{VNLB\_ES}}}{P_{\mathrm{VNLB}}} = \frac{\frac{NL}{\gamma} p(\gamma n)}{Np(Ln)} = \frac{L}{\gamma} \frac{p(\gamma n)}{p(Ln)} \tag{3}$$

R は，

$$\frac{\gamma}{L} > \frac{p(\gamma n)}{p(Ln)}$$

が成り立つ限り，1 より小さい値となる．すなわち，上記が成り立つとき，VNLB に対し，VNLB-ES は，物理ノード数 N によらず消費電力が削減されるといえる．実際には，新たに物理ノードを稼働させる消費電力と比べると，すでに物理ノードが稼働した状態から新たに仮想ノードを稼働するためにかかる消費電力の差は大幅に小さくなるという物理ノードの性質によって，多くの場合，$\frac{p(\gamma n)}{p(Ln)}$ は 1 に近い値となり，

$$R \simeq \frac{L}{\gamma} < 1$$

となる．

上記式では，$P_{\mathrm{VNLB\_ES}}$ は収束後の稼働仮想ノード数の割合が γ となる理想状態での消費電力を想定している．しかし，提案アルゴリズムでは，τ 以上 γ 以下の稼働仮想ノード数の割合となった物理ノードは，負荷分散操作を終了する．また，ここでは処理の内容によらず $p(k)$ の値は一定となると想定しているが，実際には処理の内容に応じて異なる．

5. 評価

VNLB-ES の有効性を確認するため，シミュレーション，および，実機を用いた実験による評価を行った．

5.1 設定

VNLB と VNLB-ES の基本性能を同条件で比較するため，シミュレーションおよび実験では，初期状態を，一部の物理ノードが一様に高負荷となった状態とし，その後，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．

表 2 に，シミュレーションおよび実験で用いたパラメータを示す．実験環境のサーバが持つコア数に合わせ，物理ノードあたりの仮想ノード数の上限を 16 とした．VNLB において，過負荷状態と見なす閾値 γ は，正しく全体負荷 L が予測された想定のもと，$\gamma = L \times 1.25$ とした．1.25 は，文

献 [7] の評価における閾値に合わせた値である．VNLB-ES の閾値設定は，既存の負荷分散手法（文献 [14] など）で用いられてきた値に合わせ，$\gamma = 0.75$，$\tau = 0.25$ とした．これらは，OS の正常動作が阻害される可能性がある閾値として設定された値である．初期値として与える全体負荷の合計は，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 まで変化させた．全体負荷の上限を 0.7 と設定したのは，$\gamma = 0.75$ では，全体負荷が 0.7 のとき，収束時に EVN が High 領域のみに存在する過負荷状態となり，VNLB と VNLB-ES が同じ挙動となるためである．また，上記設定では，VNLB は全体負荷が 0.8 以上のとき，物理ノードあたりの稼働仮想ノード数の比率の上限が 1 を超える．本評価では，物理ノード間の負荷調整の基本性能の評価を目的とし，split および redistribute は行わない設定とした．

本評価のシミュレーションでは，物理ノード数は可変とし，最大 1,000 まで変化させた．最大仮想ノード数は 16,000 である．また，シミュレーションの計測値としては，100 回の試行結果の平均値を採用した．シミュレータとしては，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．消費電力としては，図 4 で計測した値を用いた．

一方，実機を用いた実験では，著者らが有する 6 ラック，37 台[*1]のサーバを小規模データセンタに見立て，それらに提案アルゴリズムをインストールして評価を行った．本評価では，VNLB-ES の動作において収束状態は一意に決まる．VNLB においても，負荷分散の過程は複数ありうるが，収束状態はほぼ一意となる．後に述べるシミュレーション評価では，100 回のシミュレーション試行において，収束時の消費電力は，標準偏差が最大でも 0.03（kW）以下と小さな値であり，試行に応じた変化がほぼなかった．このため，実機を用いた計測結果としては VNLB，VNLB-ES ともに 1 回の試行の結果を用いた．処理としては，ネットワーク経由で映像取得と解析処理を実行するプロセスを実行した．このプロセスは，1 つのスレッドをほぼ 100%使用する負荷を発生させる．消費電力は，各サーバの電源を管理する Power Distribution Unit（PDU）経由で取得した．

5.2 負荷分散処理収束後の消費電力

図 6 は，シミュレーション，および，実機を用いた実験において，負荷分散処理が収束した後の消費電力を示している．本シミュレーションの物理ノード数は，実機で用いたサーバ数と同数（37 台）としている．Theoretical VNLB，Theoretical VNLB-ES は，それぞれ式 (1)，(2) に，図 4 の値を代入した理論値である[*2]．

[*1] 37 台は実験環境において確保可能な共有サーバ数の最大数．
[*2] 使用する CPU 数に応じた消費電力の増加量が計測では一定の割合ではないため，理論値は線形増加とはならない．

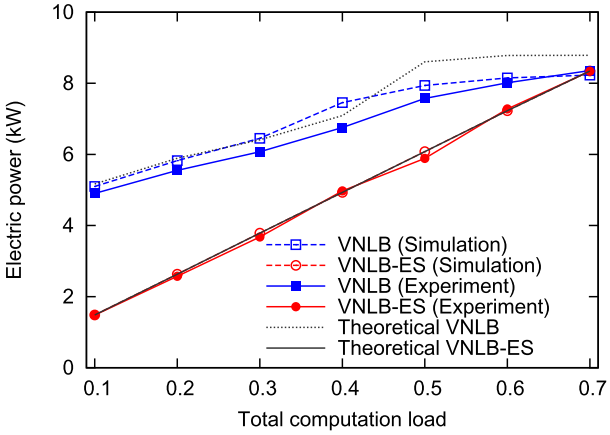

図 6 収束後の消費電力
Fig. 6 Electric power in the converged status.

VNLB では，すべての物理ノードの性能が同一のとき，物理ノード上で稼働する仮想ノード数が均一になるよう処理を分散させる．本評価環境では，物理ノード稼働台数はつねに 37 となる．ただし，物理ノード上で動作している仮想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．一方，VNLB-ES では，物理ノード上で稼働する仮想ノード数が 0 ならば，停止/一時停止状態となり，稼働台数は減少する．停止できる台数は，負荷が小さいほど多くなるため，特に負荷が小さいときに電力を削減できる．

図のとおり，VNLB，VNLB-ES それぞれ，シミュレーションと実機による実験における消費電力の値はほぼ一致する結果となった．ただし，シミュレーションで用いた電力計測時（図 4 の結果）の CPU 利用率と，実アプリケーション稼働時の CPU 利用率の違いより，消費電力に誤差が生じる．特に，使用する CPU コア数が少ないときに消費電力に差が生じるため，VNLB において，シミュレーションと実機による実験に誤差が出ていると考えられる．また，VNLB では，理論値と，シミュレーションおよび実機による実験結果との間に差が生じている．これは，理論値が完全に均等に負荷が分散された場合の値を示しているのに対し，実際の VNLB のアルゴリズムの動作では，稼働仮想ノード数の割合が γ 以下となったときに動作が終了するため，物理ノードの負荷に偏りが残り，稼働 CPU 数が少ない状態が残ったことによる．一方，VNLB-ES は，ほぼ理論値と同じ値となった．

図より，全体負荷量にかかわらず，VNLB よりも VNLB-ES が少ない消費電力となっていることが確認できる．特に，全体処理負荷が小さいとき，VNLB-ES は消費電力を大きく削減できている．

実機による実験において，VNLB では，全体負荷量が 0.1 のときの電力消費量は 4.9 kW であったのに対し，VNLB-ES では，1.4 kW となっており，およそ 70%の消費電力削減となった．

5.3 大規模データセンタにおける消費電力

大規模なデータセンタにおいても，前節の小規模環境と同様の消費電力の削減効果があることを示すため，1,000台のサーバが存在する環境を想定したシミュレーションを行った．**図 7** は，物理ノード数を 1,000 としたシミュレーションにおいて全体処理負荷を変化させたときの，負荷分散処理収束後の消費電力である．

図より，全体負荷に対する VNLB，VNLB-ES における消費電力は，前節の小規模環境とほぼ同様の傾向が得られることが分かる．最も負荷が低い条件である全体負荷量が 0.1 のとき，VNLB の電力消費量は 137.1 kW，VNLB-ES では，39.7 kW となった．すなわち，小規模環境と同様，本環境においても，VNLB と比べ，VNLB-ES は，約 70%の電力削減となった．これは，式 (3) に示したとおり，理想状態では，ある全体負荷において，VNLB と VNLB-ES の消費電力の比 R は，全体の物理ノード数によらず一定となることに合致する．

図 8 は，シミュレーションおよび実機を用いた実験において，全体負荷を変化させたときの R を示したものである．シミュレーション結果は，物理ノード数が 10，100，および，1,000 の場合を示している．図のとおり，シミュレーション，実機による実験において，物理ノード数の変化にかかわらず，各全体負荷において，物理ノード数によらずほぼ同等の消費電力の比となった．ただし，物理ノード数が 10 の場合，全体負荷が 0.7 のときに R の値は若干 1 を超えている．これは VNLB-ES が物理サーバ単位で停止/一時停止させる動作をするために生じた誤差である．

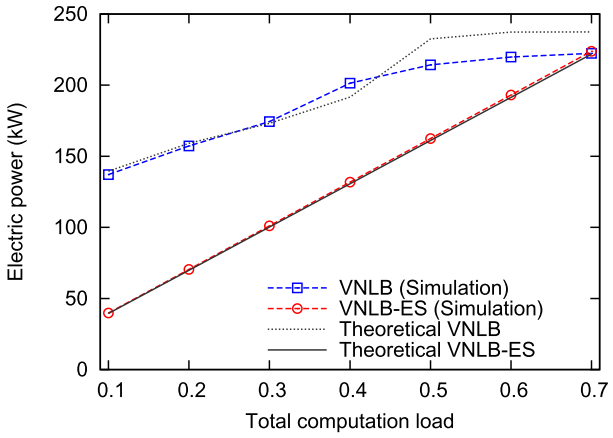

図 7 大規模環境における収束後の消費電力

Fig. 7 Electric power in the converged status on a large-scale environment.

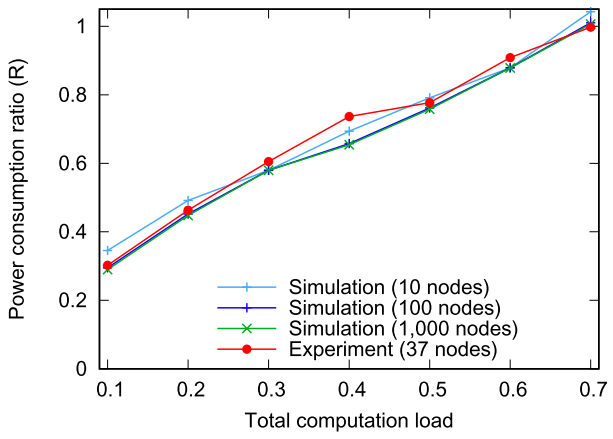

図 8 VNLB と VNLB-ES の消費電力の比

Fig. 8 Power consumption ratio of VNLB and VNLB-ES.

5.4 消費電力の時間推移

本実験に用いた実装においては，物理ノード上での過負荷・低負荷の判定は，自律的に動作する仮想ノードからの通知に基づくイベントドリブンによって行う．したがって，実際に仮想ノード数の変化が生じてから，過負荷・低負荷の判定までにかかる時間は小さい．また，swap 操作を行う際，Live Migration は行っておらず，プロセスの起動と終了，およびパラメータの引き継ぎを行っており，1 秒以内のオーバヘッドで swap は実行可能である．一方，実験に用いたサーバは起動開始から完了までには 270 秒前後，停止には 2 秒程度かかる．上記環境のもと，VNLB，VNLB-ES において，消費電力が収束するまでにどの程度の時間を要するか計測した．

図 9，および，**図 10** は，実機 37 台を用いた実験における消費電力の時間推移である．全体負荷が小さい場合と，比較的大きい場合の挙動の変化を確認するため，全体負荷量が 0.1（10%）の場合（図 9），および，0.6（60%）の場合（図 10）それぞれの時間推移を調べた．X 軸は経過時間（秒），Y 軸は消費電力（kW）である．

図 9 によれば，VNLB は，swap 操作が行われたにもかかわらず，時間経過によらずほぼ一定となった．一方，VNLB-ES は，swap 操作にともない，物理ノードが停止状態に移行するため，時間が経過するに従って消費電力が減少した．図に示したとおり，およそ 360 秒経過した時点

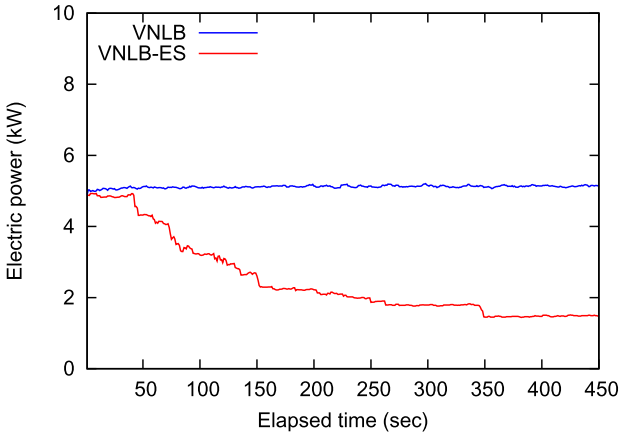

図 9 10%処理負荷時の消費電力の時間推移

Fig. 9 Time transition of the electric power when computation load is 10%.

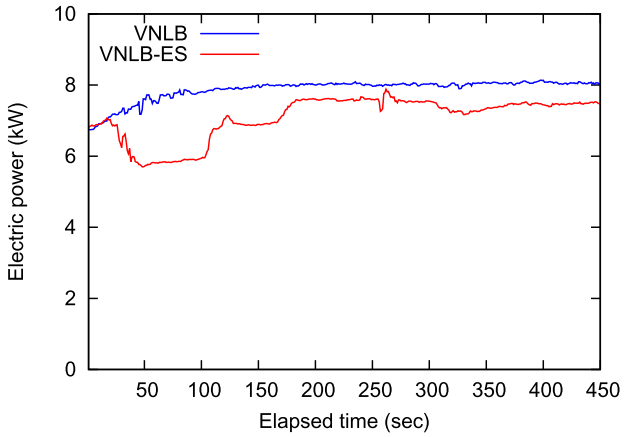

図 10 60%処理負荷時の消費電力の時間推移
Fig. 10 Time transition of the electric power when computation load is 60%.

で，1.4 kW 程度の消費電力に収束している．450 秒経過時点の消費電力は，VNLB は 4.9 kW，VNLB-ES は 1.4 kW となった．

図 10 においては，VNLB の消費電力は初期段階で微増し，8 kW 程度で安定状態となった．この消費電力の微増は swap 操作により比較的高い処理負荷の状態の物理ノードが増えたことによる．一方，VNLB-ES では，初期状態で仮想ノードのすべてが稼働状態ではない物理ノードは一時停止状態に移行したが，過負荷状態にあるノードからの swap によって Moderate 状態となる物理ノードもあった．すなわち，負荷の集約と分散が同時に起こった．このため，集約にともなう物理ノードの停止と，負荷を分散させるための新たな物理ノードの起動動作が混在し，時間経過にともない全体の消費電力の増減が起きている．物理ノード起動時の初期段階にかかる電力は定常状態よりも高いため，物理ノードが起動するとき，少量の電力消費増が起こっている（130 秒付近など）．VNLB-ES は，およそ 360 秒経過した時点で，7.5 kW に収束した．

本実験環境では，VNLB-ES では消費電力の収束にかかる時間は，いずれの場合もおよそ 360 秒程度となった．これは，1 度処理負荷の変化が起きると，しばらく変化しないイベント向けのアプリケーションなどでは十分な時間と考えられる．しかし，たとえば，ウェブ経由の処理など，処理の内容の変化が短時間に頻繁に起きるアプリケーションでは不十分な可能性がある．この課題に対応するには，起動済みの空物理ノードを予約状態としていくつか用意しておき，停止/一時停止状態と起動状態の移行を高速化するなどの方策が考えられる．

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物理ノードを停止/一時停止させることで電力消費の削減が可能である．シミュレーションおよび実機を用いた実験によって，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 70%削減可能となることを確認した．

しかしながら，現状の VNLB-ES には，前節でも述べた頻繁な負荷の変動があるアプリケーションへの対応のほか，いくつか改善すべき点も残されている．たとえば，Sleep 領域に属する仮想ノードのポインタを単一の物理ノードが保持するため，このポインタを保持する物理ノードがシステム全体の単一障害点となりうる．この課題を解決するためには，ポインタ情報を Low 領域に存在する仮想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，たと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．これらの機能の実現方法は今後の課題である．

今後，上記課題への対応や，収束にかかる時間の理論的な見積りなどのさらなる解析，実装の汎用化，長期稼働する実アプリケーションを考慮した対応など，より高度な省電力化方式としての検討を進める．

参考文献

[1] Felber, P., Kropf, P., Schiller, E. and Serbu, S.: Survey on load balancing in peer-to-peer distributed hash tables, *IEEE Communications Surveys & Tutorials*, Vol.16, No.1 (2014).
[2] DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P. and Vogels, W.: Dynamo: Amazons Highly Available Key-value Store, *Proc. SOSP '07* (2007).
[3] Hadoop Distributed File System (online), available from ⟨http://hadoop.apache.org/hdfs⟩.
[4] Nakagawa, I. and Nagami, K.: Jobcast – Parallel and distributed processing framework: Data processing on a cloud style KVS database, *Journal of Information Processing*, Vol.21, No.3 (2013).
[5] Ganesan, P., Bawa, M. and Molina, H.G.: Online Balancing of Range-Partitioned Data with Applications to Peer-to-Peer Systems, *Proc. VLDB '04* (2004).
[6] Konstantinou, I., Tsoumakos, D. and Koziris, N.: Fast and Cost-Effective Online Load-Balancing in Distributed Range-Queriable Systems, *IEEE Trans. Parallel and*

Distributed Systems, Vol.22, No.8 (2011).
[7] Shao, X., Jibiki, M., Teranishi, Y. and Nishinaga, N.: Effective Load Balancing Mechanism for Heterogeneous Range Queriable Cloud Storage, *Proc. IEEE 7th International Conference on Cloud Computing Technology and Science* (*CloudCom 2015*), pp.405–412 (2015).
[8] Aspnes, J., Kirsch, J. and Krishnamurthy, A.: Load balancing and locality in rangequeriable data structures, *Proc. PODC '04* (2004).
[9] Ganesan, P., Bawa, M. and Molina, H.G.: Online balancing of range-partitioned data with applications to peer-to-peer systems, *Proc. NDB '04* (2004).
[10] Vu, Q.H., Ooi, B.C., Rinard, M. and Tan, K.L.: Histogram-based global load balancing in structured peer-to-peer systems, *IEEE Trans. Knowledge Data Engineering*, Vol.21, No.4, pp.595–608 (2009).
[11] Konstantinou, I., Tsoumakos, D. and Koziris, N.: Fast and cost-effective online loadbalancing in distributed range-queriable systems, *IEEE Trans. Parallel Distributed Systems*, Vol.22, No.8, pp.1350–1364 (2011).
[12] Schütt, T., Schintke, F. and Reinefeld, A.: Range queries on structured overlay networks, *Computer Communications*, Vol.31, No.2, pp.280–291, Elsevier Science Publishers B.V. (2008).
[13] Kapil, D., Pilli, E.S. and Joshi, R.C.: Live virtual machine migration techniques: Survey and research challenges, *The 3rd IEEE International Advance Computing Conf.* (*IACC 2013*), pp.963–969 (2013).
[14] Galloway, J.M., Smith, K.L. and Vrbsky, S.S.: Power aware load balancing for cloud computing, *Proc. World Congress on Engineering and Computer Science*, Vol.1, pp.19–21 (2011).

木全 崇 （正会員）

2011 年大阪市立大学大学院創造都市研究科博士前期課程修了．同年情報通信研究機構ユニバーサルコミュニケーション研究所入所．2016 年より同総合テストベッド研究開発推進センター研究技術員，現在に至る．分散システム，オーバレイネットワーク，仮想ネットワーク，コネクテッドカー，およびその応用システムに関する研究開発に従事．

寺西 裕一 （正会員）

1995 年大阪大学大学院基礎工学研究科博士前期課程修了．同年日本電信電話株式会社入社．2005 年大阪大学サイバーメディアセンター講師．2007 年同大学大学院情報科学研究科准教授．2011 年より情報通信研究機構研究マネージャーおよび大阪大学サイバーメディアセンター招へい准教授，現在に至る．分散システム，オーバレイネットワーク，センサネットワークおよびその応用システムに関する研究開発に従事．2010 年度本会論文賞受賞．博士（工学）（2004 年 3 月，大阪大学）．電子情報通信学会，IEEE 各会員．

細川 貴史

1996 年京都大学大学院工学研究科博士前期課程修了．同年日立製作所システム開発研究所入所．1999 年郵政省（現，総務省）入省．2017 年総務省電波政策課企画官．2018 年より情報通信研究機構総合テストベッド研究開発推進センター統括，現在に至る．テストベッドによる研究開発の推進に従事．

原井 洋明

1998 年大阪大学大学院基礎工学研究科博士後期課程修了．同年通信総合研究所（現，情報通信研究機構）入所，2011 年情報通信研究機構ネットワークアーキテクチャ研究室長等を経て，2018 年より同総合テストベッド研究開発推進センター長，現在に至る．革新的ネットワーク，光ネットワークに関わる研究開発，および，テストベッドによる研究開発の推進に従事．博士（工学）．電子情報通信学会，IEEE 各会員．

下條 真司 （正会員）

1986 年大阪大学大学院基礎工学研究科博士後期課程修了．1987 年同大学助手．1989 年同大型計算機センター講師．1991 年同助教授．1998 年同教授．2000 年同大学サイバーメディアセンター副センター長，2005 年同センター長，2007 年同副センター長，2008 年より 3 年間情報通信研究機構大手町ネットワーク研究統括センター長・上席研究員．2011 年大阪大学サイバーメディアセンター教授．2015 年より同センター長．現在に至る．マルチメディアシステムアーキテクチャ，P2P・グリッド，次世代インターネット等の研究開発に従事．2005 年大阪科学賞．2006 年総務大臣表彰．2010 年度本会論文賞受賞．博士（工学）．電子情報通信学会フェロー，IEEE CS 会員．本会フェロー．