Vol.2022-DPS-193 No.11
2022/12/8

複数の時間間隔に基づくオーバレイネットワークにおける適応的仮想ノード管理手法の検討

久保 達也[1]　川上 朋也[1]

概要：センサデバイスの低廉化が普及の背景にある IoT では，位置や時間と密接に関連付いた情報が大量に扱われる．大量のセンサデータを高スケーラビリティで扱うための手法として，筆者らは特に時間の結びつきに着目してクエリを効率良く扱う手法を提案している．負荷の集中を避けるために仮想化を行い，ルーティングのアルゴリズムを変更することにより効率化を図る手法を提案したが，効果が限定的であった．また，実際の P2P ネットワークの特徴であるデバイスの性能差を考慮していなかった．そこで本稿では仮想ノードの数は一定という前提を変えて仮想ノードの数を適応的に割り当てる手法と，クエリをまとめて処理しさらにメッセージ数を削減する手法を検討する．前者は各物理ノードが持つ性能指標とネットワーク全体での目標とするノード数に基づいて，仮想ノード数を各物理ノードが独立して決定する．後者はこれまで考慮していなかったクエリ間の繋がりを処理で考慮して既存手法より更なる効率化を行う．複数の条件下で，処理能力に対し適応的な割当が機能できるかおよびメッセージ数の削減が行えるかについて評価を行った．

1. はじめに

近年の急激なデバイス数の増加そして回線品質向上に伴い急速な成長を続けるインターネットは，その中でもとりわけセンサデバイスの低廉化や社会からのニーズが高まりを受けた Internet of Things（IoT）分野の発展が著しい．これに伴う大量かつ流動的なセンサデバイス及び測定データからなるネットワークやサービスを取り扱うには，従来のクライアントサーバモデルでは負荷の上昇に耐えられなくなることが考えられる．負荷が高くなりすぎるとやがてサーバはクエリを処理しきれなくなり障害が発生し，サービスの可用性を低下させることとなる．対策としてはサーバの設備増強が考えられるるが，しかし年に十数%と予想されるクライアント側（IoT 機器）の増加に追いつくことは技術的にも経済的にも限界がある．

そこで，この問題の解決策として考えられるのが Peer to Peer（P2P）方式によるネットワークである [1]．P2P はネットワーク上で対等なノード（端末）が協調して処理を行うモデルであり，中央集権的なサーバを持たずにサービスを構築することができる．また，P2P はインターネットでの繋がりとは関係なく仮想的なネットワークを構築するため，扱うデータの性質などに応じてネットワークの構造を変えられるという利点がある．

IoT で扱うデータは，温度や電力消費量のようなセンサデータが多いことが特徴として挙げられ，これらは時間や位置情報に強く結びついている．そしてこの性質を利用した手法が多数提案されているが，利用者は「過去の 8 月の気温」「毎週日曜日の電力消費量」のような一定間隔を開けた複数のデータを一度に要求することが考えられる（図 1）．このように特定の時間間隔を指定するデータの要求のことを「間隔クエリ（interval query）」と定義し，これらを効率的に扱うことができる構造化オーバレイとして，Chord [2] を基とした複数の時間間隔を扱うことができるオーバレイネットワークについて研究を行っている [3, 4]．

2. 関連研究

2.1 オーバレイネットワーク

サーバを持たない P2P では，自立した各ノードが IP ネットワーク上に論理的な異なる構造のネットワークを形成する．このようにアプリケーション層で作られるネットワークはそれより下層の IP ネットワークに覆いかぶさるように仮想化していることからオーバレイネットワーク

[1] 福井大学大学院工学研究科
Graduate School of Engineering, University of Fukui

（Overlay Network）と呼ばれる．これらは仮想のものであるため既存のネットワークに制約を受けずに機能を付加することができるという特長がある．

オーバレイネットワークには，主に構造化と非構造化の二種類がある．その中で構造化オーバレイネットワークは，環状や木構造といった数学的な規則に基づいて予め決められたトポロジに従ってネットワークを構築するオーバレイネットワークである．ネットワークの構造が明確であるため，データを検索するときに検索先のノードを一意に特定することができる長所がある．一方，ノードの追加などネットワークが変化した際には規則を保つためその都度修正を行う必要があり維持コストがかかる．そのため，ノードの頻繁な変化や，ノードの障害といった予測できない変化には弱くなる．

なお，オーバレイネットワーク上で行われた通信を実際に送信する際にはノードの IP アドレスに基づいて実際のネットワーク（通常はインターネット）上で通常と同じルーティングを行う．そのため，オーバレイネットワーク上では最短経路の効率良い通信が行えていても，実際はネットワーク上で遠回りをして過剰なトラヒックを発生させている可能性がある．このようにオーバレイネットワークには実際のネットワークとの齟齬が必ず存在している [5]．

2.2 DHT および Chord

分散ハッシュテーブル（Distributed Hash Table, DHT）は，構造化オーバレイを構成するための手法の一つである．DHT ではネットワーク上のノードと入力されるデータはすべて同じ空間上のユニークなキー（識別子，ID）を割り当てられる．具体的には，SHA-1 などのハッシュ関数を使ってデータあるいはノードの IP アドレスのハッシュ値を算出し，キーとして用いる．そのキーの値によってノードはネットワークのどの位置に配置されるかが定められ，データはどのノードに格納されるかが決定される．データを検索する際にはクエリからハッシュ値を算出し，そこから問い合わせるべきノードを特定する．ハッシュ関数を利用することでどのようなデータも同じキー空間上に配置でき，かつデータを均等に分布しノード間の負荷が分散できるという利点がある．

DHT のアルゴリズムは様々なものがあるが，本研究においては Chord [2] をベースとしたネットワークを提案している．Chord のネットワーク上の各ノードは，自身よりキーが大きくかつ最も近いキーを持つノードを Successor，反対にキーが自身より小さい最初のノードを Predecessor としてリンクする．そして最大のキーを持つノードの Successor が最小のキーを持つノードに戻ることにより，環状のネットワークを形成する．これに加えて，ショートカットのために離れた場所にあるノードの情報を Finger Table として保持する．データを入力する際には，データのキーより

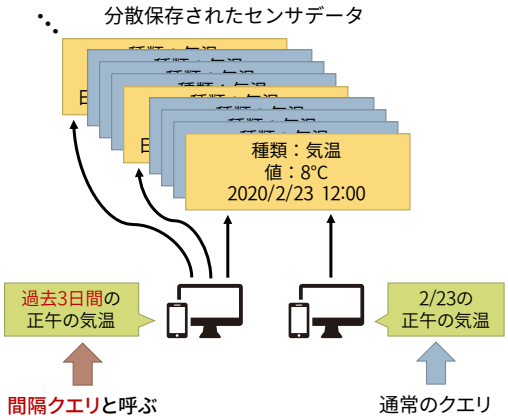


図 1: 間隔クエリの概念図

キーが大きくかつ最も近いノードが担当となる．すなわち各ノードは Predecessor との間のキーを持つデータを格納する．Chord は環状のネットワークであるため常に一方向にデータを送信すれば目的のノードに到達することができ，末端部などでの特別な処理も不要で簡素なアルゴリズムで実装できるという特長がある．

2.3 複数の時間間隔に基づくネットワーク

IoT において中心的存在となるセンサデータは高頻度で更新され位置・時間の情報に深く結びついているという特徴がある．センサデータの時間的な結びつきにより，ネットワークの利用者（またはシステム）はデータをある特定の時間間隔という条件で複数要求することが考えられる．例としては「去年までの 7 月の気温」や「毎週日曜日の電力消費量」といったものである．このような複数のクエリで構成されるクエリをまとめて間隔クエリと呼び，概要を図 1 に示す．間隔クエリは過去のデータによる統計情報を用意に得ることを可能とし，ビッグデータ等に有用であると考えられる．

Chord を間隔クエリを効率良く扱えるよう改良するには，まず Finger Table を間隔クエリを想定した週，月，年といった利用者から要求されることが想定される時間単位のものとする．また，入力データのハッシュ値を取るという DHT の機能を取り除いている．2.2 章で述べた通り，DHT においてはハッシュ関数が値を均等に分布させる性質を長所として活かしている．しかし，その性質は同時に似たようなデータでも全く異なるノードに配置される可能性があるということでもある．この性質は連続したデータを順番に取り扱う間隔クエリにおいてはランダムアクセスを増加させ効率を悪化させるため，時刻をキーとしてそのままネットワークへ格納する方法を取っている．このような類似例としては Chord$^{\#}$ [6] がある．

2.4 ノード仮想化とルーティング

前節で述べたとおり，複数の時間間隔に基づくネットワークではハッシュ値を使用しない．そのままでは負荷分散のための仕組みが存在しないため，入力データの偏りがそのままネットワークに反映されて負荷が集中してしまう．そこで，代替の負荷分散としてノードの仮想化を導入した [4]．オーバレイネットワークにおけるノードの仮想化とは，ある端末（以下，物理ノードと呼ぶ）がネットワーク上に 1 つだけではなく複数のノード（仮想ノード）を持つことである．これによって一つの物理ノードが担当するキーの範囲が広がり，仮想ノードの総数が十分かつ均等に割り当てられていれば確率的に負荷は分散されていく．ハッシュ値によってネットワークの入力データを分散するのではなく，逆にネットワークの側が分散することによって同様の効果をもたらすことを狙ったものといえる．

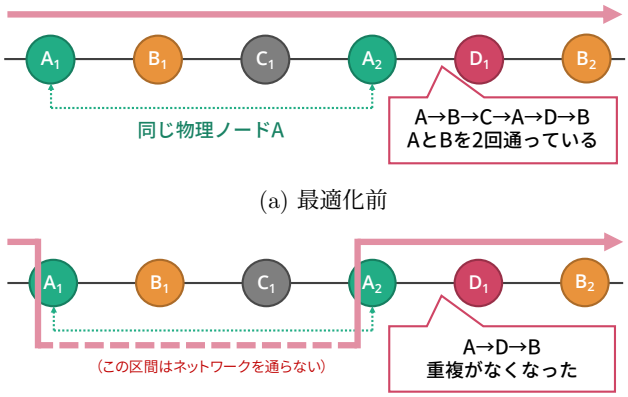
一方で仮想化を行うことの欠点としてはネットワーク上のノード数が増加することである．ノード数が増えればそれだけネットワークの規模が拡大し，同じクエリでも多くのノードを超えて遠く伝達しなければならなくなる．特に仮想化されたネットワークにおいてはネットワーク上に仮想ノードのみが存在することとなり，物理ノードの存在は考慮されない．そのため，ネットワーク上では最短経路での最適な通信が行われているように見えても，実際には同じ物理ノードの間で繰り返し行われてる冗長な通信が発生していることが考えられる．これを軽減するために，仮想ノードを前提とするルーティングの最適化を行った [3]．

具体的に図 2 に示す．ここで円の中のアルファベットは物理ノードの名前，添字は同じ物理ノードの持つ仮想ノードの番号である．どちらもクエリを仮想ノード上を重複なく転送する過程を表したものだが，図 2a では同じ物理ノード A と B を二回通過している．これでは同じ場所に二回もクエリが転送されることになり冗長な通信である．そこで図 2b のように仮想ノードだけでなく物理ノード全体から次のノードを探しルーティングを行うよう拡張した．この手法により，個々のノードの負荷を分散させつつネットワーク全体への負荷を減少させることができた．

3. 適応的な仮想ノード割当

3.1 ノードの性能差について

ここまで述べたオーバレイネットワークに仮想ノードを導入する方法は，全て 2.4 節で述べた通り，負荷を均等にすることを目指す前提でありすべてのノードに同じ仮想ノード数を割り振るとしていた．しかし，P2P ネットワークは管理者の管理下に置かれる少数のサーバではなく無数のデバイスにより構成されている．そのためサーバのようにサービスを提供するために計画的に導入されているのではなく，計算処理能力，ネットワークの帯域幅，ストレージ容量などの性能は各ノードごとに異なっている [7]．P2P の



(a) 最適化前

(b) 最適化後

図 2: 物理ノード全体によるルーティング

性能指標についての問題は複雑であり，例えばモバイル端末は据え置き型の端末と異なりバッテリー駆動であるため電力の資源が有限であるという新たな制約がある [8]．すなわち，負荷を均等に分散することは負荷を平等に分散することとは必ずしも同じではないということである．そのため均等に割り当てても物理ノードによっては過負荷になったり，逆に性能を持て余す状態になることが考えられる．

3.2 仮想ノードの割当数

これまでの前提をより現実的な条件に進めるため，物理ノードごとの性能が異なる場合を想定した環境を導入することを検討する．性能評価指標には様々なものがあるが，ここでの指標は「持てる仮想ノードの数」で表されるとする．物理ノードは例えば処理性能や帯域幅を基に，あらかじめ持てる仮想ノードの数を見積もった状態でネットワークに参加する．そしてこの上限を超えないように高性能の物理ノードに多く，低性能の物理ノードに少なく仮想ノードを配置していく．このような方法は単純な二分化のモデルであるが，多くの場合では十分に性能を反映できるとされている [9]．このように各物理ノードが持てる仮想ノードの数を保持しているが，仮想ノードの数が多すぎるとオーバーヘッドが発生するためこれらの数全体まではノードを持てない．代わりに，定められた目標の個数だけの仮想ノードがネットワーク上に存在するように調整する．

ここからは具体的な割り当て方にについて説明する．今，n_k 個の仮想ノードを持てる物理ノード V_k がネットワークに参加するとする．V_k はパラメータとしてネットワークの目標ノード数 T および現在ネットワーク上にいる仮想ノードの合計数 N_{Sum} を受け取る．このとき式 1 が V_k が持つ仮想ノード数を示す．

$$\frac{n_k}{N_{\mathrm{Sum}}} \cdot T \tag{1}$$

実際に持てる仮想ノードの数は整数であるため，答えが実数となる式 1 の値を整数に丸める必要がある．ここで

は，切り捨て（式 2），切り上げ（式 3），四捨五入（式 4）の三種類について検討した．

$$V_k^F = \left\lfloor \frac{n_k}{N_{\mathrm{Sum}}} \cdot T \right\rfloor \tag{2}$$

$$V_k^C = \left\lceil \frac{n_k}{N_{\mathrm{Sum}}} \cdot T \right\rceil \tag{3}$$

$$V_k^R = \left\lfloor \frac{n_k}{N_{\mathrm{Sum}}} \cdot T + 0.5 \right\rfloor \tag{4}$$

なお，使用する変数の中でも特に N_{Sum} はネットワーク全体を表す指標であり，分散環境では直接得ることができないものである．Chord を含む多くのオーバレイネットワークでは経路表を維持するために他ノードとの定期的な通信が必要であり，これを利用して取得することを想定している．新ノード追加のための処理を行った際に数値を加算し，経路表の定期更新の際に同時に他のノードとノード数についての情報を交換し合う．

4. 間隔クエリの統合的な処理

ここまで物理ノードの性能について検討を行ってきたが，文献 [3] において挙げた今後の課題であるルーティングのさらなる効率化についても取り組んだ．第 2.4 節に示したように，これまで行ってきたルーティングの最適化はあくまで単一のクエリの転送経路内での重複を防ぐことを目指したものであった．しかし間隔クエリは複数の連続したクエリで構成されるものである．一つのクエリ内で冗長な通信を防ぐことは既にできているが，その後のクエリでまた同じ物理ノードが担当するデータである可能性もあり，ここで防げていない冗長な通信が残っていると考えられる．これに対処するため，複数のクエリにまたがって内容を先読みするという仕組みを導入した．

概要を図 3 に示す．この図は 5 つのクエリで構成される間隔クエリであり，数字がキーすなわち時刻を示す．時刻を時間単位とすると，2 時間ごとのデータを要求していることとなる．今，物理ノード A の仮想ノード A_1 がこの間隔クエリを受信し，A は他に仮想ノード $A_2, A_3, A_4, \cdots$ を持っているとする．これまでは，

(1) A は $A_1, A_2, A_3, \cdots$ のいずれかが先頭のクエリ 1 の担当範囲であるかどうかを判定する
(2) 担当範囲であればクエリに応答して，間隔クエリから取り除く
(3) 先頭のクエリを $A_1, A_2, A_3, \cdots$ のどこから送信すれば最も効率が良いか判定し，送信

という手順を踏んでいた．

しかし，例えば後にあるキー 7 のデータを A_3 が担当していた場合，ここでは処理できるクエリであることに気付かれずまた A に間隔クエリが戻ってくることになる．この冗長な通信を削減するために，上の手順 1 を以下のように変更する．

A は $A_1, A_2, A_3, \cdots$ のそれぞれが間隔クエリの各要素（$1, 3, 5, \cdots$）の担当範囲であるかどうかを判定する

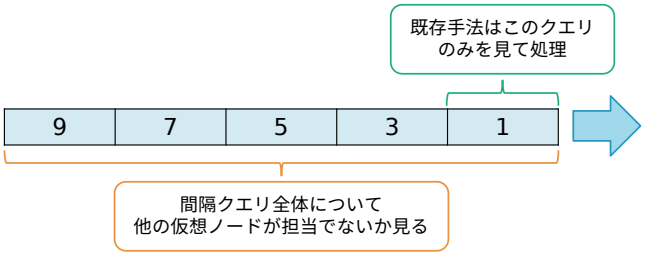


図 3: 間隔クエリの先読み

この操作によりクエリが 2 回同じ物理ノードを通ることはなくなる．

なお，それ以降の手順にあるルーティング先は同じとしている．すなわち，間隔クエリの後方に先読みできる箇所があっても次は間隔クエリの先頭のクエリを目的地として送信を行い，他のクエリを取り出して送信先にはしない．これは間隔クエリは先頭からキーの小さい順に並んでおり，かつこのネットワークもキーの小さい順からノードを並べた構造であるため，常に先頭から送信したままのほうが効率が良くなるためである．

また，仮想ノードの配置の方法についても変更を行った．これまでは物理ノードが仮想ノードを置く際にはキー空間上のランダムな位置に配置することとしていた．しかし間隔クエリの起点から終点までの長さはキー空間全体の大きさに比べかなり小さくなり，別の仮想ノードが先のデータを持っているという状況が起きにくいと考えられる．そのため，物理ノードはまず最初に 1 つの仮想ノードをネットワーク上にランダムに配置し，残りの仮想ノードはその最初のノードからネットワークの反対側までの範囲内のランダムな位置に配置する方法を導入した．つまり，最初のノードのキーを K，キー空間のサイズを n ビットとすると，残りの仮想ノードのキーは K から $K + 2^{n-1}$ の範囲のどこかになる．この変更により同じ物理ノードが持つ仮想ノードに近接性ができて間隔クエリの先読みが機能しやすくなることが期待できる．

5. 評価

ここでは物理ノードの性能を考慮したオーバレイネットワークのシミュレーションにより提案手法の有用性を示すための評価を行った．シミュレータは Java を用いて実装し，本章ではシミュレーション環境とその結果について述べる．

5.1 適応的な仮想ノード数割当

物理ノードの性能差に応じた仮想ノード割当ができるか

表 1: 各分布で持てる仮想ノードの数

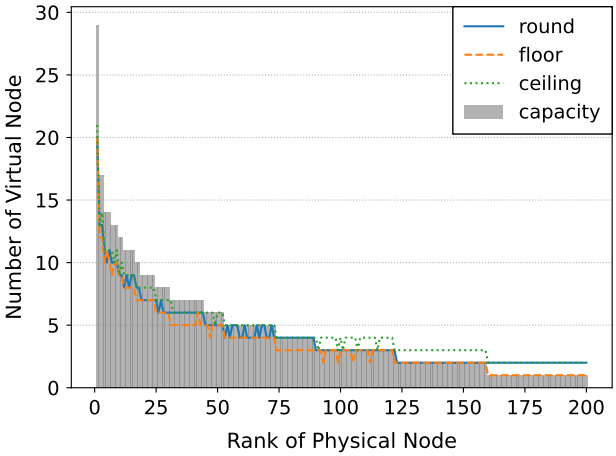
	指数分布	正規分布	一様分布
容量の合計	874	879	873
ノードごとの平均	4.37	4.39	4.37
予定量に対する余裕	9.25%	9.87%	9.13%

どうか検証するために，200 個の物理ノードに 800 個の仮想ノードを割り当てることを想定した実験を行った．既存手法であれば，全ての物理ノードに 4 個の仮想ノードを割り当てることとなる．200 個の物理ノードは，指数分布，正規分布，一様分布の 3 つの確率分布に従っているとする．いずれも 800 個の仮想ノードを全体で持てるように平均値が 4 より少し大きくなるようにパラメータを調節した．表 1 に各分布における合計容量であり，目標の 800 個に対し概ね 9%の余裕がある．よって，適切な割当を行えば全てのノードの負荷を容量以下に抑えて過負荷を起きないようにすることができる条件である．そして，全ての分布に対し仮想ノードの割当を決定する式 2 から式 4 に対応する四捨五入，切り捨て，切り上げの 3 つの丸めについて評価した．この式のうち N_{Sum} については正確な値を把握することは難しいと想定し，真の値から 5%の誤差を持たせた値を各ノードは取得するとした．

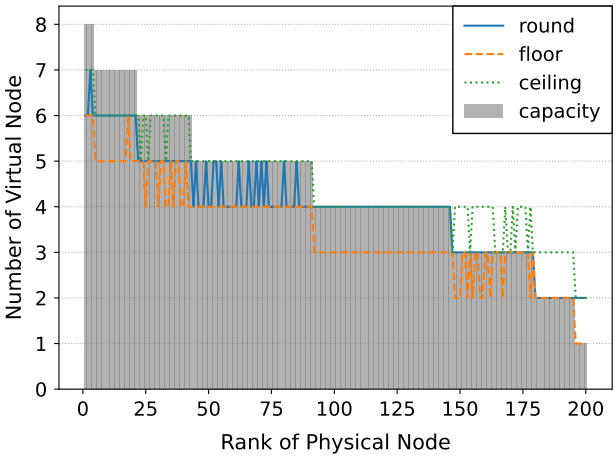
5.1.1 物理ノードに割り当てられた数

図 4 に 3 つの分布それぞれの物理ノードの容量，及び実際に割り当てられた仮想ノード数を示す．グラフの横軸が物理ノード一つ一つに対応しており，容量の大きいものから小さいものへ並べられている．縦軸は仮想ノード数で，背景にある灰色の棒が各物理ノードの容量を，3 本の折れ線がそれぞれの四捨五入（round），切り捨て（floor），切り上げ（ceiling）による実際の割当数を示す．なお，これらのグラフでは同様に見やすさのために既存の手法については省略した．既存手法であれば，全てのノードにこの場合は 4 個の仮想ノードが割り当てられるため縦軸 4 に横線が引かれることになる．

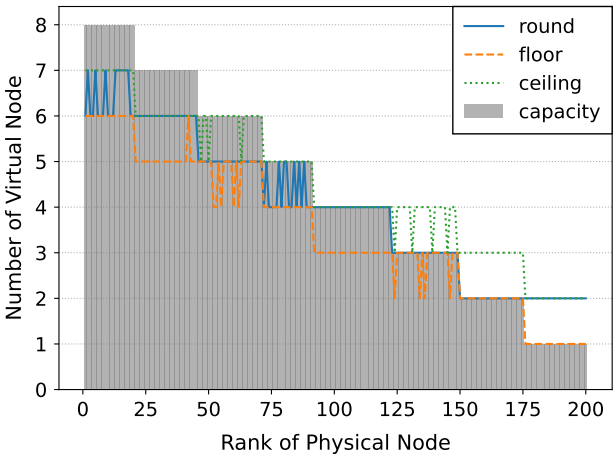
図 4a は物理ノードの容量が指数分布に従ったときのものである．この分布は一部のノードが高い性能を持つ一方で大多数のノードの性能は低くなり，パレートの法則のように一部の高性能なノードが性能のほとんどを占めている状態である．いずれの丸め手法でも概ね分布に沿った割当が行われていることが分かる．50〜125 位の物理ノードの間で N_{Sum} の誤差による仮想ノード割当のばらつきがあるが，ノードの負荷を大きく上昇させたりはしていない．丸めの方法の中でもノード数を多めに割り振ることになる切り上げは，100 位くらいから常にノードの容量よりも 1 多く割り振り続けており，負荷を規定以下に保つ点では四捨五入や切り捨てのほうが優れている．特に四捨五入は性能上位のノードでも多めにノードを割り振っており性能の有効活用という点でも優秀である．一方で下位の容量が 1 のノードに対しても 2 つの仮想ノードを割り振り続けており，この区間で容量に適応できているのは切り捨てのみだった．



(a) 指数分布



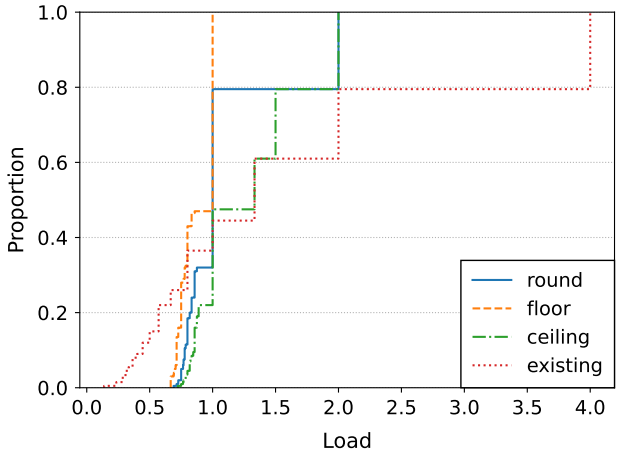
(b) 正規分布



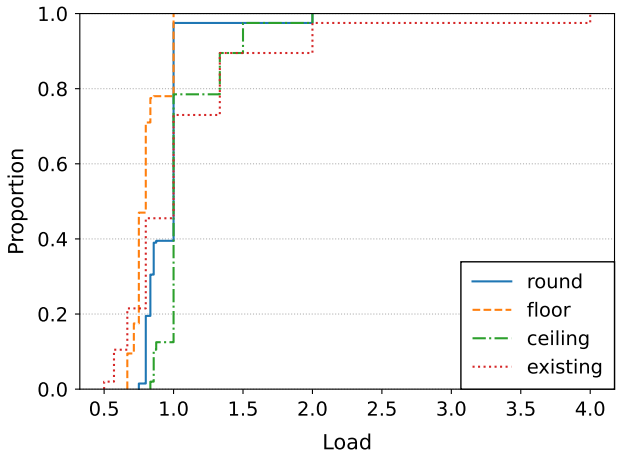
(c) 一様分布

図 4: 物理ノード性能と割当数の関係

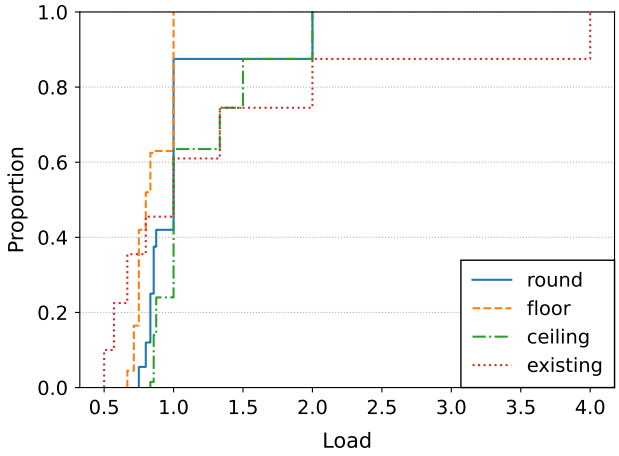
図 4b は物理ノードの容量が正規分布に従ったときのものである．指数分布に比べ分散が小さいため性能差は小さく，また平均値付近の容量のノードが増加している．図 4a



(a) 指数分布



(b) 正規分布



(c) 一様分布

図 5: 物理ノードの負荷の分布

と同様に，性能下位のノードでは切り上げ，そして四捨五入は割当の超過が起きている．一方で，高性能なノードに対しては切り上げが最もノードの性能を有効活用できていることがわかった．図 4c は一様分布のものであるが，正規分布のときとほぼ同じ傾向となっている．

表 2: 丸め手法別の実際の割当数

	四捨五入	切り捨て	切り上げ
指数分布	801	705	903
正規分布	802	694	897
一様分布	803	698	898

5.1.2 物理ノードの負荷

図 5 は物理ノードの負荷の分布を表したものである．横軸は（割当数）/（物理ノードの容量）で表される比率で，縦軸は累積比率である．負荷が 1 を超えると過負荷となるためそれを抑えることが重要である一方で，負荷が 0 に近いほどノードの性能を有効に活用できていないということにもなる．そのため，超えないようにしつつなるべく負荷を 1 に近づけることが理想となる．また，ここでは既存手法（existing）として全てのノードに一定の仮想ノード（ここでは 4 個）を割り当てる場合でも実験を行った．

図 5a の指数分布は偏りが大きいため特に提案手法の効果が出ていることがわかる．既存手法では 5 割以上のノードが過負荷になっている一方で四捨五入は 2 割，切り捨てであれば過負荷のノードはなくなった．高性能なノードについては，既存手法が負荷が 5 割以下となっている箇所も多い中で提案手法は 7 割以上の負荷がかかっており，性能を大きく引き出せていることがわかる．一方で性能の差が小さい正規分布（図 5b）や一様分布（図 5c）では，ノード数を少なめに割り振る切り捨てが既存手法よりも利用率が低くなっている箇所があった．ここでもう一度図 4a と図 4c を見ると，容量が 4 の箇所で切り捨てだと 3 個しか割り振られていないことがわかる．既存手法では一律に 4 個を割り振るためここでは既存手法のほうが利用率が高くなっている．特に正規分布では他の手法に比べて少ない 20%ほどのノードしか負荷が 1 になっていない．よって，切り捨ては負荷を抑えるためには有効だが，利用率という観点からは効率が落ちることがわかった．

この実験では 800 個の仮想ノードを目標にしたが，丸めの方法により多め少なめが生じており，それを表 2 に表す．四捨五入はほぼ目標とする 800 個のノードを割り当てたが，切り捨ては 87%切り上げは 112%とそれぞれが少なめ，多めという結果となった．

5.2 間隔クエリの統合的な処理

ルーティングの効率化手法については，文献 [3] と同じ条件での実験を行った．ネットワーク上の各物理ノードが 1, 2, 4, 8, 16, 32 個の仮想ノードを持ったとき，物理ノード数を 250, 500, 750, 1000 個と変化させてメッセージ数を観測した．なお，仮想ノードがないときには既存手法と提案手法の結果は全く同じになる．間隔クエリを 20 回送信し，転送にかかったメッセージ数の平均を評価指標とする．間隔クエリの起点としては，一様分布および Zipf 分布 [10]

表 3: メッセージ数の実験条件

パラメータ	値
キーの単位時間	1 時間
キー空間の大きさ	16 ビット (65536 時間)
各仮想ノードのキー	一様なランダム選択
仮想と物理の関係	一様なランダム選択
間隔クエリのパターン	168 時間間隔で 52 個
間隔クエリの起点	一様分布，Zipf 分布

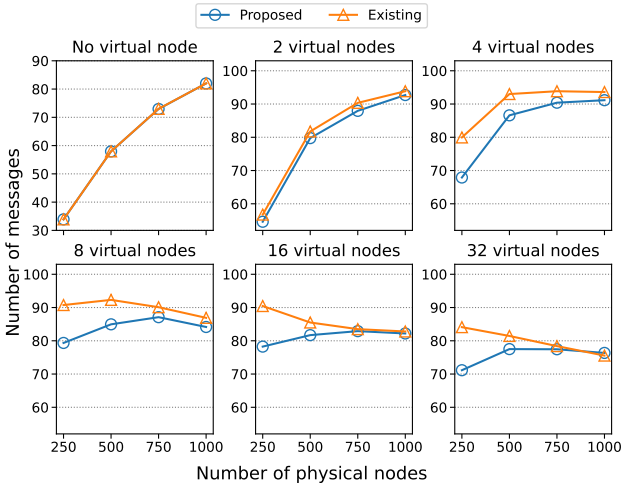


図 6: 一様分布でのメッセージ数

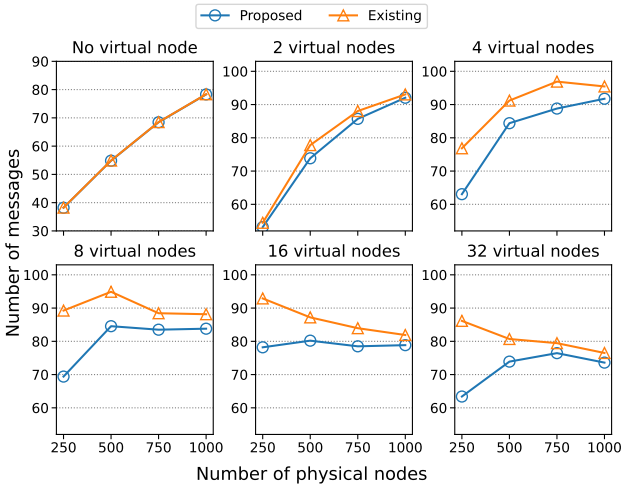


図 7: Zipf 分布でのメッセージ数

を用いた．詳細な実験条件を表 3 に示す．

図 6 および図 7 に結果を示す．一様分布および偏りのある Zipf 分布のいずれでもメッセージ数を削減する効果がはっきりと確認できた．特に同じ仮想ノード数でも物理ノードの数が少ない場合ほど減少は顕著であり，また 4〜8 個の仮想ノードがあるときに最も効果があることがわかった．既存手法と異なり物理ノードの数が少ないときに効果が増したのは，仮想ノード数が同じである分一つの物理ノードの担当範囲が増して提案手法が働きやすくなったからであると考えられる．偏りのある Zipf 分布のときは特に仮想ノード数が多いときの効果が大きくなっており，この手法はデータの人気のばらつきがあっても有効なものであるといえる．一方で，図 6 の仮想ノード数が 32 個のとき，物理ノードの数が 1000 になるとかえって提案手法の方が悪化した．これは，キー空間の大きさが 6.5 万のところに 3.2 万個の仮想ノードを置いた上に第 4 節で述べたように仮想ノードの位置を制限したため，仮想ノードの密度が上がりすぎたためと考えられる．そのため，この手法を効率よく動作させるためには仮想ノードの数を増やしすぎないようにする必要がある．

6. 考察

7. 適応的な仮想ノード割当

提案手法により目的である仮想ノード数を適応的に割り当てることができた． ノード数の丸めの方法を複数種類検討したが，切り捨ては過負荷が起きないがノードの利用率は低く切り上げは過負荷が起きやすいがノードの利用率は高く保つことができた．このように，ノードの負荷と効率にはトレードオフの関係があると考えられる．どの手法を使うかについては，ノードの性能指標の「深刻さ」によって変わると考えられる．すなわち，性能指標が電力効率などを考慮してこの程度の負荷以下に抑えるのが望ましいという参考程度のものであれば利用効率を上げるように，逆にこの負荷を超えると動作に支障をきたすというような絶対的なものであれば負荷を下げるように優先するようなことが考えられる．

また，丸めの手法により仮想ノードの目標数と実際に仮想ノードが割り当てられる数には丸めの方法に 10%ほどの差があることも分かった．しかし，ノード数の目標については厳密に目標値に合わせる必要は低いと考えられる．元となる Chord はノード数 N に対し $O(\log_2 N)$ での検索が可能であり，ノード数が 10%ほど増減しても性能に大きな影響を与えるとは考えられにくい．よって，前述のノードの性能指標の重要さを優先すべきであると考えられる．

追加実験として，適応的な仮想ノード割当を行う状態と行わない状態で第 5.2 節と同じ条件でにメッセージ数を測定したところ，ルーティング手法は変更していないがわずかにメッセージ数を削減する効果があることが分かった．これは，高性能なノードに多くの仮想ノードが集中したことで第 2.4 節で述べた既存研究のルーティング最適化手法が働きやすくなったためだと考えられる．このことからこの手法は副次的にネットワークへの負荷軽減への効果も得られると考えられ，メリットは大きなものだといえる．

7.1 間隔クエリの統合的な処理

提案手法により，物理ノードの数が少ない場合は 10%程度の削減効果が得られた．既存手法の問題点として，メッ

セージ数の削減率が最大でも 5%程度に留まっていたこと [3] と比較すると大きな効率化が行えたといえる．特にノード数が多いときの削減効果の大きさは，仮想ノードの配置を制限する方法によって間隔クエリと他の仮想ノードのヒット率が向上したことが理由の一つであると考えられる．一方で，仮想ノードが多くなりすぎるとかえってメッセージ数が増大する現象も起こった．この手法を有効に働かせるには仮想ノード数が増えすぎないように制限する必要があるということで，本稿で提案した適応的な仮想ノード割当法と組み合わせることが考えられる．

既存手法ではデータを検索するときに先頭のクエリだけを見て処理していたが，提案手法では間隔クエリの全てのクエリと全ての仮想ノードとの総当りが必要となっている．このことが処理負荷を増大させる可能性がある．しかしこの処理は単純な数値の比較であり外部との通信を生じたりするものではないこと，また仮想ノードの数は制限する必要があることが分かったため重大な問題になるとは考えにくい．しかし，間隔クエリが長期間になると無視できなくなる可能性もあるため，クエリの順番などを使った計算量の削減についても検討していきたい．

8. おわりに

8.1 まとめ

近年広まる IoT は，時間や位置に深く結びついたデータを高頻度かつ大量に扱うという特徴を持つ．従来のモデルに代わる分散ネットワークの仕組みとしてこれまで提案してきた複数の時間間隔に基づくネットワークは，P2P 環境におけるノードの性能差を考慮していないものだった．また，ルーティングを最適化することによってネットワークへの負荷軽減を行っていたが，効果は限定的であった．

本稿では，適応的な仮想ノードの割当を提案しノードの性能差を考慮し，間隔クエリの統合的な処理を提案してルーティングのさらなる効率化を行った．メッセージ数の減少に伴い各物理ノードへのアクセス回数も均等に減少し，特に外れ値的にアクセスの多いノードを減らすことができた．低性能のノードは負荷を下げて高性能なノードには利用率を上げる適応的な割り当てを行うことができた．また，既存手法を大きく上回るメッセージ数の削減を行うことも達成できた．

8.2 今後の課題

今後の課題として最も大きいことは，適応的な仮想ノードの割当と間隔クエリの統合的な処理を同時に行うことである．現状ではこの 2 つは独立した関係のない手法となっているが，ノード数の制限や仮想ノードの集中配置といった点でお互いを補えるような性質を持っていることが実験でわかった．実際の環境に近づけつつ負荷の軽減を行うというより大きな目標のために，今後はこれら 2 つの手法を融合させることを目指していく．

適応的な仮想ノードの割当については，分散環境におけるパラメーターの伝達をより具体的に検討することが課題として挙げられる．今回はノードの合計値について一定の誤差を設けて擬似的に再現したが，実際の環境において情報が伝達されていく状況とは異なっている可能性もある．また，アルゴリズムの詳細な動作についても検討していきたい．特に今回の手法ではネットワーク上の仮想ノードの合計値を基に割当を行ったが，ネットワークが生成された直後は仮想ノードがほとんどいないため別の手法を併用する必要があると考えられる．また，丸めの手法を工夫しより負荷軽減と効率を両立できる方法についても今後の展望として望まれる．

謝辞 本研究の一部は JSPS 科研費 22K12009, 放送文化基金,「北陸地域の活性化」に関する研究助成事業による成果である．

参考文献

[1] Zarrin, J., Aguiar, R. L. and Barraca, J. P.: Resource Discovery for Distributed Computing Systems: A Comprehensive Survey, *Journal of Parallel and Distributed Computing*, Vol. 113, pp. 127–166 (2018).

[2] Stoica, I., Morris, R., Karger, D., Kaashoek, M. F. and Balakrishnan, H.: Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications, *SIGCOMM Comput. Commun. Rev.*, Vol. 31, No. 4, pp. 149–160 (2001).

[3] 久保達也，川上朋也：複数の時間間隔に基づくオーバレイネットワークにおけるルーティング効率化手法，情報処理学会論文誌， Vol. 63, No. 2, pp. 526–538 (2022).

[4] 久保達也，川上朋也：複数の異なる時間間隔に基づく構造化オーバレイネットワークでのノード仮想化の検討，2020 年度情報処理学会関西支部 支部大会, G-42 (2020).

[5] 藤樫淳平：オーバレイネットワーク構築のためのネットワークトポロジ提供機構に関する研究，修士論文，奈良先端科学技術大学院大学 (2009).

[6] Schütt, T., Schintke, F. and Reinefeld, A.: Range Queries on Structured Overlay Networks, *Computer Communications*, Vol. 31, No. 2, pp. 280–291 (2008).

[7] 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, pp. 473–492 (2014).

[8] Qian, H. and Andresen, D.: Jade: An efficient energy-aware computation offloading system with heterogeneous network interface bonding for ad-hoc networked mobile devices, *Proceedings of the 15th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, pp. 1–8 (2014).

[9] Rao, A., Lakshminarayanan, K., Surana, S., Karp, R. and Stoica, I.: Load Balancing in Structured P2P Systems, *Peer-to-Peer Systems II*, pp. 68–79 (2003).

[10] 山下高生，栗田弘之，高田直樹：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，情報処理学会論文誌, Vol. 58, No. 12, pp. 1977–1992 (2017).