DEIM Forum 2023 2a-8-3

フォグの負荷及び通信状況を考慮したコストに基づく推論処理の動的負荷分散

小久保柚真† 天笠 俊之††

† 筑波大学大学院 理工情報生命学術院システム情報工学研究群 〒305–8577 茨城県つくば市天王台 1 丁目 1-1
†† 筑波大学 計算科学研究センター 〒305–8577 茨城県つくば市天王台 1 丁目 1-1
E-mail: †kokubo@kde.cs.tsukuba.ac.jp, ††amagasa@cs.tsukuba.ac.jp

あらまし IoT に適した分散処理環境としてフォグコンピューティングが注目を集めている．実世界のユースケースにおいては，IoT デバイスの移動や一時的なデータ量の増加により，特定の処理ノードやネットワークが高負荷になることが考えられる．既存のフォグコンピューティングにおける分散推論手法では，推論処理を静的に配置している，通信状況を考慮していないといった問題があり，想定されるユースケースを実現するには不十分であった．そこで本研究では，推論及び転送に必要とする時間をコストとして見積もり，コストが最小となる推論処理の割り当てを効率的に決定する手法を提案する．実験により，フォグノードの負荷とネットワークの状態を捉えた上で，推論処理を動的かつ高速に分散できることが示された．

キーワード フォグコンピューティング，分散処理，知識グラフ・オントロジ処理，IoT，ストリームデータ処理

1 序 論

近年，通信技術の発展やセンサの性能向上により，交通，農業，医療などをはじめとする幅広い分野で IoT（Internet of Things）の活用が始まっている [1], [2]．多くの IoT システムでは，各 IoT デバイスから送信されるデータを収集・集計・分析するバックエンドとして，クラウドコンピューティング環境が利用されている．しかしながら IoT デバイス数の増加に伴い，通信帯域の圧迫やクラウドへの負荷が集中することによる処理の遅延などが問題になっている [3]．このような理由からクラウドサーバで集中的に処理を行うには限界があった．

そこで IoT に適した分散処理環境として，フォグコンピューティングが提案された [4]．フォグコンピューティングは処理能力を IoT デバイスのある LAN 内に配置し一次処理を行うもので，レイテンシの低減，帯域コストの削減等のメリットがある．これにより上述したクラウドコンピューティングの課題を解決することができる．

一方，エッジ側の IoT デバイスでは，計算能力の向上に伴い，より高度でセマンティックな処理が可能となり，機械学習を用いた推論や分類などの複雑なタスクの実行が可能になってきている．複雑でセマンティックなデータを表現し，異なるデバイス間でデータの相互運用性を実現するために，RDF [5] がデータのセマンティゼーション [6] という文脈で使われることが多くなっている．データを RDF で表現することで，知識ベースを利用してデータを推論し，様々な情報を導き出すことができる．このため，RDF 形式のストリームを対象としたシステムが注目されている．

既存のフォグコンピューティングにおける RDF 推論手法には，推論を行う各フォグノードの負荷を考慮していないという問題があった．この問題に対し，以前の我々の手法 [7] では，CPU 使用率を用いて負荷状況を把握し，負荷が閾値を超える場合には適用する推論ルールを分散することで負荷の軽減を図った．しかし夜間になるとネットワークの利用者が増え，建物全体の通信状況が悪くなるといったことが起きるため，負荷だけを考慮するのは想定されるアプリケーションを実現するうえでは不十分である．また考慮するパラメータが増えることで，分散を行う際の閾値やルールの割り当てを定めるのが困難になる．

IoT タスクスケジューリングを対象としたいくつかの研究 [8], [9] では，消費電力やレイテンシ，タスクのデッドラインなど複数の要素を考慮したタスクの処理ノードへの割り当てを最適化問題として定義し，その問題を効率的に解くというアプローチがとられている．

そこで本研究ではこの考え方を取り入れ，まずクラウド・フォグ間でのルールの割り当てを推論と通信にかかる時間からなるコストの最小化を目的とする最適化問題として定式化する．そしてこの最適化問題の近似解を効率的に得るヒューリスティックアルゴリズムを提案する．

Raspberry Pi 4 を用いた実験により，提案手法はフォグノードの負荷とネットワークの状況を考慮した上で，推論処理を動的かつ高速に分散できることが示された．

本稿の構成は，以下の通りである．まず 2 章で本研究の前提となる知識について説明する．次に 3 章で本研究の関連研究について述べる．4 章ではノードの負荷に加え通信状況を考慮したコストに基づく分散を行う提案手法について説明する．5 章では実デバイスを用いて実施した実験結果を示す．最後に 6 章にて本稿のまとめを述べる．

2 前提知識

2.1 フォグ/エッジコンピューティング

フォグコンピューティングは，Cisco Systems 社が提唱した分散処理アーキテクチャである．IoT デバイスのあるローカルなネットワーク内にデータを一次的に処理するフォグノードを配置し，処理したデータを必要とするデバイスへ送信する．図 1 にフォグコンピューティングの概念図を示す．フォグは多層で構成される場合もあるが，本研究においては図 1 のようにクラウド，フォグノード，IoT デバイスの 3 層からなるシンプルなアーキテクチャを対象とする．

フォグノードで処理することができればクラウドを経由することなく結果を送信できるため通信の遅延が少なく，同時にクラウド・フォグ間の帯域コストを削減することもできる．またデータをローカルで処理できるため，セキュリティレベルの向上やプライバシー保護も期待できる．一方フォグノードはクラウドサーバと比較して処理能力が限られている点や，一時処理を行うために追加のデバイスが必要となるといった点も考慮しなければならない．

なお，フォグコンピューティングと類似したアーキテクチャにエッジコンピューティング [10] がある．両者は同じ意味で使われることもあるが，エッジコンピューティングは IoT デバイスそのものが処理能力を持ち推論等を行うのに対し，フォグコンピューティングは別途処理ノードを配置しクラウドとの間で中間処理を行うという点で使い分けられることが多い．

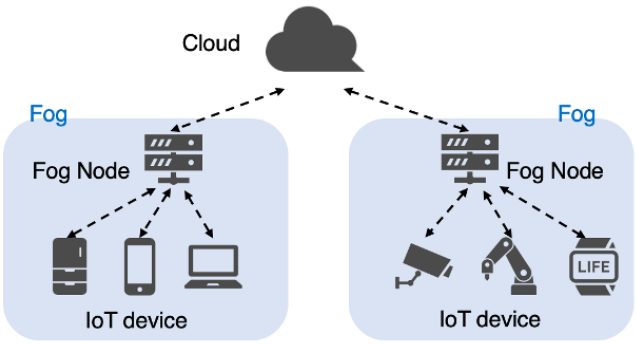

図 1: フォグコンピューティング

2.2 セマンティック技術

2.2.1 RDF

RDF（Resource Description Framework）は，主語 (Subject)，述語 (Predicate)，目的語 (Object) の 3 要素で構成されるトリプルによってリソースの関係を表現する枠組みであり，W3C（World Wide Web Consortium）によって勧告されている [5]．トリプルの各要素に，IRI（International Resource Identifier）という Web 上の特定のリソースを一意に識別する識別子を用いることで，データを統一的に扱うことが可能となる．RDF の表記には RDF/XML や JSON-LD，Notion 3 などさまざまなものがあるが，提案手法では人間にとっても理解しやすく広く利用されている Turtle [11] を利用する．

2.2.2 推論（Semantic Reasoning）

推論とは，明示的に主張された事実や公理の集合から未知の事柄を導出することをいう．推論はオントロジーに基づく推論とルールに基づく推論の 2 つに大別することができる．

前者のオントロジーとは，概念間の関係を体系的にまとめたものであり，代表的なオントロジー記述言語として，RDFS [12] と OWL [13] がある．

後者は一般にルールベース推論と呼ばれ，事前に用意したルールを適用することで新たな事実を導く．本研究はこちらに該当し，各ノードが受信した RDF トリプルに対して，条件を列挙した複数の推論ルールを適用し，条件を満たす場合には推論によって導出された結論を新たなトリプルとして生成する．

一部の推論ルールにおいては，集計関数等を用いたより複雑な条件を記述するために，RDF 形式のデータに対して問い合わせを行う SPARQL（SPARQL Protocol and RDF Query Language）[14] を利用する．SPARQL にはいくつかのクエリタイプがあるが，問い合わせ対象の RDF データからクエリにマッチする値を取得する SELECT と，クエリにマッチする値を用いて新たなトリプルを生成する CONSTRUCT を用いてルールを記述する．

3 関連研究

3.1 フォグコンピューティングにおける分散推論

エッジデバイスの性能向上に伴い，従来クラウドサーバで集中的に行っていた推論処理をフォグノードに分散することで応答性の改善を図る様々な手法が提案されている．

Su らの研究では．クラウドとエッジデバイスに推論を分散させることで，どのようにパフォーマンスを向上させることができるかを分析している [1]．ユースケースとしてスマートシティの交通システムを想定し，タクシーの軌跡データ（位置，速度，方向など）を用いて，右折や渋滞など 16 の行動の推論を行った．また，実験では 4 つの RDF フォーマットごとに転送時間，推論時間の比較も行っている．実験により，エッジデバイスを用いることで全体の処理時間を減らすことができ，特にエッジで処理できる場合には非常に高速であることが分かった．しかしエッジに配置するルールについては静的であり，あらかじめ決められたルールを適用している．

Seydoux らは Su らの研究において，ルールの配置が静的であったことに注目し，フォグコンピューティングにおいて推論ルールを動的に分散するためのアーキテクチャとアプローチの提案を行った [15]．この研究ではフォグが複数層で構成される階層型のトポロジーを対象としており，ルールを適用できる最下ノードで推論を行うことにより，応答時間の高速化とスケーラビリティの向上を図った．しかしながら，推論を行うフォグノードの処理能力とデータ量の動的な変化を考慮していないという課題がある．フォグノードはクラウドサーバと比較して処理能力が乏しいため，処理を分配することでかえって全体の実行時間が長くなってしまうことがある．また各ノードの負荷を

考慮していないため，情報源の移動や特定の時間におけるデータ量の一時的な増加により，特定のノードに負荷が集中し遅延が増加してしまう．

この問題に対し，以前の我々の手法 [7] では，CPU 使用率を用いて負荷状況を把握し，高負荷なノードがある場合にはサーバと処理を分散することで負荷の軽減を図った．実験により，フォグノードが高負荷な場合，負荷を考慮しないナイーブな手法と比較して平均 1.67 倍の高速化を達成した．しかしこの手法ではネットワークの状態を考慮できておらず，また処理の分散を行う閾値を事前に決定しておく必要があった．よって本研究ではこの 2 点の課題の解決に取り組む．

3.2 IoT タスクスケジューリング

Azizi らは異種フォグコンピューティング環境において，タスクを効率的にスケジューリングする 2 つの semi-greedy アルゴリズムを提案している [8]．IoT タスクスケジューリングは一般に応答時間の最小化，消費電力の最小化，またはこの両方を目的とした研究に分類されるが，著者らの手法は 3 つ目に該当し，タスクの期限要件を満たしながらもシステム全体の消費電力を最適化する．またどのノードに割り当てたとしても要件を満たせない場合にはその違反時間が最小となるノードを選択し，違反時間の最小化も考慮するというところに大きな特徴がある．これによりタスクが期限要件を満たす割合やシステムの消費電力だけではなく，期限違反時間の総量という点においても既存アルゴリズムを上回ることが示された．

Yadav らは周囲の車両に搭載された計算資源を動的なフォグノードとして活用する Vehicular Fog Computing におけるタスクオフローディングに取り組んでいる [9]．タスクの割り当てをレイテンシと消費電力から計算されるコストを最小化する最適化問題として定義し，この問題を効率的に解くヒューリスティックアプローチ ECOS の提案を行った．ECOS は，1）過負荷なノードの予測，2）オフロードするタスクの決定，3）オフロード先ノードの決定という 3 ステップによって構成される．高い CPU 使用率と低いメモリ使用率を必要とするタスクを選択し，車両ノードの移動制約を満たす中で最小のコストとなるノードに対しタスクを割り当てる．シミュレーションにより，制約を踏まえた上で消費電力とレイテンシを効果的に削減できることを示した．

これらの研究がフォグノード間での並列的な分散を対象としているのに対し，我々の研究ではフォグとクラウド間での分散を対象としているところが大きな違いである．

4 提 案 手 法

本研究ではフォグノードの負荷とネットワークの状態を考慮した上で，最適な割り当てを動的に決定する手法の提案を行う．まずルールの割り当てを推論と通信に要する時間からなるコストの最小化を目的とする最適化問題として定式化する．そしてこの最適化問題の近似解を効率的に得るアルゴリズムを提案し，実験にて有効性の検証を行う．

4.1 想定するシナリオ

本研究ではスマートシティアプリケーションにおける交通や安全に関する推論をシナリオとして設定する．街中の電灯や信号などに計算資源を配置し，これをフォグノードとして周囲のセンサや付近の人が所有するスマートフォンからの情報を収集し推論を行うシステムを想定する．実験にはスマートシティにおける地域の環境情報に関するデータを使用し，各汚染物質ごとの汚染度の推定や汚染地域の通知を行う．データセットと推論の詳細については， 5.1 節にて述べる．交通や安全に関する推論はリアルタイム性が重視されるため，可能な限り高速に推論結果を得られることが望まれる．また各フォグノードの性能は，遠隔の潤沢な計算資源を持つクラウドサーバと比較して劣るため，この点についても考慮する必要がある．

4.2 アーキテクチャ

本研究で使用するシステムの全体図を図 2 に示す．このアーキテクチャについては，以前の我々の研究にて設計したものである [7]．システムには IoT ノードとフォグノード，そしてクラウドサーバが含まれており，それぞれのノードの構成について図 2 の右側に示している．IoT ノードからフォグノードへの転送にはソケット通信を，フォグノードからクラウドサーバへの転送には MQTT [1]を用いている．MQTT は http と比較して軽量かつ双方向で多対多の通信ができるといった特徴があるため，IoT において一般的に用いられているプロトコルである．MQTT ではアプリケーションの特性に合わせて，転送の品質を表す QoS（Quality of Service）を設定することができるが，本システムにおいては正確に一回だけ転送することを保証する QoS2 を採用した．これは，制御信号を用いてフォグノードの判別や実行時間の計測を行っており，そのメッセージがロストしてしまうのを防ぐためである．

各ノードの構成について，IoT ノードはデータを生成，収集するデータ収集器と RDF 化するための RDF アノテータで構成される．

フォグノードは実際に推論を行うための推論器と推論に用いる知識ベース，そして負荷を分散するための分散機構で構成される．推論器については，RDF を処理するための Java フレームワークである Apache Jena [2]を利用した．

クラウドサーバはフォグノードと同様に推論器，知識ベース，そしてフォグノードとの間で処理を分散するための分散機構があり，加えて MQTT による Publish/Subscribe 通信を行う際にその仲介役となる MQTT Broker が配置されている．MQTT Broker には mosquitto [3]を使用した．

4.3 システムモデル

本研究が対象とする問題を定式化する上で必要となる推論時間，通信時間，そしてこれら 2 つから構成される通知時間を定義する．

1：https://mqtt.org/

2：https://jena.apache.org/

3：https://mosquitto.org/

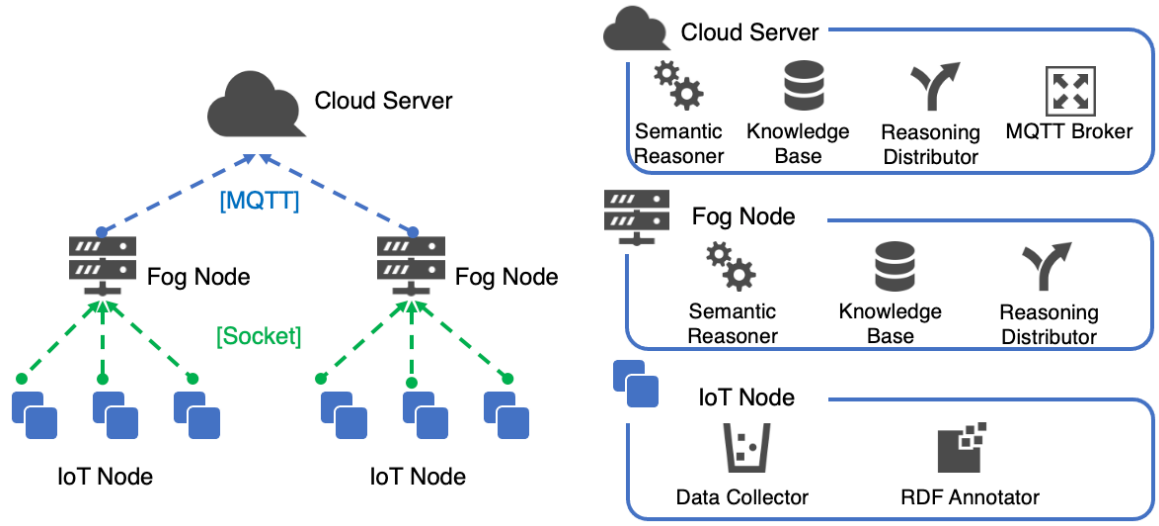

図 2: 設計したシステムの全体図

4.3.1 推論時間

フォグノードにおける総推論時間 R_F およびクラウドサーバにおける総推論時間 R_C は以下のように求められる．

$$R_F = \sum_{j=1}^{n} w_{Fj} r_{Fj} \tag{1}$$

$$R_C = \sum_{j=1}^{n} w_{Cj} r_{Cj} \tag{2}$$

式 1,2 において，r_{ij} はノード i でルール j の推論にかかる時間，w_{ij} はノード i でルール j の推論を行うかどうかを表す 2 値変数であり，0 または 1 をとる．$w_{ij} = 1$ の場合，処理ノード i でルール j をの推論を行い，$w_{ij} = 0$ の場合，処理ノード i ではルール j の推論を行わない．本研究では，フォグノードとクラウドサーバ間での負荷分散を対象としているため，$i = F$ の場合フォグで処理を，$i = C$ の場合クラウドで処理することとする．

ここでフォグノードは処理能力が限定されており，負荷の影響を受けやすいため，負荷による推論時間の増加を考慮する必要がある．本手法では CPU 使用率を用いて負荷状況を把握し，各ルールの推論時間 r_{Fj} に負荷による遅延を加えることで負荷の影響を考慮する．以下の式 3 に負荷による推論時間の増加を反映したフォグノードにおける各ルールの推論時間を示す．

$$r_{Fj} = r_j + \alpha F_{cpu} \tag{3}$$

r_j は CPU 使用率が 0%の時のルール j の推論時間，α が CPU 使用率に基づく推論時間の変動量，F_{cpu} が推論時の CPU 使用率である．各フォグノードが独立してルールの推論時間と負荷変動率を計測することで，フォグデバイスの異種性を考慮することができる．なお，クラウドサーバについてはフォグと比較して潤沢な計算資源があるという前提に基づいているため，負荷を考慮していない．

4.3.2 通信時間

通信時間は伝搬遅延と伝送遅延の和として定義する．ここで伝搬遅延とはノード間の物理的距離によって発生する遅延であり，クラウドサーバが海外等の遠隔地に設置されている場合には大きくなりやすい．一方，伝送遅延はパケットが転送される際の遅延であり，パケットサイズが大きい場合やネットワークの帯域幅が十分でない場合に大きくなる．ノード x, y 間の伝送遅延は，式 4 に示すようにノード x, y 間の転送量 D_{xy} を帯域幅 B_{xy} で割った値として計算される．よってノード x, y 間の通信時間は伝搬遅延 t^p_{xy} と伝送遅延 t^d_{xy} の和として式 5 のように求められる．

$$t^p_{xy} = \frac{D_{xy}}{B_{xy}} \tag{4}$$

$$t_{xy} = t^p_{xy} + t^d_{xy} \tag{5}$$

以降 IoT ノードとフォグノード間の通信時間を t_{IF}，フォグノードとクラウドサーバ間の通信時間を t_{FC} と表記する．

4.3.3 通知時間

通知時間はデータが IoT ノードより転送されてからフォグノードまたはクラウドサーバで推論を行い，その結果を IoT ノードと同じフォグに属するアプリで受け取るまでの時間とする．フォグで処理したルールの通知時間 T_F は図 3 の紫色＋青色の経路，クラウドで処理したルールの通知時間 T_C は紫色＋オレンジ色の経路を辿った時間になる．よって T_F, T_C は前述の推論時間と通信時間の和として，以下の式 6,7 で求められる．

$$T_F = t_{IF} + R_F + t_{IF} = 2t_{IF} + R_F \tag{6}$$

$$T_C = t_{IF} + t_{FC} + R_C + t_{FC} = t_{IF} + 2t_{FC} + R_C \tag{7}$$

4.4 問題の定式化

本研究の目的は，フォグノードの負荷とネットワークの状態を考慮したルールの割り当てを動的に決定し，全ルールの平均通知時間を短縮することである．前節で定義した通知時間をコストとして，各ルールごとにフォグノード，クラウドサーバのどちらで推論を行うかを決定する．そして全ルールの合計コストを計算し，その値が最も小さくなる処理ノードの割り当てを

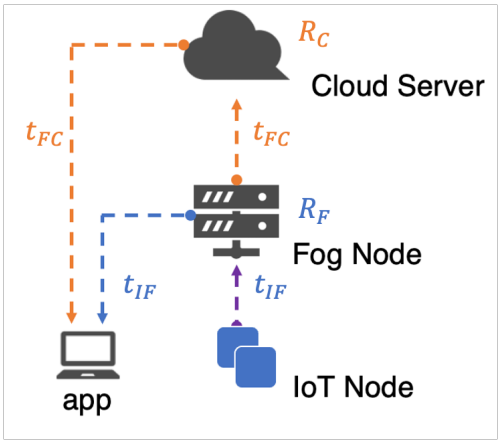

図 3: 通知時間の構成

採用する．コストは予想される通知時間からより算出されるため，コスト最小となる割り当てを採用することで，合計通知時間および平均通知時間が最小となることが期待される．

よって本研究で目的とするルールの割り当ては，合計コストが最も小さくなるルールの割り当て χ を求める問題として式 8 のように定式化される．

$$\min_{\chi} \sum_{i=\{F,C\}} \sum_{j=1}^{n} w_{ij} x_{ij} \tag{8}$$

$$s.t.\ w_{ij} = \{0,1\}, \quad \sum_{i=\{F,C\}} w_{ij} = 1,$$

$$x_{ij} = \begin{cases} T_F = 2t_{IF} + R_F & (i = F) \\ T_C = t_{IF} + 2t_{FC} + R_C & (i = C) \end{cases}$$

$$R_F = \sum_{j=1}^{n} w_{Fj} r_{Fj}, \qquad R_C = \sum_{j=1}^{n} w_{Cj} r_{Cj}$$

w_{ij} はルールをクラウドサーバかフォグノードのどちらに割り当てるかを示す 2 値変数である．$w_{ij} = 1$ の場合，処理ノード i でルール j をの推論を行い，$w_{ij} = 0$ の場合，処理ノード i ではルール j の推論を行わないことを表す．各ルールはどちらか一方でのみ処理されるため，各ルール j について $w_{ij} = 1$ となるのは必ず 1 つになる．x_{ij} は 4.3 節にて定義した通知時間であり，これをコストとして用いる．

注意すべき点として，R_F，R_C は，処理ノード i で推論するルール数が多くなるほど増加する．各ルールごとの推論結果を個別に転送すると通信が高頻度になってしまう．加えて本アーキテクチャにおいて，推論はあらかじめ使用するルールによって推論モデルを構築しまとめて推論を行うため，全ルールでの推論を終えた後まとめて結果を転送する．そのため一方の処理ノードで推論するルールが増えると，そのノードで処理する他のルールの通知時間も遅くなる．これにより他のルールの割り当てについても考慮しなければならない．また整数制約 w_{ij} により，この最適化問題は NP 困難な整数計画問題となるため，効率的な解法が必要となる．

4.5 コストに基づく割り当てアルゴリズム

N 個のルールをフォグかクラウドのどちらかに割り当てる組み合わせは 2^N 通りであり，ルール数の増加に伴ってコストの計算時間が大きく増加する．そこで 4.4 節の最適化問題の解を効率的に得るヒューリスティックアプローチを提案する．

アプローチの基本的な方針としては推論に時間のかかるルールを優先してクラウドに移行するというものである．時間を要するルールほどクラウドで処理することによるコストの低下が大きくなるため優先的にクラウドで処理する．このようにクラウドへの割り当てを各ルールの推論時間の降順に制限することで，割り当ての組み合わせを $N + 1$ 通りに削減できる．

上記のアイデアを取り入れた割り当て決定アルゴリズムを Algorithm1 に示す．このアルゴリズムでは，入力として フォグの負荷変動量α, 使用するルール数 N, あらかじめ昇順に整列されたフォグ・クラウドのルール毎の推論時間 $r_1, ..., r_N, r_{C1}, ..., r_{CN}$, フォグ・クラウドで処理したルールの通知時間 t_{IF}, t_{FC} を受け取り，$N+1$ 通りの中でコスト最小となる割り当て I を出力する．

まず実行時のフォグノードの CPU 使用率を計測し，その値をもとに推論時間に加える遅延を計算する（1-2 行目）．続いてクラウドに割り当てるルールの数を 1 つずつ減らしながら，N+1 通りの割り当てごとコストを計算する（3-14 行目）．最後に N+1 通りの中で最小値を検索し，最小値を持つインデックスを最良の割り当てとして返す（15-22 行目）．このアルゴリズムは図 2 におけるフォグノードの分散機構で実行される．

フォグノードではシステム起動時に，各ルールの推論時間と負荷による推論時間の変動量，フォグ・クラウドで処理したルールの通信時間を計測し，ルールごとの推論時間については昇順に整列しておく．またクラウドでの推論時間もこのタイミングで通知される．システムを長時間稼働する際には一定間隔ごと計測・通知することで通信状況の変化やルールの追加も考慮することができる．

5 実　　験

5.1 実験環境及び各種パラメータ

実験で使用した各デバイスの性能について，以下の表 1, 2, 3 に示す．IoT ノードは MacBook Pro，フォグノードは Raspberry Pi 4，クラウドサーバは Amazon Elastic Compute Cloud (EC2) の m5.2xlarge インスタンス上に実装した．なお EC2 については東京リージョンを利用している．

実験で使用したパラメータについて，以下の表 4 に示す．フォグノードは最大 5 台で，それぞれ最大 5 台の IoT ノードが接続されるため，使用する IoT ノードの最大数は 25 である．いずれの実験についても 10 回実施し，最大値と最小値を除いた 8 回分の結果の平均値を掲載している．

また遅延制御には tc コマンドを，伝搬遅延の計測には ping コマンドを，伝送遅延における帯域幅の計測には iPerf コマンドを用いた．各試行ごとにばらつきはあるものの，実行環境における平均的な IoT ノード・フォグノード間の伝搬遅延は 7ms, フォグノード・クラウドサーバ間の伝搬遅延は 17ms で，帯域幅についてはそれぞれ 18Mbps, 34Mbps 程度であった．

Algorithm 1 コストに基づくルール割り当ての決定

Input: フォグの負荷変動量α, ルール数 N, フォグ・クラウドのルール毎の推論時間（昇順に整列済み）$r_1, ..., r_N, r_{C1}, ..., r_{CN}$, フォグ・クラウドで処理したルールの通知時間 t_{IF}, t_{FC}

Output: 割り当て I //I 番目までのルールをフォグで処理する

$F_{cpu} \leftarrow getCpuUsage()$ //CPU 使用率を取得
$Load \leftarrow F_{cpu} * \alpha$
for $i \leftarrow 0$ to $N+1$ **do**
  $R_F, R_C, cnt \leftarrow 0$
  for $j \leftarrow 0$ to N **do**
    if $(j < i)$ **then**
      $R_F \leftarrow R_F + r_j + Load$
      $cnt \leftarrow cnt + 1$
    else
      $R_C \leftarrow R_C + r_{2j}$
    end if
  end for
  $result[i] \leftarrow (R_F + T_F) * cnt + (R_C + T_C) * (N - cnt)$
end for
$minReslut \leftarrow result[0]$
$I \leftarrow 0$
for $k \leftarrow 1$ to $N+1$ **do**
  if $minResult > result[k]$ **then**
    $minResult \leftarrow result[k]$
    $I \leftarrow k$
  end if
end for
return$I$

表 1: IoT ノード

OS	macOS 11.6
CPU	2.4GHz quad-core Intel Core i5
Memory	16GB

表 2: フォグノード

OS	Raspbian GNU/Linux 11 (bullseye)
CPU	1.5GHz quad-core ARM Cortex-A72
Memory	8GB

表 3: クラウドサーバ

OS	Ubuntu 20.04.4 LTS
CPU	Intel Xeon Platinum 8259CL CPU @ 2.50GHz
Memory	32GB

5.2 データセット及び設計したルール

データセットには，City Pulse プロジェクト [16] で提供されているデータを利用した．このプロジェクトでは，スマートシティに関する交通，天気，イベントなどのデータやベンチマークなどの各種ソフトウェアを提供している．

その中でも大気汚染に関する人工データを使用した．データには，データを取得したセンサの緯度経度，取得日時，対象とする物質，AQI (Air Quality Index) が含まれている．AQI は大気汚染の程度を示す指標であり，この値が 0 から 33 の間であれば大気の状態は Very Good，34 から 66 の間であれば Good のように分類される．

表 4: 実験に使用したパラメータ

フォグノードの数	1, 3, 5
IoT ノードの数	1, 3, 5 / フォグノード
フォグノードへの一回あたりの転送量	60 triples
フォグノードへの転送間隔	100[ms]
フォグノードの推論間隔	5[s]
フォグノードに加えた CPU 負荷	0, 50, 100[%]
フォグ・クラウド間に加えた遅延	0, 100, 200[ms]

提供されているデータには，あらかじめ RDF フォーマットの一種である Turtle でアノテーションが付与されている．そのため実験では IoT ノードからこのデータをそのまま転送する．使用した RDF データ数は 1 IoT ノードあたり 3000 トリプル，最大で 75000 トリプルである．

実験のために設計した 8 つのルールについて，以下の表 5 に示す．各ルールごとの推論時間は，事前の計測に基づいて決定した．フォグノードとクラウドサーバの処理時間には約 10 倍の差があったため，クラウドサーバにおける各ルールの推論時間については一律フォグノードの 1/10 とした．

また，R1-6 については Apache Jena の Rule Engine を，R7,8 については Jena の SPARQL Processor（ARQ）を用いて推論を行った．

表 5: 各ルールの詳細

ルール	説明
R1	AQI の分類 (Very Good)
R2	AQI の分類 (Good)
R3	AQI の分類 (Fair)
R4	AQI の分類 (Poor)
R5	AQI の分類 (Very Poor)
R6	AQI の分類 (Hazardous)
R7	最大の AQI とその物質の特定
R8	汚染地域の通知

5.3 結果の通知にかかる時間の比較

提案手法の有効性を検証するために，様々な条件下で 3.1 節で説明した以前の我々の手法と通知時間の比較を行った．この時の通知時間の比較結果を図 4 に，コストに基づくルールの割り当て結果を表 6 に示す．この実験では，フォグノードの数は 1 つに固定し，IoT ノードの数，フォグノードにかける負荷，そしてフォグノード・クラウドサーバ間に加える遅延を変化させた．図 4 の各グラフにおいて横軸は IoT ノードの数，縦軸は全ルールの平均通知時間である．また，青色の Proposed1 がフォグノードの負荷のみを考慮する以前の我々の手法，オレンジ色の Proposed2 が本研究で提案するフォグノードの負荷とネットワークの状態を考慮した手法の結果である．また表 6 において，ルールの割り当てに F6C2 のような表記を使用してい

るが，この場合フォグで 6 つのルールを，クラウドで残り 2 つのルールを用いて推論することを表している．

まず負荷，遅延の条件によらず Proposed1，Proposed2 ともに IoT ノード数が増えると通知時間が線形に増加することがわかる．フォグノードに接続する IoT ノードが増えることで，一回あたりの推論量及び転送量が増えるため，これは自然な結果である．

遅延が 0ms の場合，負荷が大きくなるにつれ Proposed1 の通知時間は大きく減少している．Proposed1 はフォグノードの負荷のみをもとにルールの割り当てを行っているため，遅延によらず負荷が 0%の場合は全てのルールをフォグで，50%の時は R1-6 の 6 つのルールをフォグで処理しており，100%の場合は全てクラウドで処理している．今回の実験環境においては，IoT ノード・フォグノード間の通信時間とフォグノード・クラウドサーバ間の通信時間の差が小さいため，遅延を加えない場合には，高い処理能力を有するクラウドサーバでの処理を優先した方がよい．このことが結果からも読み取れる．一方 Proposed2 の通知時間はどの負荷条件においても Proposed1 の 100%の通知時間と同程度であり，表 6 からも常に全ての推論をクラウドサーバで行っていることがわかる．よって Proposed2 は実験環境の特性をコストを用いて判断し適切な割り当てを決定できていることがわかる．

遅延が 100ms の場合は，2 つの手法に大きな差が見られない．これはフォグノードにおける推論時間とクラウドサーバに移行することによる通信時間の増加が拮抗しているためであり，Proposed2 におけるルール割り当ては Proposed1 とほぼ同様になっている．

遅延が 200ms の場合，負荷 100%の時は Proposed2 の方が良いが 0%の時は Proposed1 の方が僅かに良いという結果になった．負荷 100%の場合には，Proposed1 は遅延の影響を無視して全ての推論をクラウドサーバに移行するため，通知時間が大きく増加する．一方負荷が 0%の場合は Proposed1 は全ての推論をフォグノードで行う．通信遅延が大きい場合には全ての推論をフォグノードで処理する方が望ましいが，Proposed2 は表 6 から一部のルールがクラウドに割り当てられていることがわかる．この原因として，Proposed2 は処理時間の短いルールから順にフォグに割り当てるため，最後に残る処理コストの大きいルールをフォグに移行しづらい，また実験に使用したフォグノードとクラウドサーバの処理時間に約 10 倍の差があるため，クラウドでの処理を選択しやすいという 2 点が考えられる．

一般的に遅延が大きくなればフォグでの推論量を増やし，負荷が大きくなればクラウドでの推論量を増やすことが望ましいと考えられるが，この傾向は表 6 からも確認することができる．また，Proposed2 は事前に CPU 使用率等の閾値を定めなくとも Proposed1 と同程度以上の結果が得られることがわかった．よって本研究の提案手法はフォグノードの負荷やネットワークの状況を捉え，推論処理を動的に分散できることが示された．

5.4 フォグノード数を変化させた時の通知時間への影響

続いて IoT ノードの数を 5 つに固定し，フォグノードの数を変化させた時の通知時間への影響を調査した．結果を図 5 に示す．

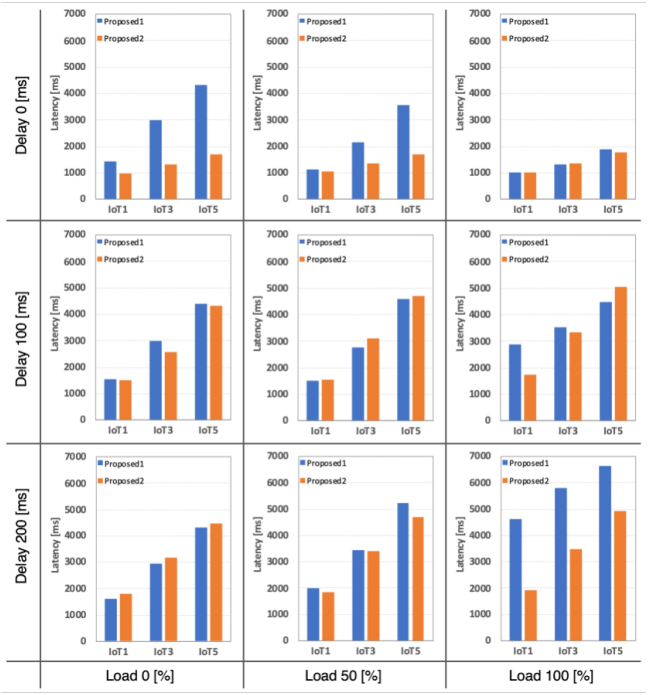

図 4: 様々な条件下における提案手法 1，2 の通知時間の比較

表 6: ルール割り当て結果

		CPU 負荷 [%]		
		0	50	100
遅延 0 [ms]	IoT 1	F0C8	F0C8	F0C8
	IoT 3	F0C8	F0C8	F0C8
	IoT 5	F0C8	F0C8	F0C8
遅延 100 [ms]	IoT 1	F6C2	F6C2	F6C2
	IoT 3	F6C2	F5C3	F5C3
	IoT 5	F6C2	F5C3	F4C4
遅延 200 [ms]	IoT 1	F7C1	F7C1	F7C1
	IoT 3	F6C2	F6C2	F6C2
	IoT 5	F6C2	F6C2	F6C2

結果から全体としてフォグノード数が増加することで，通知時間がわずかに増加することが確認できる．これはクラウドサーバが各フォグノードからの推論処理に対応する必要があることに起因する．

遅延が 0ms の場合は全ての推論をクラウドサーバで行うため，負荷による影響を受けていない．また遅延が 100ms の場合と 200ms の場合でほとんど通知時間が変化しないことから，遅延による通信時間の増加をコストに反映し，適切な分散が行われていると考えられる．

5.5 通知時間の分析

最後に割り当て決定に要する時間が通知時間全体に及ぼす影響について調査するために通知時間の分析を行った．通知時間の内訳を図 6 に示す．下から順に IoT ノードからフォグノードへの転送時間，割り当て決定にかかる時間，フォグノードでの

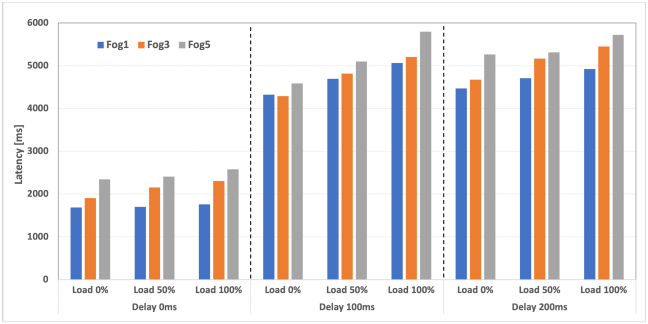

図 5: フォグノード数を変化させた時の通知時間の変化

推論時間，フォグノードからクラウドサーバへの転送時間，クラウドサーバでの推論時間を表している．使用した IoT ノード数，フォグノード数はともに 1 である．

結果から全体に占める割り当てにかかる時間の割合は非常に小さいことがわかる．よって提案手法は通知時間に影響のない時間で割り当てを決定できることが示された．

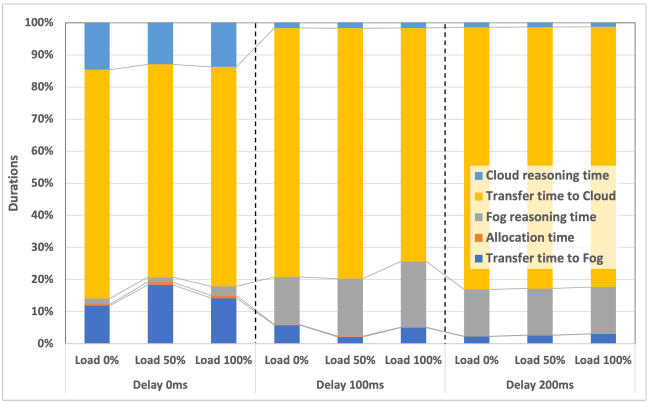

図 6: 通知時間の内訳

6 結　　論

本研究では，フォグ・クラウド間のルールの割り当てを推論と通信に要する時間からなるコストの最小化を目的とする最適化問題として定式化したうえで，最適化問題の近似解を効率的に得るアルゴリズムを提案した．Raspberry Pi 4 を用いた実験により，提案手法はフォグノードの負荷やネットワークの状況を捉え，推論処理を動的かつ高速に分散できることが示された．

今後の課題としては，依存性のあるルールの割り当てやユーザのニーズを反映した割り当てを効率的に決定する手法の検討が挙げられる．本研究で扱った推論ルールは適用する順序に制限がなく，どのような割り当てが選択されても影響がなかったが，適用する順序に制約があるルールやクラウドサーバでしか処理することができない高負荷なルールも考えられるため，この問題への対応を検討している．

文　　献

[1] Xiang Su, Pingjiang Li, Jukka Riekki, Xiaoli Liu, Jussi Kiljander, Juha-Pekka Soininen, Christian Prehofer, Huber Flores, and Yuhong Li. Distribution of semantic reasoning on the edge of internet of things. In *2018 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pp. 1–9, 2018.

[2] Tse-Chuan Hsu, Hongji Yang, Yeh-Ching Chung, and Ching-Hsien Hsu. A Creative IoT agriculture platform for cloud fog computing. *Sustainable Computing: Informatics and Systems*, Vol. 28, p. 100285, 2020.

[3] Mohammad Manzurul Islam, Sarwar Morshed, and Parijat Goswami. Cloud computing: A survey on its limitations and potential solutions. *International Journal of Computer Science Issues*, Vol. 10, pp. 159–163, 07 2013.

[4] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, p. 13–16, New York, NY, USA, 2012. Association for Computing Machinery.

[5] W3C. Rdf 1.1 concepts and abstract syntax. `https://www.w3.org/TR/rdf11-concepts/`, 2013. (accessed Oct. 12, 2022).

[6] Feifei Shi, Qingjuan Li, Tao Zhu, and Huansheng Ning. A survey of data semantization in internet of things. *Sensors*, Vol. 18, p. 313, 01 2018.

[7] 小久保柚真, 天笠俊之. フォグコンピューティングにおける RDF 推論処理の動的な負荷分散. 第 14 回データ工学と情報マネジメントに関するフォーラム (DEIM 2022), K21-2, 2022.

[8] Sadoon Azizi, Mohammad Shojafar, Jemal Abawajy, and Rajkumar Buyya. Deadline-aware and energy-efficient IoT task scheduling in fog computing systems: A semi-greedy approach. *Journal of Network and Computer Applications*, Vol. 201, p. 103333, 2022.

[9] Rahul Yadav, Weizhe Zhang, Omprakash Kaiwartya, Houbing Song, and Shui Yu. Energy-latency tradeoff for dynamic computation offloading in vehicular fog computing. *IEEE Transactions on Vehicular Technology*, Vol. 69, No. 12, pp. 14198–14211, 2020.

[10] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, Vol. 3, No. 5, pp. 637–646, 2016.

[11] W3C. Rdf 1.1 turtle, terse rdf triple language. `https://www.w3.org/TR/2014/REC-turtle-20140225/`, 2014. (accessed Oct. 12, 2022).

[12] W3C. Rdf schema 1.1. `https://www.w3.org/TR/2014/REC-rdf-schema-20140225/`, 2014. (accessed Oct. 12, 2022).

[13] W3C. Owl 2 web ontology language document overview (second edition). `https://www.w3.org/TR/owl2-overview/`, 2012. (accessed Oct. 12, 2022).

[14] W3C. Sparql 1.1 query language. `https://www.w3.org/TR/sparql11-query/`, 2013. (accessed Oct. 12, 2022).

[15] Nicolas Seydoux, Khalil Drira, Nathalie Jane Hernandez, and Thierry Monteil. EDR: A Generic Approach for the Distribution of Rule-Based Reasoning in a Cloud-Fog continuum. *Semantic Web – Interoperability, Usability, Applicability*, Vol. 11, No. 4, pp. 623–654, August 2020.

[16] Muhammad Intizar Ali, Feng Gao, and Alessandra Mileo. Citybench: A configurable benchmark to evaluate rsp engines using smart city datasets. In *In proceedings of ISWC 2015 - 14th International Semantic Web Conference*, pp. 374–389, Bethlehem, PA, USA, 2015. W3C.