IoTデータの処理・利活用を促進するダイナミックリソースコントローラー技術

Dynamic Resource Controller Technology to Accelerate Processing and Utilization of IoT Data

● 久保田　真　● 福田茂紀　● 野村佳秀　● 阿比留健一

あらまし

近年，IoT(Internet of Things)のビジネス活用の本格化に向けて，エッジコンピューティングと呼ばれる考え方が注目されており，各社で取組みが始まっている。これは，全てのデータ処理をクラウド上で行うのではなく，現場近くのノードに分散させることでIoTによる通信トラフィックの爆発的な増加を抑制し，ネットワークのレスポンス向上を図るものである。富士通研究所ではこの概念を拡張し，現場の端末近くに設置したコンピュータ(エッジノード)をクラウドに融合させ，現場の状況変化に応じてデータ処理や蓄積の場所をエッジノードとクラウドの間でダイナミックに最適化するダイナミックリソースコントローラー(DRC)技術を開発した。これにより，既存のエッジコンピューティングと異なり，ネットワークリソースの使用量が削減できる。またそれだけでなく，余剰のネットワーク帯域があれば積極的にネットワークリソースの使用量を増やして新たな発見につながる未加工の生データをクラウドに集めたり，デバイス増加やエッジノード負荷変動の際に，一部処理の実行場所をリソースに余裕のあるエッジノードに変えたりすることで，システムの安定性維持が可能になる。

本稿では，デバイス増加やエッジノード負荷の変化時に適切な処理実行場所を高速に決定するDRC技術の仕組みと，その評価結果を紹介する。

Abstract

Recently, a concept called edge computing has been attracting attention for full-scale application of the Internet of Things (IoT) to business and companies have started working on its use. Edge computing is intended to improve a computing system's response to terminals by distributing data processing between nodes provided near the site, rather than having all data processed in a cloud. It is hoped this will restrain the huge increase in the volume of communication traffic produced by the IoT. Fujitsu Laboratories has extended this concept and developed dynamic resource controller (DRC) technology, which integrates computers (edge nodes) installed near the terminal on the site into the cloud. This technology dynamically optimizes the locations of data processing and storage between the edge nodes and cloud according to changes in the site environment. Unlike the existing edge computing, this not only reduces the usage of network resources but also allows for a positive increase in the usage of any surplus network bandwidth to gather raw data to the cloud that may lead to new discoveries. It can also be used to deal with addition of devices and edge node load variation by changing the location of some of the processing to edge nodes with surplus resources, which helps maintain the stability of the system. This paper describes the mechanism of the DRC technology, which allows a computing system to make a quick decision on the appropriate location of processing execution to deal with addition of devices and variation of the edge node load, and presents the results of its evaluation.

ま　え　が　き

近年，ICTおよびクラウドサービスが，ヒトの活動を支えるインフラとして普及し定着してきた。加えて，ネットワーク接続機能を持つセンサーデバイスの低価格化が進んでいる。今後，実世界（現場）にある車や照明，各種家電製品など多種多様なモノがネットワークにつながり，現場の状況を把握・分析してヒトの活動を支援する，いわゆるIoT（Internet of Things）の普及により，実世界とサイバー世界の融合が進んでいくと考えられる。

本稿では，このようにICTの利活用形態が変化していく中で，IoTにより得られたデータを処理し利活用するシステムの性能の最大化・安定化に向け，筆者らが取り組んでいるダイナミックリソースコントローラー（DRC）技術について，その意義や先進性を紹介する。

ICTの利活用形態の変化

これまでのICTの利活用形態は，必要なときにヒトがパソコンやスマートフォンからクラウド上のデータにアクセスする形態が一般的であった。しかし，あらゆるモノがネットワークにつながるIoTでは，実世界に置かれた大量のモノが，現場状況を示す生データをネットワークを介してクラウドなどに定常的にアップロードするようになる。クラウドでは，収集したデータを蓄積し，加工・集計・分析を行う。IoTの真の価値はこのデータの分析結果の活用にあり，新たな発見につながる情報としてヒトに届けたり，分析結果に従って現場のモノを制御したりすることで，ヒトの日々の活動やビジネスを改善できると期待されている。

ネットワークにつながるモノの数は，2020年には530億個にもなると予測されている。[1] そのため，管理コストの増大，モノを起点とした膨大かつ新たな形態の通信トラフィックによるネットワークリソースの枯渇・輻輳（ふくそう）が問題となる。また，輻輳に伴うネットワークのレスポンス低下，現場の機器を制御する際のリアルタイム性の不足，個人データを含む現場状況を全てクラウドに蓄積することによるプライバシー漏えいのリスクもある。

エッジコンピューティングの課題

前章で述べた問題を解決するために，エッジコンピューティングと呼ばれる広域分散処理が考え出された。エッジコンピューティングは，これまでデータセンター内に閉じて行われていたデータ蓄積やアプリケーション処理を，エッジノードと呼ばれる現場の端末近くに設置したコンピュータ上で行う。すなわち，現場のモノから発生したデータを近傍のエッジノードで処理し，クラウドには処理結果だけ通知する。これにより，通信トラフィックを削減し，ネットワークリソースの枯渇やそれに伴うレスポンス低下問題を解消する考え方であり，各社で取組みが始まっている。[2]

しかし，既存の取組みでは，データ処理の場所が固定されていたため，以下のような課題があった。

(1) モノの増加に伴うエッジノードのリソース溢（あふ）れ

ここでは，工場のフロアにエッジノードを設置し，エッジノードにバイタル解析処理，カメラ映像解析処理，製造品の集計処理を行わせ，クラウドには異常と検査完了数だけ通知させる工場監視ソリューションを想定する。例えば，監視エリア拡大のためにカメラの数を増やしたり，あるラインで作業員を一時的に増員したりした場合，収集するデータ量の増加により解析処理負荷がエッジノードの許容負荷を超えると，エッジノードがダウンしてシステムが停止するリスクがある。そしてこのシステムダウンは，システム管理者がシステム再設計を行うか，ノードリソースを増強するまで解消されない。

(2) クラウドへの生データの収集不足

上記システムでは，クラウドで把握できるのは作業員の明らかなバイタル異常，そのときの立ち位置，検査の進捗状況までである。つまり，エッジノードで生データを捨てているため，通信トラフィックは削減できているが，生データはクラウドには収集されない。もしも異常しきい値未満の値を含むバイタル情報や作業映像情報などの生データがあれば，詳細分析により，例えば作業員のあくび頻度と検査エラー確立の間に因果関係があると分かり，新たな歩留り改善策を実施できるなどの発見につながる可能性がある。

（3）現場環境の変化に合わせたシステム再設計コスト

現場のモノから収集したデータをどのエッジノードやクラウドで解析・集計させるかは，SEがシステム内のエッジノードとクラウドの負荷，またネットワークにかかる負荷とインフラ増強にかけられるコストのバランスを考えて設計する。しかし，（1）のような現場の状況変化が頻発すると，その都度システム設計をやり直すのはSEにとって大きな負担になる。特にモノの数やエッジノードの数が多い大規模システムにおいては，1例で1か月単位の工数を要する場合があり，一時的に現場作業員を増員するような細かな変化にSEが対応するのは現実的ではない。

DRC技術

本章では，富士通研究所で研究開発を進めているDRC技術について述べる。

DRCは，エッジコンピューティングの概念を拡張したものであり，単にデータ処理や蓄積の場所をエッジノードに移すだけでなく，エッジノードをクラウドに融合させるものである。つまり，現場状況の変化に応じて，エッジノードとクラウド間でデータ処理や蓄積場所をダイナミックに最適化することを特長としたIoT対応クラウド技術である。端末・機器の近傍や広域ネットワークの端に配したエッジノードを，広域にまたがる拡張クラウド環境とみなす（**図-1**）。

これにより，単にモノを起点とした膨大な通信トラフィックの問題を分散処理により解消するだけでなく，現場のモノの増減・移動に伴うエッジノードやネットワークの負荷変動の発生時に，データ処理や蓄積の実行場所を変えてシステムの安定性を維持することが可能になる｛**図-2**（1），（1）′｝。更に，ネットワークリソースの余剰帯域を見つけた場合に，一部現場のデータ処理の実行場所をエッジノードからクラウドに移すことによって，新たな発見につながる生データをクラウドに集めることができる｛図-2（2），（2）′｝。このように，エッジノードやネットワークの負荷変動，余剰リソースの発生時にDRCが処理場所の最適化を自動的に行うため，SEはシステム再設計の作業から解放される｛図-2（3），（3）′｝。

なお，本技術の応用として個人が必要とするコンテンツデータを，各個人の位置情報やネットワーク余剰リソースを活用してエッジノードに事前配備する方法もある。これにより，個人がコンテンツにアクセスする際の体感品質（レスポンス性能）を高めることも可能である。[3]

DRCの基本アーキテクチャーを**図-3**に示す。ア

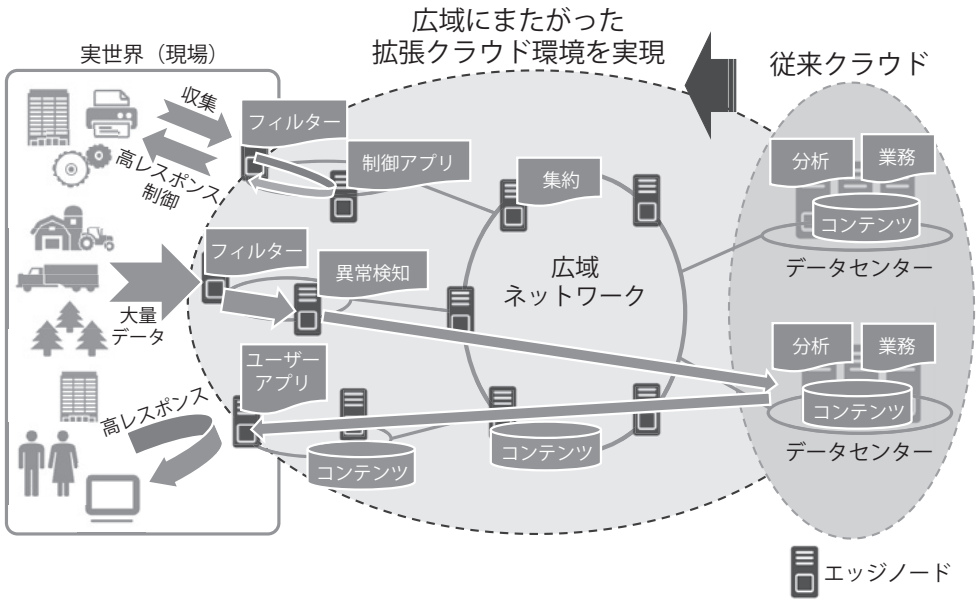


図-1　DRCの概念

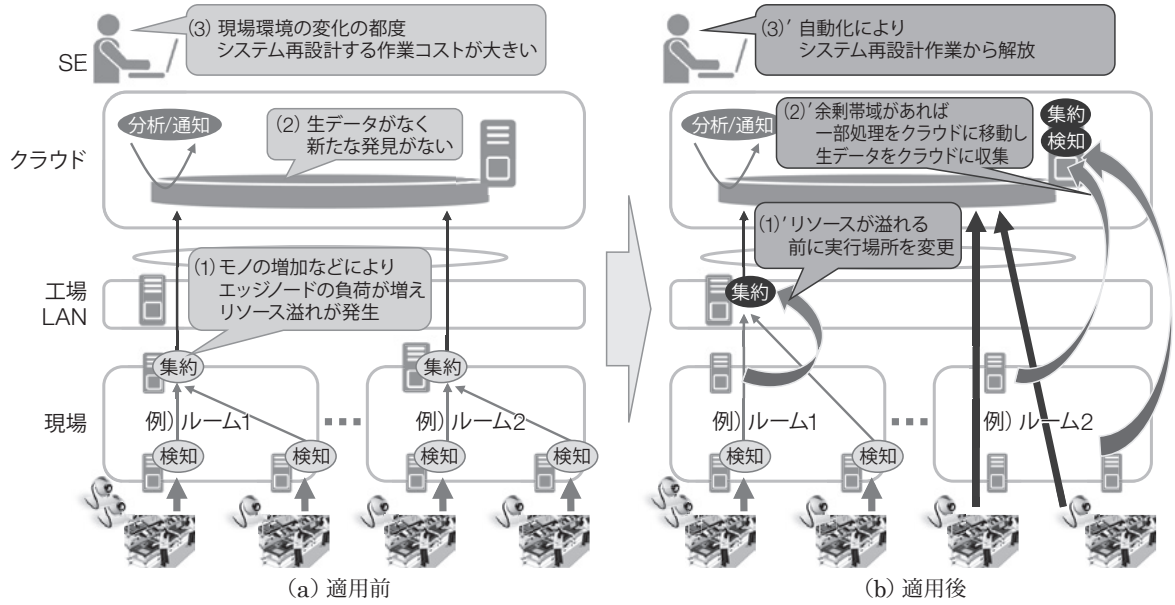


図-2　DRCの適用例

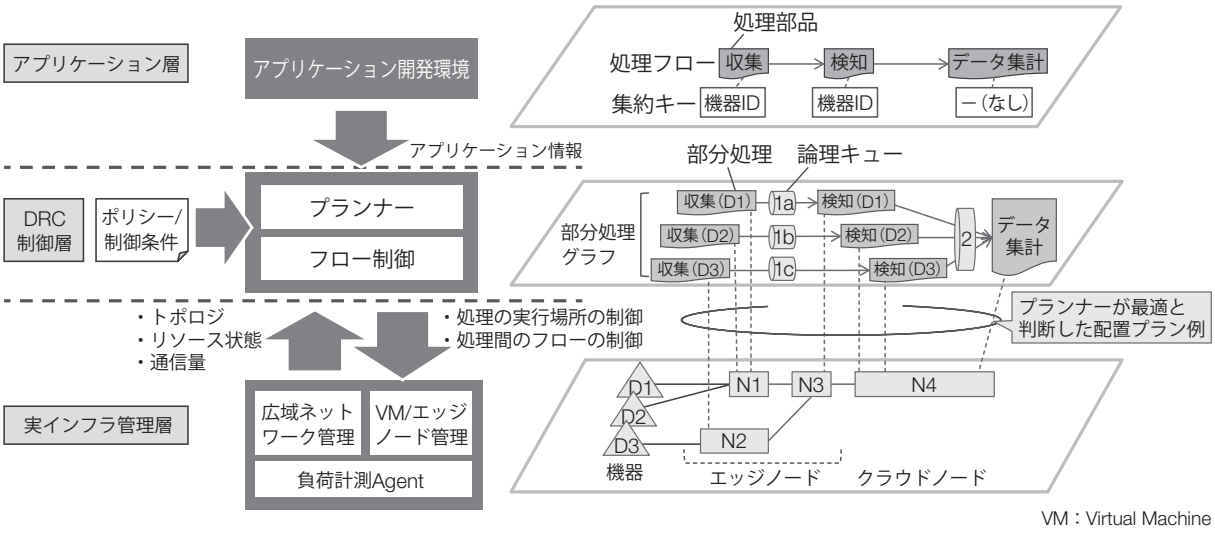


図-3　DRCの基本アーキテクチャー

プリケーション層，実インフラ管理層，それらの中間に位置するDRC制御層の3層で構成される。前章の振舞いを実現する動作概要を以下に示す。

DRC制御層は，まず分散化対象のアプリケーション情報をアプリケーション層から取得する。次に，モノとエッジノードとクラウドノードのトポロジ，ノードとネットワークのリソース状態情報，および実際に流れている通信量情報を実インフラ管理層から取得する。これらの情報を使って，アプリケーションの実行先ノードを決定する（決定方法は後述）。この決定は，アプリケーションやインフラが変化する度に繰り返す。

このように，DRC制御層ではアプリケーション情報とインフラのリソース状態，およびその変化を見ることで，変化に追従して処理の場所を変えることが可能になる。

以下では，これら3層の役割と機能，および追従に必要な入力情報について示す。

● アプリケーション層

分散化対象とするアプリケーションを開発・管

理する層である。

後述のDRC制御機構がアプリケーションを分散化する際に，アプリケーション単位でしか実行場所を変えられない場合，特定ノードにだけ負荷が偏るなどリソース利用効率が低くなる。そこでDRCでは図-3に示すように，アプリケーションの処理フローを処理部品の組合せとして記述し，この処理部品単位で実行先ノードを決められるようにしている。

更に，論理的には一つの処理であっても，物理的にはデータ収集対象デバイスなどの処理対象ごとに処理の実行エッジノードを割り当てられると，各エッジノードのリソースを最大限に使うことができ，システム全体のリソース効率を高められる。そこでDRCでは，各処理部品の分割単位を決めるキー（図-3の集約キー）を指定可能にしている。これにより，一つの処理部品をこの集約キー単位の「部分処理」に分割し，この部分処理単位で実行ノードを割り当てられるようにしている。例えば図-3では，収集処理の集約キーは機器IDであるため，一つの処理部品を機器D1，D2，D3に応じた三つの部分処理に分割し，この単位で実行ノードを割り当てる。

アプリケーションの開発手順を，富士通研究所で開発したプロトタイプ画面を用いて説明する。[4] アプリケーション開発者は，まず**図-4**に示す処理フロー記述領域に処理部品をドラッグ＆ドロップし，部品を線でつないでフロー形式でアプリケーションロジックを記述する。各処理部品の実行ノードの割当ては後述のDRC制御機構が決めるため，開発時に意識する必要はない。次に，プロパティ設定領域において，処理部品の集約キーと，各種パラメーターの値を指定する。最後に，開発したアプリケーションの処理フロー情報および集約キー情報を，アプリ登録ボタンによりDRC制御機構に登録する。

● DRC制御機構

DRCの中核であり，エッジノードやネットワークの負荷変動・余剰リソースに応じてアプリケーションの部分処理の実行場所を最適化する「プランニング技術」と，それに従って実インフラ上のモノや部分処理間をつなぐ「フロー制御技術」の二つの技術から成る。以下，それぞれについて説明する。

(1) 処理実行先プランニング技術

後述する効果が最大となるように，各部分処理への実行ノードの割当てを最適化する技術である。割当てに必要な入力情報を以下に示す。

・分散化して得たい効果を示すポリシー・制約条件
・分散化対象のアプリケーションの処理フロー情報

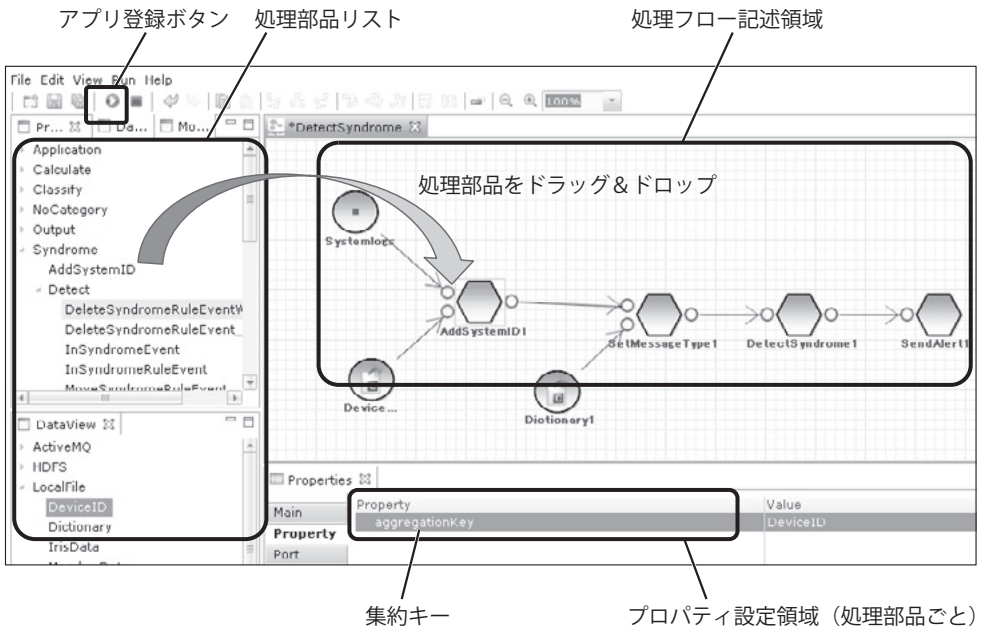


図-4　処理フロー作成画面

・分散化先の実インフラの情報（詳細後述）
・分散化した後の各アプリケーションの稼働ログ

これらの情報を用いて最適な割当て方を決定することをプランニングと呼ぶ。このプランニングを行う機能実体が，図-3のプランナーである。以降，上記の入力情報を使ったプランニング方法を示す。

まず，プランナーは，分散化対象情報としてアプリケーション層で定義した処理フロー情報，および各処理の集約キー情報を読み込む。次に，分散化する際に何を最大化するか（効果）の指標を決める。この効果は，アプリケーションの運用方針に依存するため，DRC側では決められない。そこでプランナーは，求める効果を示す分散化ポリシーを外部から入力可能にしている。以下に，設定可能なポリシー例を示す。

・ネットワーク帯域を許容値まで最大活用
・ネットワーク帯域を最小化
・エッジノード負荷が一定以上なら処理をクラウドへ移動
・モノや端末からのレスポンス性能を最大化

なお，特定の部分処理の実行ノードは，運用者が決めたい場合もある（メール通知処理は，監査証跡のために必ずクラウドに配置するなど）。そこで，各部分処理の実行ノードに関する制約条件も併せて外部から入力可能としている。

更に，分散化対象の処理フロー情報，処理の集約キー情報，および実インフラの機器リスト情報を使ってノードを割り当てる単位を，機器単位の部分処理に分割する（図-3の部分処理グラフ）。

その後，実インフラから集めた情報（具体的には，収集先の機器リスト，ノードのトポロジ，CPU負荷などのノードの状態，帯域上限と残帯域などのノード間のリンク状態の情報）を入力として，部分処理の実行先ノードとしてあり得る組合せパターンを作成し，この中で分散化の効果が最も高い組合せパターンを最適なプランとして決定する。例えば，ポリシーが「ネットワーク帯域を許容値まで最大活用」であれば，トラフィック量が許容値以下でかつ最大となる組合せパターンを最適なプランとして決定する。そして，各部分処理を配備するなどの制御を行い，部分処理を各ノード上で実行させる。[5]

以上のプランニングの手順は1回で終わりではなく，入力に変化が起きる度に繰り返し行う。例えば，ポリシーが「ネットワーク帯域を許容値まで最大活用」で，かつネットワーク余剰帯域が減少した場合に，再プランニングを行い，図-3の部分処理のうち検知（D2）だけ実行場所をクラウドノードN4からエッジノードN3に変更する。これにより，実インフラ状況が変化しても，ポリシーを満たす最適な処理配置が維持される。

なお，各部分処理のCPUリソースや帯域リソースの使用量は，アプリや現場環境に依存するため，実際に動かしてみないと分からない。そこで，プランナーは分散化後の各部分処理の稼働ログも参照して得たリソース使用量情報も入力にすることで，2回目以降のプランの精度向上を可能としている。

（2）フロー制御技術

フロー制御機構は，プランナーが決定した各部分処理のノード割当てのプランに従い，部分処理間で集約キーを持つデータを転送し届けるためのルーティング制御を行う。以下に，フロー制御の方法を示す。

フロー制御機構は，まず図-3の部分処理グラフから，部分処理間をつなぐ論理的なキューを特定する。その後，この論理キューを対向する部分処理が割り当てられたノード上に作成する。例えば，図-3の論理キュー 1aの場合，対向部分処理「収集（D1）」「検知（D1）」が割り当てられたノードN1とN3上に，「収集（D1）」発データを「検知（D1）」に転送するルーティングテーブルのルールを設定する。

プランナーが部分処理の実行先ノードを変えたときは，ルーティングテーブルの差分だけノードに配付してプランナーの決定に追随することで，ノード変更時の制御を効率化している。

● 実インフラ管理機構

部分処理の分散実行先とするノードの構成・能力と，部分処理の割当てを計算するための入力情報を管理する機構である。例えば，以下のような情報を管理する。

・トポロジ（機器，エッジノード，クラウドノードのネットワーク構成）
・ノードの能力・負荷（CPU，メモリなど）
・リンクの能力・負荷（帯域，遅延など）
・データの収集対象の機器リスト

・分散化後の各部分処理の稼働状況（通信量など）

情報の管理手段はDRC技術からは切り離しており，API（Application Programming Interface）経由で情報を取得する。手段の例としては，

(1) 上記トポロジとリンクの能力・負荷の管理実体としては，エッジノードを含めた広域ネットワークの管理システム

(2) ノードの能力・負荷の管理実体としては，エッジノードやクラウドノード上の負荷計測Agentからの収集情報を管理するクラウドのVM管理機構やエッジノードの管理機構

を想定する。

プランニング高速化技術

筆者らは前述のプランナーについて，インフラが大規模化した場合であっても，現場状況の変化検知時に処理の実行ノード再割当てを短時間で行い，変化に追従する技術を開発した。以下に，この技術の特徴について述べる。

前章で説明した部分処理に実行ノードを割り当てる組合せ最適化問題は整数計画問題に属し，一般に問題の規模に対して計算量が指数的に増大するという性質を持つ。このため，部分処理数やノード数の増大に伴って計算時間が膨大になってしまう。例えば，既存の組合せ最適化計算方法により，3000個の部分処理から成るアプリケーションを1000台のノードで分散実行する。この想定で，機器を1台追加するためのプランニング計算を行ったところ，8コアのCPUを搭載したサーバ環境では35時間以上かかった。これでは，現場で起きる変化への追従時間として現実的ではない。

そこで筆者らは，データを現場から収集して処理を行うIoT向けアプリケーションの特徴として，複数の現場データを集約する処理が多いこと，および入力より出力データ量が少ない処理が多いことに着目した。すなわち，分散化する効果の高いノードの組合せを探す探索アルゴリズムに，これらの特徴に合わせた後述の工夫を組み込んだ。こうして，実行先ノードの候補のうち効果の高いものに絞り込むことによって，準最適解（最適ではないがそれに近い解）を高速に探索可能にした。

使える工夫の内容は，分散化に求める効果（高速化や低コスト化など）に依存するが，以降では，求める効果がネットワーク帯域の最小化の場合について，**図-5**を用いて説明する。[6),(7)]

(1) ネットワークの経路に応じた絞込み探索

現場からクラウドまでの最短経路上で処理を行えば，データの遠回りがなくなるため，ネットワークリソースの利用効率が高い。特に，複数の現場からのデータを入力として集約する部分処理は，各現場からの最短経路が重なるトポロジ上の合流点となるノードで処理を実行する。このようにできれば，集約のための迂回がなくなるため，同様にネットワークリソースの利用効率が高くなる。

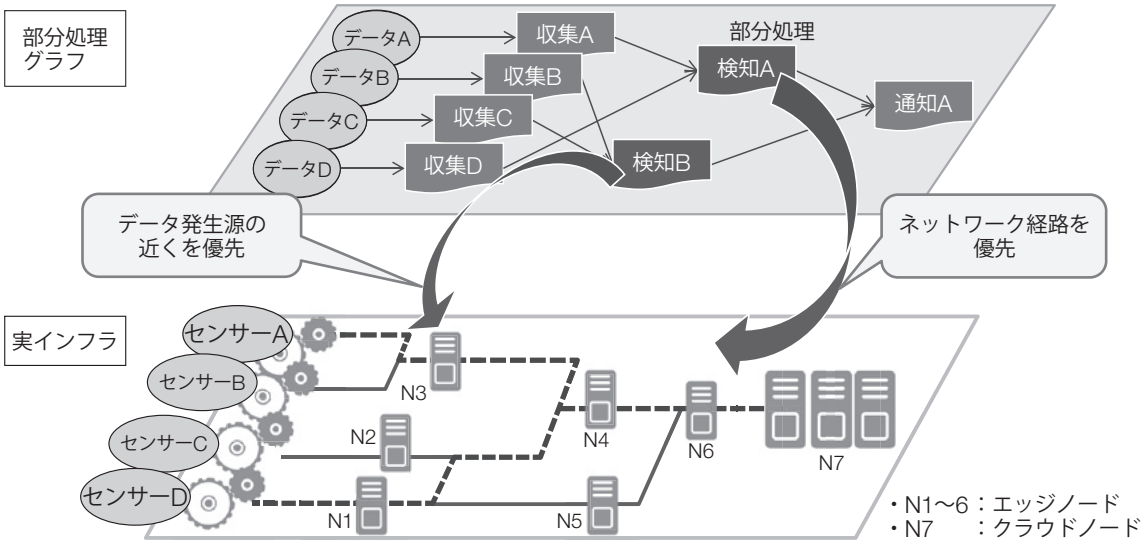


図-5　部分処理の実行先の絞込み探索

そのため，各部分処理のノード探索開始場所を現場からクラウドまでの最短経路（図中の破線）上に絞り，かつクラウドを目的地とした経路の合流点（図中のN6やN4）から順に探索する。こうすることで，より効果の高いノードから探索でき，探索時間を短縮できる。

(2) 処理のデータ入出力量比に応じた絞込み探索

入力データ量に対する出力データ量の割合が小さい部分処理は，なるべく出力データが大量のモノや処理に近いノードを割り当てる。これによって，より小量なデータは迂回させても，結果的に全体のトラフィックを削減できる。

そのため，出力データ量の割合が一定以下の部分処理については，データ発生源近くのノード（図中のN1，N2，N3）から順に探索することで，より効果の高いノードから探索でき，探索時間を短縮できる。

これら二つの探索ロジックは，各部分処理の入出力量比によって使い分ける。

変化への追従性能評価

前章の手法をシミュレーションで評価した。以下，アプリケーション登録後のプランナーの初回計算時間と，分散化済みの状態で現場環境が変化したときのプランナーの再計算時間の評価結果を示す。

(1) 評価1：初回プラン計算にかかる時間

DRC制御機構にアプリケーションを登録した後，各部分処理に関する初めてのプランニングに要する計算時間を評価した。

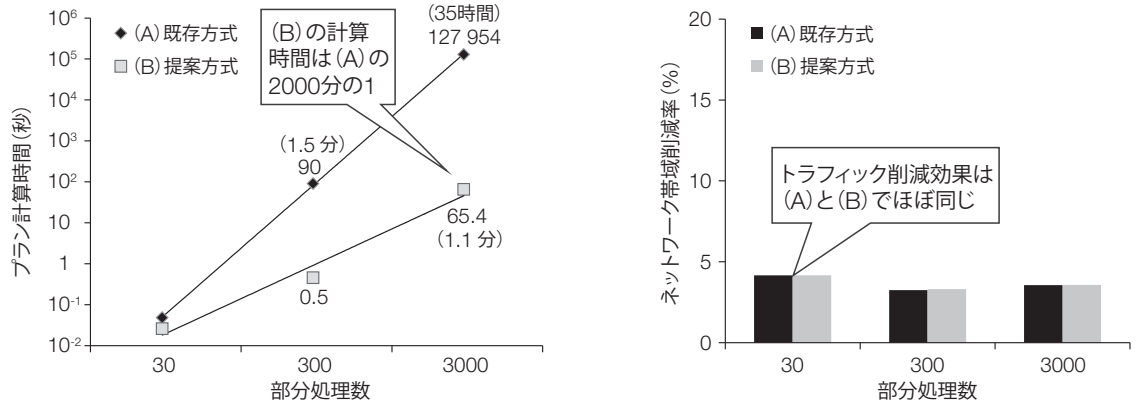
想定するシステム構成は図-5のとおりである。アプリケーションの部分処理数は30 ～ 3000個で，エッジノード数は1000個とした。また分散化ポリシーはネットワーク帯域最小とし，各処理ノードの負荷が一定値以下となるような制約条件を与えた。

以上の条件で，各部分処理の最適なノード探索方式として，(A) 探索開始ノードをランダムに決めて最適ノードを探索する既存方式と，(B) 前節の工夫を用いて探索する提案方式の二つを用い，計算時間とネットワーク利用帯域の削減率を比較評価した結果を**図-6**に示す。帯域の削減率とは，全部分処理をクラウド上で実行した場合のネットワーク利用帯域に対する削減率を示す。

同図 (a) により，部分処理数3000個の場合の計算時間は，(A) 既存方式では35時間以上かかるが，(B) 提案方式では65秒で済むことが確認できた (約2000分の1)。その一方で，同図 (b) によりトラフィック削減効果は (A) (B) で同じであることから，提案方式はネットワーク帯域の削減率を落とすことなく計算時間を高速化できていることが分かった。

(2) 評価2：センサー増加に追従するための再計算時間

部分処理が既に分散実行中であるシステムにおいて，現場にセンサーが追加された場合，既に実行中の部分処理に加え，追加センサー用の部分処理も再プランニングが必要になる。このプランニングに要する計算時間を評価した。



(a) 初回プラン計算時間　　(b) ネットワーク利用帯域削減率

図-6　初回プラン計算時間の評価結果

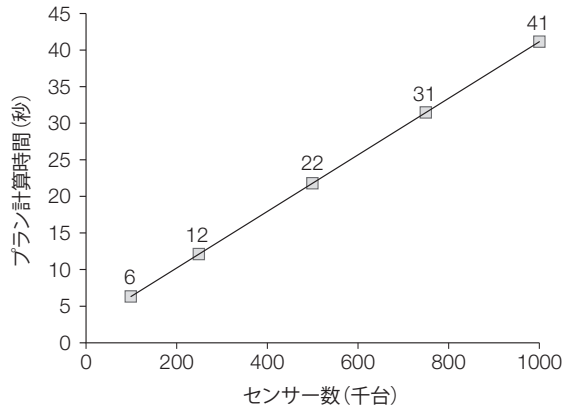


図-7　提案方式の計算時間の評価結果

想定するシステム構成は（1）と同じく図-5の構成である。センサーは，追加前はシステム全体で10万～100万個が稼働しており，途中でその0.2％に相当する数を追加する条件とした。

以上の条件で，提案方式の計算時間を評価した結果を**図-7**に示す。（B）提案方式では，センサー100万個のシステムにおいて，0.2％のセンサーが増加したときに41秒で再計算が完了した。センサー数と再プランニングの計算時間が線形関係にあるため，大規模システムでも短時間で変化に追従できることが分かった。なお，（A）既存方式の計算時間は，センサー10万個のシステムでも1日以上かかる指数グラフとなり，値のオーダーが全く異なるため図には記載していない。

以上のシミュレーションによる評価の結果，提案手法は，既存のシンプルな探索方法に比べて大幅に計算時間を短縮できることが確認できた。また，部分処理数3000個，エッジノード数1000個程度のシステムであれば，数分のオーダーで最適化の初回計算を完了することも確認できた。更に，稼働中のシステムにおいて，例えば100万センサーのシステムにおいて，最適化し直すための計算時間が1分で完了できることも確認でき，少しずつ変化していくシステムに対しても，現実的な計算時間で変化に追従が可能であることが分かった。

むすび

本稿では，今後のIoTのビジネス活用の本格化に向けて，現場環境の変化に応じて，エッジノードとクラウド間でデータ処理や蓄積場所をダイナミックに最適化することを特長としたダイナミックリソースコントローラー技術を紹介した。この技術により，モノの増減・移動に伴うエッジノードやネットワークの負荷変動の発生時に，データ処理や蓄積の実行場所を変えてシステムの安定性を維持できる。また，ネットワークリソースの余剰帯域を活用するために一部現場のデータ処理の実行場所をエッジノードからクラウドに移し，新たな発見につながる生データをクラウドに集められることを示した。

更に，本技術の特長である，最適な処理場所を計算するプランニングの仕組みについても紹介し，大規模システムにおいても短時間で追従できることをシミュレーション評価により示した。この技術の一部は，FUJITSU Cloud Service IoT Platformにおいて既に実用化済みである。[8]

今後は，利用シーンの拡大に向けて，スマートデバイスもエッジノードとして使えるように，モノとエッジノードの接続関係，およびノード間のトポロジが頻繁に変化するモバイル環境においても，プランナーやフロー制御が対応できるように技術強化を進めていく。

参考文献

（1）総務省：平成27年版情報通信白書.
http://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h27/index.html
（2）エッジコンピューティングの破壊力．日経エレクトロニクス．2015年11月号.
（3）阿比留健一ほか：分散サービス基盤技術によるユーザーの体感品質（QoE）向上．FUJITSU，Vol.66，No.6，p.60-68（2015）.
http://www.fujitsu.com/jp/documents/about/resources/publications/magazine/backnumber/vol66-6/paper06.pdf
（4）Nomura，Y et al.：Massive event data analysis and processing service development environment using DFD. In Services（SERVICES），2012 IEEE Eighth World Congress on，p.80-87，June. 2012.
（5）福田茂紀ほか：センサネットワークへのイベント処理の分散配備．BS-4．ネットワークを支える最適・自律・省電力制御技術，シンポジウムセッション．電子情報通信学会ソサイエティ大会講演論文集，2011（2）.

（6）富士通研究所：クラウド環境を広域ネットワーク上で最適化する分散サービス基盤技術を開発.
http://pr.fujitsu.com/jp/news/2014/03/14.html
（7）広域処理分散自動化におけるスケーラブルな外部環境変化への追従手法の提案．マルチメディア，分散，協調とモバイル（DICOMO2014）シンポジウム，平成26年7月．情報処理学会論文誌，コンシューマ・デバイス＆システム，Vol.5，No.4，31-41，Oct. 2015.
（8）富士通：IoTデータ活用基盤サービス「FUJITSU Cloud IoT Platform」の提供開始.
http://pr.fujitsu.com/jp/news/2015/06/10.html

著者紹介

久保田　真（くぼた　まこと）
ソフトウェア研究所
分散システムソフトウェアプロジェクト 所属
現在，ダイナミックリソースコントローラー技術の研究開発に従事。

野村佳秀（のむら　よしひで）
システム技術研究所
モバイル＆IoTソフトウェア開発技術プロジェクト 所属
現在，ビッグデータ，IoT向け統合開発実行環境の研究に従事。

福田茂紀（ふくた　しげき）
ネットワークシステム研究所
フロントNW運用プロジェクト 所属
現在，ダイナミックリソースコントローラー技術の研究開発に従事。

阿比留健一（あびる　けんいち）
ソフトウェア研究所
分散システムソフトウェアプロジェクト 所属
現在，ダイナミックリソースコントローラー技術の研究開発に従事。