社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

リソース分離型データセンターにおける処理負荷やネットワークの変化を考慮した動的資源管理手法

生駒　昭繁† 　大下　裕一†† 　村田　正幸†

† 大阪大学 大学院情報科学研究科　〒565–0871 大阪府吹田市山田丘 1–5
†† 大阪大学サイバーメディアセンター　〒560–0043 大阪府豊中市待兼山町 1–32
E-mail: †{a-ikoma,murata}@ist.osaka-u.ac.jp, ††yuichi.ohsita.cmc@osaka-u.ac.jp

あらまし　CPU やメモリのような資源がネットワークで接続されて構成されるリソース分離型データセンター（DDC）は効率的な資源利用が可能である。ただし、資源間の通信遅延による実行サービスの性能が低下するという問題があり、実行性能を考慮した資源管理が必要である。また、1 つの資源で処理できる量には限りがあるので、割当資源が処理すべき要求が増加すれば、実行待ち時間が発生する。さらに、通信されるデータ量も増加するので、輻輳による通信遅延の更なる増加も発生する可能性がある。DDC において、性能要件を満たし続けるためには、処理負荷やネットワークの変化が性能に与える影響を考慮した資源の増強や経路の再割当を行う動的資源管理手法が必要である。本研究では、割当サービスに対する需要が動的に変化する環境において、サービスの性能要件を満たし続けるように動的に割当資源の管理を行う手法を提案する。本手法では、処理負荷やネットワークの変化が性能に及ぼす影響の定式化と、各サービスの資源の増強や再割当可能性と追加資源としての重要度に基づいた資源割当コストの定義によって、将来の需要増加の際に必要となる資源が利用可能である状態を形作る。手法の有効性を示すために、DDC に対する動的にサービスの需要が変化する環境での資源割当シミュレーションを行い、従来手法と比較してサービスの性能要件を損なわない資源管理が可能であることを示した。

キーワード　ディスアグリゲーション、データセンターネットワーク、資源割当

Dynamic resource management considering workload and network changes in a disaggregated data center

Akishige IKOMA†, Yuichi OHSITA††, and Masayuki MURATA†

† Graduate School of Information Science and Technology, Osaka University
Yamadaoka 1–5, Suita, Osaka, 565–0871 Japan
†† Cybermedia Center, Osaka University
Machikaneyamatyou 1–32, Toyonaka, Osaka, 560-0043 Japan
E-mail: †{a-ikoma,murata}@ist.osaka-u.ac.jp, ††yuichi.ohsita.cmc@osaka-u.ac.jp

Abstract　Disaggregated data center (DDC) enable efficient resource utilization. However, a DDC faces performance degradation due to resource communication latency, necessitating resource management considering network. There is a limit to the load that can be handled by a single resource. If the load on allocated resources increases, execution latency and network congestion may occur. Threfore, we formulate the impact of workload and network changes on performance and define resource allocation costs based on service demand and resource importance. Based on these, we ensure resource availability for future service demand increases in environment with dynamically changing demand for services. Simulation results demonstrate the our method's effectiveness in maintaining service performance requirements compared to conventional approaches under dynamically changing service demands in a DDC.

Key words　Disaggregation, data center network, resource allocation

1. はじめに

リソース分離型データセンター（以降、DDC）は、従来サーバ内に集約されていた CPU や CPU やメモリ等の資源を独立させ、それらをネットワークでつなぐことによって構成したデータセンターである。サーバ単位ではなく資源単位での管理が可能であり、資源利用率の向上が実現できる [1]。この有効性から、私たちは大規模なデータセンターと比べて保有する資源量が限られているエッジデータセンターへの適用を考えている。

しかし、DDC は各資源がネットワークで接続されるため、資源間の通信遅延によって実行アプリケーションの性能が低下するという問題がある [2]。そこで、私たちはネットワークがアプリケーションの処理性能に及ぼす影響を考慮した、資源割当手法 RA-CNP [3] を提案した。ただし、RA-CNP は実行中のサービスに対する処理量の変化がなく、事前に処理が必要とする資源量が見積もることができることを前提としており、DDC にかかる負荷の変化を考慮できていなかった。1 つの資源で処理できる量には限りがあるので、需要の高まりにより割当資源が処理すべき要求が増加すれば、実行待ち時間による性能低下が発生する。また、DDC での処理の総量が増えれば、通信されるデータ量も増加する。負荷の増加に対処しない場合、輻輳が発生し性能が低下する可能性もある。性能要件を満たし続けるためには、処理負荷やネットワークの変化を考慮した動的資源管理手法が必要である。

本研究では、割当サービスに対する需要が動的に変化し、割当資源やネットワークトラヒックが動的に変化する環境において、サービスの性能要件を満たし続けるように割当資源の増強や資源間の経路の再割当を行う資源管理手法を提案する。本手法では、サービスの需要予測を行い、予測情報と割当資源間の通信遅延と割当資源での実行待ち時間に基づいて、資源の増強/解放や経路の再割当を判断する。また、処理負荷やネットワークの変化が性能に及ぼす影響を定式化する。そして、各サービスの需要に基づいた資源の増強や再割当可能性と、追加資源としての重要度に基づいた資源割当コストを定義し、サービスの性能要件を満たしながらサービスの利用資源の割当コストを最小化できるように割当/解放資源を決定する。これにより、将来の需要増加の際に必要となる資源が利用可能である状態を形作る。

2. 動的資源管理手法

2.1 リソース分離型データセンター

DDC は、計算資源、メモリ資源がそれぞれ独立しており、それらはリンクで接続される。同一種類の計算資源やメモリ資源は一定数ごとに同一のスイッチに接続されている。同一スイッチに接続された計算資源とメモリ資源の集合をそれぞれ計算資源プール、メモリ資源プールと呼ぶ。サービスに対する実行要求の増加による計算資源の枯渇やネットワーク負荷の増加による通信遅延の増大に対応するために、資源モニタ、予測器、アロケータの 3 機能が動作する資源モジュールを構成することで資源の動的管理を行う。各機能の対応を図 1 に示し、各機能の動作を下に示す。

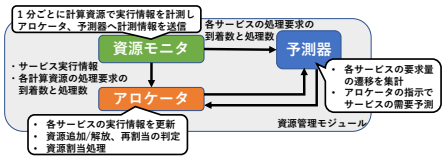

図 1: 資源モジュールの概要

2.2 想定サービス

動的なサービスの資源の追加/解放のために、複数の疎結合な μ サービスで構成されたイベント駆動型のサービスを想定する。このサービスでは、各 μ サービス処理の終了後、次の μ サービスへ実行要求を送信し、実行要求を受け取るとその μ サービスが処理を行うという流れを繰り返すことでサービスを提供する。想定するサービス例を図 2 に示す。

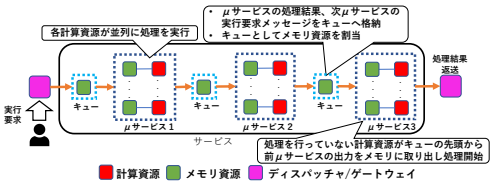

図 2: 想定サービス

2.3 記号定義

各機能が保持する情報の記号定義と、資源管理のためのサービスに関するモデリングを表 1 に示す。

表 1: 記号定義

記号	内容
	アロケータの保持情報
N^s, E^s	DDC ネットワークを表すグラフのノードとリンクの集合
L_n	ノード $n \in N^s$ に対応する資源プール内の利用可能資源数
C^s, M^s	利用可能計算資源、メモリ資源の集合
F_c	計算資源 $c \in N^s$ のクロック周波数
T_n^H	ノード $n \in N^s$ においてパケット全体を次のノードに送信する遅延
T_e^P	リンク $e \in E^s$ の伝搬遅延
R^s	DDC ネットワーク内の資源間の経路の集合
B	DDC ネットワークの帯域幅
P	ページサイズ
$\lambda_{e,n}$	DDC ネットワークリンク $e \in E^s$ の隣接ノード n から発生するパケット到着率
S	実行中のサービスの集合
N_s^v, E_s^v	サービス $s \in S^e$ の資源グラフのノードとリンクの集合
δ_n^R	資源グラフノード $n \in N^v$ に対応する DDC 資源
δ_n^P	資源グラフリンク $e \in E^v$ に対応する DDC ネットワークの経路
C_s^v, M_s^v	サービス $s \in S$ の計算資源とメモリ資源に対応する資源グラフノードの集合
N_s^μ, E_s^μ	サービス $s \in S^e$ の μ サービスグラフのノードとリンクの集合
R_μ^v	μ サービス $\mu \in N^\mu$ の実行に対応する資源グラフの経路の集合
Λ_μ^c	μ サービス $\mu \in N^\mu$ の実行に必要なクロック数
Λ_μ^f	μ サービス $\mu \in N^\mu$ の実行時に発生するページフォールトの発生回数
Λ_μ^p	μ サービス $\mu \in N^\mu$ の実行時に読み込まれるページ数
I_μ, O_μ	μ サービス μ の入力、出力データのサイズ
	資源モニタの計測情報
$U_{\mu,t}$	時刻 t で計測された μ サービス $\mu \in N^\mu$ に対する処理要求の到着率
$\rho_{\mu,t}$	時刻 t で計測された μ サービス $\mu \in N^\mu$ に対する μ サービスの処理率
$\Lambda_{\mu,t}^f$	時刻 t で計測された μ サービス μ の実行時に発生するページフォールト発生回数
$\Lambda_{\mu,t}^p$	時刻 t で計測された μ サービス μ の実行時に読み込まれるページ数
$\lambda_{\mu,t}^w, \lambda_{\mu,t}^r$	時刻 t で計測された μ サービス μ で発生するメモリ書き込み/読み込みパケット数
	予測器の予測情報
$\widehat{U}_{s,t+\Delta t}$	サービス s の予測時点から Δt 時間後の処理要求メッセージの到着率
T^{dur}	予測期間

a） 割当資源とサービスのモデリング

サービスの実行のために割り当てられた資源の対応関係を表す資源グラフとサービスの実行に必要な μ サービス間の対応関係を表す μ サービスグラフを定義する。この 2 グラフは図 3 に示すように互いに対応関係を持つ。この 2 つのグラフ構造は DDC に新規にサービスを立ち上げる場合に構成され、資源の動的な追加や解放を行うごとに更新される。

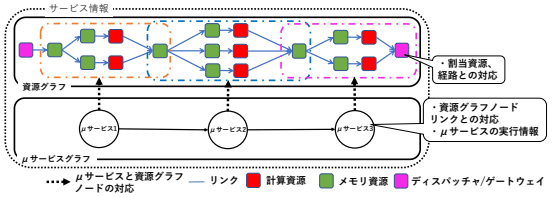

図 3: 資源グラフと μ サービスグラフの例

サービス $s \in S$ の資源グラフは $G_s^v(N_s^v, E_s^v)$ で表される。各ノード $n \in N_s^v$ は割り当てられる計算資源、メモリ資源またはゲートウェイに対応している。有向リンク $e \in E_s^v$ に対応して通信する資源間の関係を表しており、始点から終点までの経路がサービスの実行時のデータの流れを表す。ノード $n \in N_s^v$ とリンク $e \in E_s^v$ は、それぞれ割り当てられる DDC 内資源、経路との対応を持ち、それぞれ $\delta_n^R \in N^s$、$\delta_e^P \subset E^s$ として定義される。

サービス $s \in S$ の μ サービスグラフは $G_s^\mu(N_s^\mu, E_s^\mu)$ で表される。各ノード $\mu \in N_s^\mu$ はサービスを構成する μ サービスに対応しており、その実行順にノードを接続した有向グラフである。各ノード $\mu \in N_s^\mu$ に対応して、資源モニタから送信されるページフォールトの発生回数 Λ_μ^f、メモリから読み込むページ数 Λ_μ^f、実行で発生する通信量 λ_μ^r、λ_μ^w、実行に必要とするクロックカウント Λ_μ^c、入力データサイズ I_μ と出力データサイズ O_μ を情報として持つ。また、各 μ サービスの処理要求が入力される資源から、出力結果を転送する資源までの資源グラフにおける経路の集合 R_μ^v を資源グラフとの対応として保持する。

2.4 資源割当コストの定義

資源の追加/再割当を行う可能性の高い実行中サービスが必要とする可能性の高い資源のコストを高く設定する。

2.4.1 資源の追加/再割当を行う可能性の高いサービス

予測期間における要求量の予測値の最大値がサービスの実行計算資源に対して大きいほど、計算資源の負荷が高まる可能性が高いといえる。そこで、現在の時刻 t における、予測期間 T^{dur} であるときのサービス $s \in S$ に対する資源の追加/再割当可能性 $\eta_{s,t}$ を、

$$\eta_{s,t} = \frac{\max\limits_{0 \le \Delta t \le T^{dur}} \widehat{U}_{s,t+\Delta t}}{|C_s^v|} \tag{1}$$

とする。

2.4.2 サービス実行のために重要な資源

実行中の資源と近接し、利用可能資源が多い資源プールの資源は、サービスの実行性能を損なわないとともに、多くの追加資源が必要になったとしても対応できる。そこで、このような資源を重要な資源として、その重要度を定式化する。

a） サービス実行のために重要な計算資源

実行中のメモリ資源 $\delta_a^R (a \in M_s^v)$ と近接する資源は、低遅延でメモリと通信できるので重要である。また、周波数 F_c が高い計算資源であるほど、高速な処理が可能であるため重要である。よって、サービス s に対する、計算資源 $c \in N^s$ の重要度 $\gamma_{s,c}$ は、

$$\gamma_{s,c} = \frac{1}{\sum_{a \in M_s^v} H(\delta_a^R, c)} \cdot F_c \cdot L_c \tag{2}$$

である。

b） サービス実行のために重要なメモリ資源

あるサービス s のために追加されるメモリ資源 $m \in M^s$ は、キューとして働くメモリ資源 $\delta_a^R (a \in M_s^v)$ または、新規に追加される計算資源 $b \in C^s$ と通信を行う可能性があり、これらの資源と近接する資源は重要である。よって、メモリ資源 $m \in N^s$ の実行資源としての重要度 $\zeta_{s,m}$ は、

$$\zeta_{s,m} = \left\{ \frac{1}{\sum_{a \in M_s^v} H(m, \delta_a^R)} + \frac{1}{\sum_{b \in C^s} H(m, b)} \right\} \cdot L_m \tag{3}$$

である。

2.4.3 サービス実行のために重要なリンク

ホップ数や通信量が小さいほど、遅延は小さく、輻輳も発生しにくくなる。そこで、経路再割当や資源追加時に通信を行う可能性のある資源間の最短経路上のリンクとなる可能性が高く、通信量の少ないリンクを重要であるとする。ここで、リンク $e \in E^s$ が、ある資源 $a, b \in N^s$ 間の最短経路上のリンクとなる可能性を $\theta(a, b, e)$ は、近接する a, b 間の最短経路のうち、リンク e を通過する最短経路の割合として導出する。よって、資源 a, b 間の 最短経路数を $\phi(a, b)$、そのうちリンク e を通るものを $\phi(a, b|e)$、最短ホップ数を $H(a, b)$ として、

$$\theta(a, b, e) = \frac{\phi(a, b|e)}{\phi(a, b) \cdot H(a, b)} \tag{4}$$

とする。これをもとに、あるサービス s において、資源追加時に通信を行う可能性のある資源間は、実行中メモリ資源と利用可能な計算・メモリ資源間、利用可能な計算資源とメモリ資源間であり、経路再割当する可能性のある資源間は、実行中の資源間である。これらの資源のペアの集合を R^p として、実行資源としての重要度 $\kappa_{s,e}$ は、

$$\kappa_{s,e} = \frac{\sum_{a,b \in R^p} \theta(a, b, e)}{\lambda_{n_e^s, e} + \lambda_{n_e^d, e}} \tag{5}$$

である。

2.5 資源割当コスト定義

資源追加/再割り当て可能性 $\eta_{s,t}$ が高く、実行資源としての重要度 γ、ζ、κ が高い資源の割り当てをさける。よって、時刻 t における計算資源 $c \in N^s$、メモリ資源 $m \in N^s$、リンク $e \in E^s$ のコスト $\mathcal{C}_{c,t}$、$\mathcal{M}_{m,t}$、$\mathcal{E}_{e,t}$ を、

$$\begin{aligned} \mathcal{C}_{c,t} &= \textstyle\sum_{s \in S} \{\eta_{s,t} \cdot \gamma_{s,c}\} \\ \mathcal{M}_{m,t} &= \textstyle\sum_{s \in S} \{\eta_{s,t} \cdot \zeta_{s,m}\} \\ \mathcal{E}_{e,t} &= \textstyle\sum_{s \in S} \{\eta_{s,t} \cdot \kappa_{s,e}\} \end{aligned} \tag{6}$$

とする。

2.6 実行時間の見積もり

μ サービス μ に対応する資源を結ぶ、資源グラフの経路 $y \in R_\mu^v$ は、μ サービスの実行におけるデータの流れと対応するので、各リンクに対応する経路の通信遅延と、実行時間は経路上の計算資源の処理時間の和として導出できる。経路 y は、入力データのメモリ資源への転送リンク y_1、μ サービスの実行資源間の通信で利用するリンク y_2、次 μ サービスへの結果転送のためのリンク y_3 に分類することができる。そして、入力データ I_μ の転送時間 $T^O(y_1, I_\mu)$、リンク y_2 に対応する経路を用いて計算資源とメモリ資源で実行される μ サービスの処理時間 $T^E(\mu, y_2)$、出力データ O_μ の転送時間 $T^O(y_3, O_\mu)$ の和が各 μ サービスの実行時間となる。また、ディスパッチャから最初に実行される μ サービスのキューへの要求の転送時間 $T^O(e_s^{start}, I_{\mu^{start}})$ が、実行前の遅延として発生するので、サービスの実行時間 $T(s)$ は、

$$T(s) = T^O(e^{start}, I_{\mu^{start}}) + \sum_{\mu \in N_s^\mu} \max_{y \in R_\mu^v} \left\{ T^O(y_1, I_\mu) + T^E(\mu, y_2) + T^O(y_3, O_\mu) \right\} \quad (7)$$

である。なお、e_s^{start} は、資源グラフの始点をソースノードとするリンク、μ^{start} は最初に実行される μ サービスである。

a） データ転送時間 $T^O(e', D)$

サイズ D のデータの転送時間は伝送遅延 $\frac{D}{B}$ と資源グラフリンク $e' \in E^v$ に対応する通信経路のレイテンシの閾値 $\mathbb{L}_{e'}$ の和として導出する。よって、

$$T^O(e', D) = \tfrac{D}{B} + \mathbb{L}_{e'} \ . \quad (8)$$

である。資源グラフ e' に対応する資源間の経路におけるレイテンシの閾値 $\mathbb{L}_{e'}$ は、資源グラフ e' に対応する経路上のリンクにおいて、通信量が基準値 λ^P の場合の通信遅延 $T^L(e, \lambda^P, n_e^s)$ の和である。よって、

$$\mathbb{L}_{e'} = \sum_{e \in \delta_{e'}^P} T^L(e, \lambda^P, n_e^s) \quad (9)$$

である。通信量 λ の DDC ネットワークリンク $e \in E^s$ において、ノード n から転送する際のレイテンシ $T^L(e, \lambda, n)$ は、リンク e の伝搬遅延 T_e^p と、リンク e が接続するスイッチの処理遅延 $T^R(\lambda, J, T_n^H)$ の和より、

$$T^L(e, \lambda, n) = T_e^p + T^R(\lambda, J, T_n^H) \quad (10)$$

である。なお、J は当該サービスのために利用される、リンク e と同様のノード間に存在するリンクの個数である。また、複数資源間で同一のリンクを利用するとき、衝突回避のためのバッファリングが発生する。このバッファリング遅延は、通信量 λ、利用するリンク数 J、スイッチ処理速度 D の M/D/C 待ち行列モデルを用いて導出する。なお、M/D/C 待ち行列モデルは導出が困難であるため、文献 [4] で提案されている近似式を利用する。よって、

$$T^R(\lambda, J, D) = \left\{ \ D + \frac{\left\{1 + f^Q(\lambda, J, D) g^Q(\lambda, J, D)\right\} h^Q(\lambda, J, D)}{2} \right. \ , \quad (11)$$

である。ここで

$$f^Q(\lambda, J, D) = \frac{\left(1 - \frac{\lambda D}{J}\right)(J-1)\left(\sqrt{4+5J} - 2\right)}{16 \lambda D} \ ,$$

$$g^Q(\lambda, J, D) = 1 - exp\left\{ -\frac{J-1}{(J+1) f^Q(\lambda, J, D)} \right\} \ ,$$

$$h^Q(\lambda, J, D) = \frac{D \cdot (\lambda D)^J}{J \cdot J! \left(1 - \frac{\lambda D}{J}\right)^2} \left[\sum_{i=0}^{J-1} \frac{(\lambda D)^J}{i!} + \frac{(\lambda D)^J}{\left(1 - \frac{\lambda D}{J}\right) J!} \right]^{-1} .$$

である。

b） μ サービス μ の実行時間 $T_{\mu, e'}^E$

資源グラフリンク $e' \in E^v$ の隣接ノードに対応する計算資源 $c' \in N^v$ とメモリ資源 $m' \in N^v$ ペアで μ サービス μ の実行時間 $T^E(\mu, e')$ は、リンク e' に対応する資源間の経路の通信遅延 $\Lambda_\mu^f \cdot T^O(e', P \cdot \Lambda_\mu^p)$ と、計算資源 c' に対応する DDC 内計算資源における実行時間 $\frac{\Lambda_\mu^c}{F_c}$、$\mu$ サービス μ の実行待ち時間の閾値 $\mathbb{W}_\mu$ の総和である。μ サービス μ の計算資源内での実行時間は、実行に必要なクロック数 Λ_μ^c を資源グラフの計算資源に対応する DDC 内計算資源 c のクロック周波数 F_c で除算して導出する。また、実行時の通信遅延は、リンク e' に対応する経路において、1 ページフォールトあたりにかかるデータ転送時間 $T_e^O(P \cdot \Lambda_\mu^p)$ にページフォールトの発生回数 Λ_μ^f を乗算したものである。よって、

$$T^E(\mu, e') = \frac{\Lambda_\mu^c}{F_{\delta_{n_{e'}^s}^R}} + \Lambda_\mu^f \cdot T^O(e', P \cdot \Lambda_\mu^p) + \mathbb{W}_\mu \quad (12)$$

である。

c） 要求の実行待ち時間の閾値 $\mathbb{W}_\mu$

μ サービス μ の要求の実行待ち時間の閾値 $\mathbb{W}_\mu$ は、要求の到着率を基準値 λ^R、計算資源数を μ サービス μ に割り当てられた計算資源数 C_μ、処理時間を μ サービス μ の μ サービス実行時の資源グラフの各径路の実行時間の平均値 T_μ^A としたときの要求の到着率を考慮した実行待ち時間 $T^Q(\lambda^R, C_\mu, T_\mu^A)$ である。よって、

$$\mathbb{W}_\mu = T^Q(\lambda^R, C_\mu, T_\mu^A) \quad (13)$$

なお、T_μ^A は、μ サービス μ のすべての実行資源の組み合わせ R_μ^v 内の処理に該当する資源グラフリンクの実行時間 $T^E(\mu, y_2)$ の平均値である。要求の到着率を考慮した実行待ち時間 $T^Q(\lambda^R, C_\mu, T_\mu^A)$ は、M/M/C 待ち行列モデルを用いて導出する。本研究では、1 つの計算資源が μ サービスを実行でき、各 μ サービスのために割り当てられた計算資源は並列的に処理を実行すると仮定する実行待ち時間 $T^Q(\lambda, J, D)$ は、

$$T^Q(\lambda, J, D) = \left(\tfrac{J}{D} - \lambda\right)^{-1} \left(1 + \left(1 - \tfrac{\lambda D}{J}\right) \left(\tfrac{J!}{(\lambda D)^J}\right) \sum_{k=0}^{J-1} \tfrac{(\lambda D)^k}{k!}\right)^{-1} \quad (14)$$

である。

2.7 資源割当問題の定義

制約と目的関数を定義し、将来の需要増加の際に必要となる資源の利用を最小化することを目的とした資源割当問題を定義

する。

a） 割当資源の制約

DDC 内の資源は、任意のサービス $\forall s \in S$ の 1 つの資源グラフノード $n' \in N_s^v$ にのみ対応する。よって、

$$\forall s \in S, \forall n \in N_s^v \quad \sum_{s' \in S} \sum_{n' \in N_{s'}^v} 1_{\delta_n^R = \delta_{n'}^R} = 1 \tag{15}$$

このとき、$1_{\delta_n^R = \delta_{n'}^R}$ は $\delta_n^R = \delta_{n'}^R$ を満たすとき 1 であり、そうでない場合に 0 である。また、任意のサービス s の資源グラフの計算資源 C_s^v、メモリ資源 M_s^v、リンク E_s^v は DDC 内の資源に対応する。つまり、DDC 内の計算資源、メモリ資源、経路の集合 C^s、M^s、R^s として、

$$\begin{aligned} &\forall s \in S, \forall c \in C_s^v \quad \delta_c^R \in C^s \\ &\forall s \in S, \forall m \in M_s^v \quad \delta_m^R \in M^s \\ &\forall s \in S, \forall e \in E_s^v \quad \delta_e^P \in R^s_{\delta^R_{n_e^s}, \delta^R_{n_e^d}} \end{aligned} \tag{16}$$

である。

b） 資源間のレイテンシ制約

実行中サービス $s \in S$ の資源グラフのリンク $e' \in E_s^v$ に対応する資源間の経路のレイテンシ $T^L(e, \lambda_{n_e^s,e}, n_e^s)$ が閾値 $\mathbb{L}_{e'}$ 以下である必要があるので、

$$\forall s \in S, \forall e' \in E_s^v \quad \sum_{e \in \delta_{e'}^P} T^L(e, \lambda_{n_e^s,e}, n_e^s) \leq \mathbb{L}_{e'} \tag{17}$$

である。

c） μ サービスの実行待ち時間制約

実行中サービス S の μ サービスの実行待ち時間が閾値以下である必要があるので、現在の時刻 t における、各 μ サービスに対する要求の到着数 $U_{\mu,t}$ をもとに、μ サービスの実行待ち時間 $T^Q(U_{\mu,t}, C_\mu, T_\mu^A)$ が閾値 $\mathbb{W}_\mu$ を越えなければよいので、

$$\forall s \in S, \forall \mu \in N_s^\mu \quad T^Q(U_{\mu,t}, C_\mu, T_\mu^A) \leq \mathbb{W}_\mu \tag{18}$$

である。

d） 時 間 制 約

サービスの性能要件を満たすために、実行中サービス $s \in S$ の実行時間の見積もり $T(s)$ が許容時間 T_s^t 以下である必要があるので、

$$\forall s \in S \quad T(s) \leq T_s^t \tag{19}$$

である。

e） 目 的 関 数

割り当て時刻 t における、割り当てた計算資源、メモリ資源、経路上のリンクのコスト $\mathcal{C}_{\delta_c^R,t}$、$\mathcal{M}_{\delta_m^R,t}$、$\mathcal{E}_{e,t}$ が最小となるように割り当てるので、割り当て処理を行うサービス s について、

$$minimize \quad \sum_{c \in C_s^v} \mathcal{C}_{\delta_c^R,t} + \sum_{m \in M_s^v} \mathcal{M}_{\delta_m^R,t} + \sum_{e' \in E_s^v} \sum_{e \in \delta_{e'}^P} \mathcal{E}_{e,t} \tag{20}$$

である。資源割当問題は、二項組み合わせ最適化問題であり、この問題に基づく資源割当問題は NP 困難であることが証明されており、メタヒューリスティクス手法が用いられてきた [5]。本研究では、アントコロニー最適化（ACO）を用いてこの問題を解く。私たちは、RA-CNP において、同様の問題を ACO において導出している。そこで、RA-CNP と同様の方法で上記の資源割当問題の解を導出する。

3. 評　　価

実行サービスに対する要求量の変化に備えた資源割当やサービスの監視によって、要求量の変化による実行性能の変化に対応できることを示す。

3.1 比 較 手 法

提案手法の比較のために、提案手法、RA-CNP と比較する。RA-CNP は、将来の資源要求に備えた割り当てを行うが、実行サービスに対する資源追加に関しては考慮しておらず、これと比較することで実行サービスを考慮した資源割当の有効性を示す。なお、これらの手法の資源割当・解放の決定基準は同一であるとする。

3.2 評 価 環 境

a） 提供サービス

μDDC では、以下に示す 3 種類のサービスが提供されると想定する。

- サービス 1 セマンティックセグメンテーション [6]
- サービス 2 3D 物体検出 [7]
- サービス 3 セマンティックセグメンテーション [6] と鳥瞰図作成 [8] をもとにした周辺環境の同定

各プロセスはそれぞれ車両から送信される情報の前処理、推論、処理結果の後処理の 3 種の μ サービスによって構成されており、実行に必要なクロックカウント、メモリと CPU 間で発生する通信量、ページフォールトの数について表 2 に示す。表の値は、20 回実行した時の平均値と標準偏差の和である。

表 2: 各サービスの実行情報

	セグメンテーション	3D 物体検出	鳥瞰図作成
許容時間 (ms)	100	125	220
μ サービス 1			
クロック数	68199102	49289698	181852
書込/読込パケット到着率 (/ms)	2.58/4.38	1.74/0.0	0.72/2.61
ページフォールトの発生数	1977.88	144.56	31.26
ページフォールト当たりの転送ページ数	2.78	2.58	2.36
μ サービス 2			
クロック数	33095402	52219370	147762478
書込/読込パケット到着率 (/ms)	3.4/3.99	0.77/5.43	8.36/10.36
ページフォールトの発生数	1112.48	6832.98	7532.65
ページフォールト当たりの転送ページ数	1.4	1.86	2.75
μ サービス 3			
クロック数	79239	59155594	399211
書込/読込パケット到着率 (/ms)	1.39/0.0	0.81/0.0	3.72/3.13
ページフォールトの発生数	66.56	244.36	699.43
ページフォールト当たりの転送ページ数	2.09	1.31	3.82

b） 閾値の設定

レイテンシ、実行待ち時間の閾値を表 3 にのように設定する。

表 3: 閾値の設定

	レイテンシの閾値 (ms)	μ サービスの待ち時間の閾値 (ms)					
μ サービス		1	2	3	4	5	6
サービス 1	0.0047	36.43	16.58	0.34	-	-	-
サービス 2	0.0018	15.50	30.24	18.96	-	-	-
サービス 3	0.0019	26.21	12.44	0.15	0.11	73.80	1.50

c） 要求の生成

24 時間のドイツのケルン市の 1 秒ごとの車両位置を示すデータセット [9] に基づき、各サービスに対する要求を生成する。このとき、μDDC は接続する基地局から 1km 範囲内にサービスを提供すると仮定し、ケルン市を 2km ごとに格子状に区切り、その中から 3 つの範囲における車両数情報をもとに計測を行い、車両数が多い順にそれぞれ高負荷範囲、中負荷範囲、低負荷範囲の 3 範囲を設定する。なお、各車両が利用するサービスはランダムに決定され、計測においては、車両数が増加する朝 5 時から夜 20 時までの 15 時間における車両数情報をもとに計測する。

d） 予　　測

過去のサービスの要求量の監視データをもとに学習したモデルを用いて、将来の要求量を予測する。本研究では、予測の一例として Temporal Fusion Transformer モデルを用いた予測器を考え、構築したモデルの予測情報を利用する。予測器では、過去 120 秒間の需要の変化をもとに、将来の 60 秒間の 1 秒間の需要を予測する。高負荷、中負荷、低負荷範囲の平均絶対値誤差は、それぞれ 6.97、2.97、2.98 である。

e） ネットワーク

各資源が交互に配置された 3 × 3 の 2D トーラスネットワークを構成する。各スイッチには計算資源プールとメモリ資源プールが接続されており、それぞれ 24 個、20 個の資源を保有する。計算資源のクロック周波数は 3.4GHz である。スイッチ間をつなぐリンクの数は 3 つであり、1 リンクあたりの伝搬遅延は 0.025μs である。帯域幅は 50Gbps、ページサイズは 4KB に設定する。

3.3 評 価 結 果

3 つの範囲について評価を行ったときの、サービスの許容時間を満たす資源割当ができなかった回数を表 4 に示す。

表 4: 各範囲において実行時間が許容時間を上回った秒数

	高負荷範囲	中負荷範囲	低負荷範囲
提案手法	57	14	0
RA-CNP	2105	510	1

結果から、提案手法においてはどの環境においても、RA-CNP と比較して実行時間が許容時間を上回った秒数が少なく、負荷や通信量の変化に対応できていることがわかった。さらに、最も高負荷な環境では、提案手法と RA-CNP で 37 倍程度の差があり、サービスの需要や実行資源との接続を考慮した資源割当によって、資源の増強や再割当が必要な時に必要な資源が存在する状態を作り出すことに成功しているといえる。また、評価を行った 15 時間のうち、サービスの許容時間を満たすことができなかった期間は最も高負荷な環境でも 57 秒であり、その期間もわずかであるといえる。

4. お わ り に

本稿において、本研究では、割当サービスに対する需要が動的に変化する環境において、サービスの性能要件を満たし続けるように動的に割当資源の管理を行う手法を提案した。本手法では、サービスの需要予測を行い、予測情報と割当資源間の通信遅延と割当資源での実行待ち時間に基づいた資源の増強/ 解放や経路の再割当の判断、処理負荷やネットワークの変化が性能に及ぼす影響を定式化、各サービスの需要に基づいた資源の増強や再割当可能性と、追加資源としての重要度に基づいた資源割当コストの定義を行い、サービスの性能要件を満たしながらサービスの利用資源の割当コストを最小化できるように割当/ 解放資源を決定する。結果として、負荷が動的に変化する環境において、サービスの性能が損な割れる時間を従来手法と比較して小さく抑えることができることを示した。

今後の課題としては、様々な種類の資源間の通信への対応である。本研究では、複数 μ サービスとキューによって構成されたサービスを対象としており、通信する資源ペアの種類やその通信の性質は事前に把握可能であるという前提を置いている。実際のサービスを考慮すると、計算資源同士の通信や、多対多の通信のような形式も考えられ、その際にも対応できるような実行性能の定式化や資源割当コストの定義が必要である。

謝辞　本研究は JSPS 科研費 JP24K14914 の助成を受けたものです。

文　　献

[1] G. Zervas, H. Yuan, A. Saljoghei, Q. Chen, and V. Mishra, “Optically disaggregated data centers with minimal remote memory latency: Technologies, architectures, and resource allocation [invited],” *Journal of Optical Communications and Networking*, vol. 10, no. 2, pp. A270–A285, 2018.

[2] P. X. Gao, A. Narayan, S. Karandikar, J. Carreira, S. Han, R. Agarwal, S. Ratnasamy, and S. Shenker, “Network requirements for resource disaggregation,” in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, (Savannah, GA), pp. 249–264, USENIX Association, Nov. 2016.

[3] A. Ikoma, Y. Ohsita, and M. Murata, “Resource allocation considering impact of network on performance in a disaggregated data center,” *IEEE Access*, vol. 12, pp. 67600–67618, 2024.

[4] T. Kimura, “Approximations for multi-server queues: System interpolations,” *Queueing Systems*, vol. 17, pp. 347–382, 1994.

[5] I. Fajjari, N. Aitsaadi, G. Pujolle, and H. Zimmermann, “Vne-ac: Virtual network embedding algorithm based on ant colony metaheuristic,” in *2011 IEEE International Conference on Communications (ICC)*, pp. 1–6, 2011.

[6] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo, “Segformer: Simple and efficient design for semantic segmentation with transformers,” in *Proceedings of Neural Information Processing Systems (NeurIPS)*, 2021.

[7] Y. Yan, Y. Mao, and B. Li, “Second: Sparsely embedded convolutional detection,” *Sensors*, 2018.

[8] Z. Li, W. Wang, H. Li, E. Xie, C. Sima, T. Lu, Y. Qiao, and J. Dai, “Bevformer: Learning bird’s-eye-view representation from multi-camera images via spatiotemporal transformers,” *arXiv preprint arXiv:2203.17270*, 2022.

[9] S. Uppoor, O. Trullols-Cruces, M. Fiore, and J. M. Barcelo-Ordinas, “Generation and analysis of a large-scale urban vehicular mobility dataset,” *IEEE Transactions on Mobile Computing*, vol. 13, no. 5, pp. 1061–1075, 2014.