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.

リソース分離型マイクロデータセンターにおける将来の資源要求に備えた資源割り当て手法

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
E-mail: †{a-ikoma,y-ohsita,murata}@ist.osaka-u.ac.jp

あらまし　近年、リアルタイム性の高い処理の実行のために、ユーザにより近いエッジに配置可能な比較的小規模のデータセンターであるマイクロデータセンターが提案されている。マイクロデータセンターは、大規模なデータセンターと比べて保有する資源量は限られているため、無駄のない資源利用が重要となる。そこで、CPU やメモリ等の資源同士を独立させ、ネットワークでつないだリソース分離型マイクロデータセンターが提案されている。リソース分離型マイクロデータセンターでは、資源割り当てがアプリケーションの実行性能に大きな影響を持ち、割り当て方によって、十分な資源が存在する場合でもそれらをつなぐネットワーク資源が枯渇してしまう恐れがある。そこで、各資源の重要性に基づいた割り当てコストをもとに資源を割り当てることで、将来のアプリケーションの収容に必要となる資源の利用を回避し、リソース分離型マイクロデータセンターのアプリケーションの収容数を向上させる。本稿では、シミュレーションによって提案手法を評価した。その結果、小規模環境において割り当て可能な限界までアプリケーションを収容できることを示した。

キーワード　マイクロデータセンター、リソース分離、資源割り当て、マルチコアファイバ

Resource allocation method considering future resource requests in a disaggregated micro 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
E-mail: †{a-ikoma,y-ohsita,murata}@ist.osaka-u.ac.jp

Abstract　In recent years, a micro data center has been proposed to execute timesensitive applications. Micro data centers are small data centers located close to users. Micro data centers have a limited amount of resources compared to traditional data centers. Efficient resource utilization is important in a micro data center because a micro data center has a limited amount of resources, compared with large data centers. One approach for achieving efficient resource usage is to disaggregate the resources. In a disaggregated micro data center, resources such as CPUs, memories are connected via a network. In a disaggregated micro data center, resource allocation has a large impact on the performance of the application. Even if there are enough computation and memory resources, the network resources required to connect them may be already used by the previous requests. Therefore, our resource allocation method avoids using the resources that may be required for the accommodation of future applications. In this method, we define an allocation cost for each resource based on the importance of the resource and allocate resources based on the costs. We evaluate our resource allocation method by simulation. The results show that our method can accommodate applications up to the allocation limit in small-scale environments.

Key words　Micro data center, Disaggregation, Resource allocation, multicore fiber

1. はじめに

近年、計算機および計算機によって構成されるネットワークやインターネットの発展は著しく、クラウドを用いた数多くのサービスが提供されている。しかし、クラウドによる処理は、ユーザ側から離れた場所にある大規模なデータセンターと通信を行い、データセンター側で処理をしていくため、データセンターとの通信による遅延が発生し、リアルタイム性が求められる処理には不向きである。そこで、ユーザにより近いエッジに配置可能な比較的小規模のデータセンターであるマイクロデータセンター (μDC) が提案されている [1]。

マイクロデータセンターは、大規模なデータセンターと比べて保有する資源量は限られているため、多用なサービスを提供できるようにするには無駄のない資源利用が重要となる。そこで、リソース分離型 μDC(μDDC) が提案されている。リソース分離とは図 1 のように、CPU やメモリなどのサーバに集約されていたリソースを分解し、それらをネットワークでつなぐことで構成したものである [2]。μDDC は各タスクに必要な分だけ資源を割り当てていくことが可能であり、資源の柔軟な利用が可能である。また、各資源が独立しているため、CPU やメモリのような資源の進化に合わせた柔軟なスケーリングが可能となる。

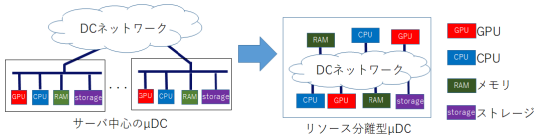

図 1: リソース分離型マイクロデータセンター

μDDC では、資源間の通信遅延による性能低下が問題となっており、特に、CPU とメモリ間との通信遅延が性能に大きく影響する [3]〜[6]。我々はこれまでの研究で、μDDC で CPU とメモリ間の通信におけるネットワークの性能がアプリケーションの実行性能に及ぼす影響について調査を行った [3]。調査の結果、実行時の通信頻度が大きく、帯域幅よりもレイテンシが性能に及ぼす影響が大きいことを示した。レイテンシは資源間の通信経路やその経路の通信量に依存するため、アプリケーションに対してどのように資源を割り当てるかが重要となる。

現在、リソース分離型のデータセンターにおけるリソース割り当て手法がいくつか提案されている [6], [7]。ただし、これらの手法はデータ通信量の増加による通信遅延の増大について考慮しきれていなく、帯域幅のみの考慮であったり、そもそもリンクをアプリケーションごとに占有する形を想定している。より多くのアプリケーションを収容するためには、ネットワーク資源を複数のアプリケーション間で共有し、効率的な資源活用を実現すると同時に、通信遅延を考慮した割り当てが必要である。

そこで、複数のアプリケーション間でネットワーク資源を共有できる μDDC を導入し、その資源割り当て方法を提案する。このアーキテクチャでは、ネットワークはパケットスイッチとマルチコアファイバで構成する。資源割り当ては、要求された時間内にアプリケーションの各タスクが完了するように、計算資源やメモリ資源、ネットワーク資源を割り当てる。

また、割り当てられた資源は、将来到着する要求へのリソースの割り当てに影響を与える可能性があり、。十分な計算資源とメモリ資源が存在する場合でも、それらをつなぐネットワーク資源が枯渇する恐れがある。そこで、資源の重要性に基づいて、各計算、メモリ、ネットワーク資源の割り当てコストを定義し、コストに基づいて資源を割り当てる。これにより、将来のアプリケーションの収容に必要となる資源の利用を回避する。

本稿では、シミュレーションによって資源割り当て手法を評価し、アプリケーションの収容数が向上することを示す。

本稿の構成は以下の通りである。2 章では提案する資源割り当て手法について説明する。3 章でシミュレーションによって提案手法の評価を行う。最後に、4 章でまとめと今後の課題について述べる。

2. 資源割り当て手法

2.1 リソース分離型のマイクロデータセンターのモデル化

本章では、μDDC ネットワークと割り当て要求をモデル化する。

2.1.1 μDDC ネットワークのモデル

μDDC ネットワークをグラフ $G^s(N^s, E^s)$ としてあらわす。なお、N^s と E^s はそれぞれ μDDC ネットワークのノードとリンクの集合である。

a) ノード

このモデルでは、ノードはスイッチ、計算資源、メモリ資源のいずれかを表す。そして、C_n^s と M_n^s は、それぞれノード n 上で利用可能な計算資源とメモリ資源の集合を表す。各計算資源 $c \in C_n^s$ に対して、1 秒あたりの浮動小数点演算回数 (FLOPS) をもとにした性能値 K_c とクロック周波数 F_c、ページサイズ P_c が定義される。さらに、スイッチ能力として、ノード n においてカットスルーによりパケットを次のノードに中継する処理遅延 T_n^N と、パケット全体を次のノードに送信する遅延 T_n^Q を定義する。ただし、ノード n がスイッチング能力を持たない場合、$T_n^N = \infty$、$T_n^Q = \infty$ とする。

b) リンク

本論文では、リソース間の通信経路として各光ファイバーコアを割り当てる。また、複数のアプリケーションの通信のために同一ノード間の複数の光ファイバーコアを使用できるようにする。ここで、利用可能な各光ファイバコアと複数の光ファイバコア群等をすべてリンクと見なす。

各リンク $e \in E^s$ は保有するファイバコア数 N_e^{core} と伝搬遅延 T_e^P を保持し、すべてのリンクの帯域幅を B とする。

2.2 割り当て要求のモデル

割り当て要求は、図 2 に示すように、要求資源の関係を表すグラフ（以下、リソースグラフとする）と、要求アプリケーションのタスクのプロセス間の関係を示すグラフ（以下、プロセスグラフとする）の 2 つのグラフとして与えられる。

リソースグラフはグラフ $G^v(N^v, E^v)$ としてあらわす。な

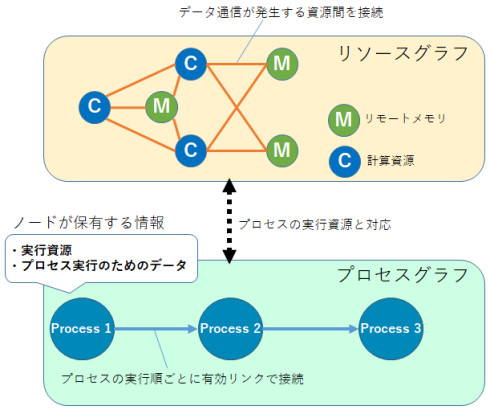

図 2: リソースグラフとプロセスグラフの概要

お、N^v と E^v はノードの集合とリンクの集合である。各ノードは必要な計算資源またはメモリ資源に対応し、各リンクはプロセスの実行時に通信を必要とするノード同士を接続する。

プロセスグラフは有向グラフ $G^p(N^p, E^p)$ で与えられ、N^p と E^p はそれぞれノードの集合、リンクの集合である。各ノード $p \in N^p$ はタスクの実行に必要なプロセスを表し、テスト環境で事前に得たプロセスの実行時のページフォルート数 σ^{pf}、ページフォールトあたりの送信されるページ数 σ^{pn}、クロック数 σ^c、リモートメモリからの読み込み時のパケット到着率 λ_p^r、リモートメモリ書き込み時のパケット到着率 λ_p^w に関する情報を保有する。また、リソースグラフとプロセスグラフの対応関係の情報が含まれ、c_p^v と m_p^v は、ノード $p \in N^p$ に対応するリソースグラフの計算資源とメモリ資源に対応するノードを示す。リンク $e \in E^p$ はプロセスの順序を示す有向リンクであり、各経路はそのタスクの完了に必要なプロセスの実行順を示す。

そして、要求アプリケーション a のタスクの集合を S_a とし、各タスク $t \in S_a$ に対して目標実行時間を定義する。これらの情報をもとに、全てのタスクが目標時間内に完了するように資源を割り当てる。

2.3 資源割り当て問題

本稿では、計算資源、メモリ資源、ネットワーク資源などの各資源に対して資源の重要度に基づく割り当てコストを定義することにより将来のアプリケーションの収容のために必要となる可能性のある資源を残していく。同時に、要求の目標時間を満たす割り当てを行うために、実行時間をモデル化する。そして、これらの定式化をもとに資源割り当て問題を定義する。

2.3.1 資源割り当て情報

要求をネットワーク上のどの資源に割り当てたかを 2 値行列 $\delta_{i,j}^N$ と $\delta_{i,j}^E$ を用いて表す。

a） 計算資源、メモリ資源の割当情報

$\delta_{i,j}^N$ は、要求されたリソースと μDDC のリソースとの対応関係を示す 2 値行列である。リソースグラフノード N_i^v が N_j^s に割り当てられる場合 $\delta_{i,j}^N = 1$ とする。そうでない場合は $\delta_{i,j}^N = 0$ である。

b） リンクの割り当て情報

$\delta_{i,j}^N$ は、リソースグラフのリンクと μDDC ネットワークのリンクとの対応関係を示す 2 値行列である。リソースグラフのリンク E_i^v が E_j^s に割り当てられる場合 $\delta_{i,j}^E = 1$ とする。そうでない場合は $\delta_{i,j}^E = 0$ である。

2.3.2 コスト

目標時間が短いアプリケーションや大規模な計算資源を必要とするアプリケーションを実行できるため、高性能な計算資源は重要である。本稿では、計算資源 c の性能を、利用可能なコア数 $|C_{Node_c}^s|$ とコアの性能 K_c の積として定義する。したがって、計算資源 c のコスト W_c は、

$$W_c = (|C_{Node_c}^s|) \cdot K_c \tag{1}$$

である。なお、$Node_c$ は計算資源 c に対応するノードである。

利用可能なメモリ領域が多いリモートメモリは大量のメモリを必要とする要求に対応することができるため、同一ノード内のリモートメモリ領域が大きいほど、リモートメモリのコストを高く設定する。したがって、メモリ資源 m のコスト W_m は、

$$W_m = |M_{Node_m}^s| \tag{2}$$

である。なお、$Node_m$ は計算資源 m に対応するノードである。

大量の計算資源やメモリ資源を必要とする要求が到着した場合、上記で定義したコストが大きい計算資源やメモリ資源が必要となると同時に、その資源間のネットワーク資源も必要となる。そこで、コストが大きい計算資源とメモリ資源の間のリンクのコストを大きくする。一方、既に任意のアプリケーションの経路として割り当てられているリンクは高速通信を行うことができない。そこで、既に割り当てられているリンクのコストは極小値 ϵ に設定する。したがって、リンク e のコスト W_e は、

$$W_e = \begin{cases} \sum_{c \in C^{cd}, m \in M^{cd}} \left(\frac{N_{c,m}^r(e)}{N_{c,m}^r}\right)\left(\frac{W_c \cdot W_m}{H_{c,m}}\right) & e \notin E^{alc} \\ \epsilon & e \in E^{alc} \end{cases} \tag{3}$$

である。このとき、C^{cd} と M^{cd} は利用可能な計算資源とメモリ資源の集合を表し、E^{alc} は割り当て済みのリンクの集合を表す。$N_{c,m}^r$ と $N_{c,m}^r(e)$ はそれぞれ資源 c、m 間の最短経路数と、その最短経路の中からリンク e を通過する最短経路の数を表す。そして、$H_{c,m}$ は資源 c、m 間の最短ホップ数である。

2.3.3 予測実行時間

要求アプリケーション a の任意のタスク $t \in S_a$ の予測実行時間 T_t^{task} を、タスク t のプロセスグラフのノード $p \in N_t^p$ に対応するプロセスを完了する時間 T_p^{exe} の合計として導出する。

$$T_t^{task} = \sum_{p \in N_t^p} T_p^{exe} \tag{4}$$

T_p^{exe} は、プロセス p の実行時の計算資源内の処理時間とリモートメモリからデータを取得するための通信遅延の和である。

$$T_p^{exe} = \sum_{i=0}^{|N^s|-1} \left\{ \delta_{c_p^p,i}^N \cdot T_{c_i^s,p}^{calc} + \delta_{m_p^v,i}^N \cdot T_{c_p^v,m_p^v,p}^{delay} \right\}$$

なお、$T_{c_i^s,p}^{calc}$ は、計算資源 c_i^s でプロセス p の実行にかかる時間、$T_{c_p^v,m_p^v,p}^{delay}$ は、要求資源 c_p^v、m_p^v が割り当てられた資源間の

プロセス p 実行時の通信遅延である。

a） 計算資源内の実行時間

計算資源 c におけるプロセス p の処理時間 $T_{c,p}^{calc}$ は、プロセス p の完了に必要なクロック数 $\sigma_{c,p}^{c}$ を計算資源 c のクロック周波数 F_c で除算したものである。

$$T_{c,p}^{calc} = \frac{\sigma_{c,p}^{c}}{F_c}$$

b） 通 信 遅 延

メモリからのデータ読み込み遅延 T^{delay} を、1 ページ分のデータを取得するのに必要な遅延を伝播遅延と伝送遅延の和として導出する。

$$T_{c,m,p}^{delay} = \left(\sum_{i=0}^{|N^s|-1} \delta_{c,i}^{N} \cdot \frac{\sigma_p^{pn} \cdot P_{c_i^s}}{B} + T_{E_{c,m}^r,p}^{latency} \right) \cdot \sigma_p^{pf}$$

ここで、$E_{c,m}^r$ はリソースグラフにおけるノード c と m 間のリンク、$T^{latency}$ は伝播遅延と応答時間の和である。また、c_i^s は C_i^s 内の計算資源を表す。

$T_{e,p}^{latency}$ は伝播遅延 T^P と応答時間 T^R の和として導出する。

$$T_{e,p}^{latency} = \sum_{i=0}^{|E^s|-1} \delta_{e,i}^{E} \left(T_i^p + T_{N_{e,i}^s}^{R} (\lambda_{i,N_{e,i}^s}^{s} + \lambda_p^r, N_i^{core}, T_{N_{e,i}^s}^{Q}) \right)$$

ここで、$N_{e,i}^s$ はリモートメモリからデータを読み出す際の通信経路におけるリンク i のソースノード、T^R はバッファリング時間を表す。

バッファリング時間は待ち行列理論でモデル化する。本稿では、各スイッチの処理速度は固定とし、複数の光ファイバコアを割り当てることで、各パケットは利用可能な光ファイバコアがあれば中継することができるとする。そして、パケットはポアソン分布に従って各スイッチに到着すると仮定し、バファリング時間を M/D/C 待ち行列モデルの応答時間としてモデル化する。ただし、M/D/C キューイングモデルにおける応答時間を正確に導出することは困難であるので、文献 [8] で示される近似値を利用する。

$$T_n^R(\lambda, c, D) = T_n^N + \tfrac{1}{2} \left\{ 1 + f^Q(\lambda, c, T_n^Q) g^Q(\lambda, c, T_n^Q) \right\} h^Q(\lambda, c, T_n^Q)$$

ただし、

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

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

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

ここで、パケット到着率を λ、光ファイバのコア数を c、スイッチの処理時間を T_n^Q とする。

2.3.4 資源割り当て問題の定義

a） 資源割り当ての制約

要求グラフのノードと muDDC ネットワークのノードは、1 対 1 の関係にあるので、

$$\begin{aligned} \textstyle\sum_{i=0}^{m} \delta_{i,j}^{N} &\le 1 \\ \textstyle\sum_{j=0}^{n} \delta_{i,j}^{N} &= 1 \end{aligned} \tag{5}$$

である。そして、各要求は、利用可能なリソースの中から割り当てられるので、

$$|M_j^s| + |C_j^s| - \textstyle\sum_{i=0}^{m} \delta_{i,j}^{N} \ge 0 \tag{6}$$

である。

b） 要求アプリケーションの時間制約

要求されたアプリケーションのすべてのタスクは、目標時間内に実行されなければならない。よって、

$$T_t^{task} \le T_{a,t} \quad \forall t \in S_a \tag{7}$$

である。

c） 目 的 関 数

本稿では、資源とリンクの割り当てコストが最小になるように資源を割り当てる。よって、

$$\begin{aligned} minimize & \textstyle\sum_{i=0}^{|N^s|-1} \sum_{j=0}^{|N^v|-1} \delta_{i,j}^{N} (W_{c_j^s} + W_{m_j^s}) \\ & + \textstyle\sum_{y=0}^{|E^s|-1} \left[1_{\sum_{y=0}^{|E^v|-1} \delta_{x,y}^{E} > 0} W_{e_y^s} \right] \end{aligned} \tag{8}$$

が目的関数となる。なお、$1_{\sum_{x \in E^v|} \delta_{x,y}^{E} > 0}$ は、$\sum_{x \in E^v} \delta_{x,y}^{E} > 0$ を満たすときは 1 を、満たさないときは 0 を表す。また、c_j^s と m_j^s はそれぞれ利用可能な計算資源の集合 C_j^s とメモリ資源の集合 M_j^s 内の任意の資源を表す。

以上の制約条件と目的関数を満たす資源を割り当てる。ただし、この問題は非線形整数計画問題であり、NP 困難である。このような問題を解く一つの方法として、メタヒューリスティックな手法があるが、本稿では Ant Colony Optimization(ACO) [9] を用いる。

2.4 Ant Colony Optimization に基づく資源割り当て

定義した資源割り当て問題を解くために ACO を用いる。本手法は、VNE-AE（Virtual Network Embedding Method based on the Ant Colony Optimization）[9] と類似しているが、以下の点で VNE-AE と異なる。(1) 本稿で定義するコストは将来の割り当て要求を考慮しており、VNE-AE とは異なる。(2) VNE-AE は仮想マシンの位置のみを探索し、通信経路は最短経路を使用するが、本手法ではリンクも ACO を用いて探索する。本手法は、エージェントを用いて適切な資源割り当てを探索する。各エージェントは以下のステップを実行する。

（1） 資源探索フェーズ： 要求されたリソースグラフのノードに割り当てるべき利用可能な資源を探索する。探索後、リンク探索フェーズに移行する。

（2） リンク探索フェーズ： 要求されたリソースグラフのリンクを割り当てるリンクを N^{ant} 個のサブエージェントを生成して探索する。探索後、更新フェーズに移行する。

（3） 更新フェーズ：探索した資源ととリンクのコストに基づいてフェロモンを更新する。

上記のステップを N^{itr} 回続け、コストが最小となる最適な割り当てを発見する。

2.4.1 資源探索フェーズ

エージェントは、利用可能な計算機資源とメモリ資源の中から割り当て確率 p_c と p_m に基づいて、選択したノードに割り当てるべき資源を確率的に選択する。

$$p_c = \frac{\tau_c^{\alpha}\left(\frac{1}{W_c^{\beta}}\right)}{\sum_{c \in C_i^{calc}}[\tau_c^{\alpha}\left(\frac{1}{W_c^{\beta}}\right)]} \tag{9}$$

$$p_m = \frac{\tau_m^{\alpha}\left(\frac{1}{W_m^{\beta}}\right)}{\sum_{m \in C_i^{mem}}[\tau_m^{\alpha}\left(\frac{1}{W_m^{\beta}}\right)]} \tag{10}$$

ここで、C_i^{calc} と C_i^{mem} はそれぞれエージェント i の割りて候補となる計算資源とメモリ資源の集合を表す。また、α と β はそれぞれ、フェロモンの重みとコストの重みを表すパラメータである。

2.4.2 リンク探索フェーズ

エージェントはリンクを探索するための N^{ant} 個のサブエージェントを生成する。各サブエージェントは、要求されたリソースグラフのリンクのうち、未割当のものを一つ選択し、選択したリンクに対応する経路を探索する。経路は通信元の資源から順にリンクを選択していく。この処理を宛先資源へのリンクが見つかるまで続けることで経路を決定する。各ステップでは、ノード n からのすべてのリンク候補に対して定義された確率 $p_{e,n}$ に基づいてリンクが選択される。

$$p_{e,n} = \frac{\tau_e^{\alpha}\left(\frac{1}{W_e^{\beta}}\right)}{\sum_{e \in C_{i,n}^{link}}\left[\tau_e^{\alpha}\left(\frac{1}{W_e^{\beta}}\right)\right]} \tag{11}$$

ここで、$C_{i,n}^{link}$ はノード n におけるエージェント i の割り当て候補となるリンクの集合を表す。

2.4.3 フェロモンの更新

全資源の探索後、エージェントはフェロモンを更新する。全資源のフェロモンはフェロモン減少率 ρ $(0 < \rho < 1)$ に基づいて減衰され、各世代の最適解の資源とリンクのフェロモンは、フェロモン増加率 ϕ に基づいて強化される。よって、μDDC ネットワーク内の任意の資源 r のフェロモン τ_r は、以下のように更新される。

$$\tau_r = \begin{cases} \rho\tau_r + \dfrac{\phi}{\sum_{x \in R^{best}} W_x + \sum_{e \in E^{best}} W_e} & r \in R^{best} \\ \rho\tau_r & r \notin R^{best} \end{cases} \tag{12}$$

また、リンク e のフェロモン τ_e は、以下のように更新される。

$$\tau_r = \begin{cases} \rho\tau_e + \dfrac{\phi}{\sum_{x \in R^{best}} W_x + \sum_{e \in E^{best}} W_e} & e \in E^{best} \\ \rho\tau_e & e \notin E^{best} \end{cases} \tag{13}$$

なお、R^{best} と E^{best} はそれぞれ、最適解の資源の集合と最適解のリンクの集合を表す。

3. 評　　価

本章では、提案手法による資源割り当てと最適な資源割り当てを比較することで、手法の有効性を評価する。

3.1 評価環境

3.1.1 μDDC ネットワーク

図 3 に示すような小規模な μDDC ネットワークを使用する。各計算資源は 5 コアで、各メモリは 250 メモリブロックである。また、各リンクは 1 本の光ファイバーコアだけを持つ。その他のパラメータは Table 1 に示す値を設定する。

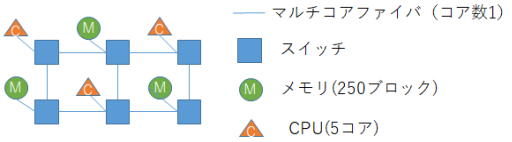

図 3: 評価用ネットワーク

表 1: 評価用ネットワークのパラメータ設定

パラメータ	値
CPU の FLOPS	13.619GFLOPS
伝播遅延	0.025μs
スイッチの処理遅延	3 μs
カットスルー時のスイッチ処理遅延	300ns
ページサイズ	4KB
帯域幅	50Gbps

3.1.2 資源要求

本評価では、表 2 に示す 3 種類の資源要求を生成する。各要求のタスクは 3 つのプロセスを実行することで実行される。なお、ページフォールトはプロセス 3 でのみ発生するとする。資源要求の並びは、表 3 に示した確率に基づいて、各タイムスロットで要求を生成することで行う。なお、p_1、p_2、p_3 はそれぞれ要求 1、要求 2、要求 3 の到着確率を表す。

3.1.3 資源割り当て手法

パラメータを表 4 に示す値に設定し、ACO に基づく資源割り当て手法を実行する。同時に、2.3.4 章で定式化した問題に対して、全探索により最適解を求めることで、ACO によって解を導出できているかを確認する。

3.2 結　　果

各要求列における棄却数を評価した。要求の棄却は目標時間を満たす資源が存在しない場合に発生する。各要求列における平均棄却数を表 5 に示し、各要求列における割り当てた計算資

表 2: 資源要求のパラメータ設定

	要求 1	要求 2	要求 3
目標時間	3000ms	500ms	250ms
必要計算資源量	3	5	5
必要メモリ資源量	3	5	5
プロセス 1			
クロック数	0.035	0.035	0.035
1ms あたりのメモリ書き込み時のパケット到着率	0.00066	0.004	0.01
1ms あたりのメモリ読み込み時のパケット到着率	0.00066	0.004	0.01
プロセス 2			
クロック数	0.054	0.054	0.054
1ms あたりのメモリ書き込み時のパケット到着率	0	0	0
1ms あたりのメモリ読み込み時のパケット到着率	0.00066	0.004	0.01
プロセス 3			
クロック数	2371.33	1960.36	1960.36
1ms あたりのメモリ書き込み時のパケット到着率	3.74	3.8	3.8
1ms あたりのメモリ読み込み時のパケット到着率	7.42	6.86	6.86
ページフォールトの発生回数	67543.25	56661.29	56661.29
ページフォールトあたりに通信するページ数	5.27	4.84	4.84

表 3: 資源要求生成のためのパラメータ設定

要求列	p_1	p_2	p_3
要求列 1	0.8	0.1	0.1
要求列 2	0.4	0.2	0.4
要求列 3	0.1	0.1	0.8

表 4: 提案手法のパラメータ設定

パラメータ	値
資源探索を行うエージェント数	20
経路探索を行うエージェント数	20
探索の繰り返し回数	20
フェロモン減衰率	0.1
フェロモン強化率	100
フェロモンの重み	2
コストの重み	1
フェロモンの初期値	1000

表 5: 各要求列における平均棄却数

要求列	提案手法の最適解	ACO によって導出した解	理論上の最適解
要求列 1	0	0	0
要求列 2	0	0	0
要求列 3	0	0	0

表 6: 各要求列における割り当て計算資源量の最大値と最小値

要求列	提案手法の最適解	ACO によって導出した解	理論上の最適解
	最大値/最小値	最大値/最小値	最大値/最小値
要求列 1	15/13	15/13	15/13
要求列 2	15/13	15/13	15/13
要求列 3	15/13	15/13	15/13

源量の最大値と最小値を表 6 に示す。

結果として、提案手法によって棄却は発生しなく、ほぼすべての計算資源を割り当てていることがわかった。これは、理論値と同様である。つまり、本手法は現在の要求の情報のみを用いているにもかかわらず、効率的に資源を割り当てていることがわかる。また、ACO を用いることで、最適解に近い結果を得ることができることも示した。

4. まとめと今後の課題

本稿ではリソース分離型マイクロデータセンターを対象として、資源の重要度に基づくコスト設定やアプリケーション実行時間のモデル化により、要求の目標時間を満たしつつ、将来の資源要求時に重要となる資源を残しておくことで、将来のアプリケーション要求に備えた資源割り当て手法を提案、評価した。結果として、小規模な環境において、収容可能な限界までアプリケーションを収容できることを示した。

今後の予定としては、より大規模な環境での評価、従来手法との比較を行い、手法の有効性を示すとともに、μDDC に最適なネットワークアーキテクチャについても評価結果をもとに検討していく予定である。

謝辞 本研究の一部は NICT 委託研究「Beyond 5G 超大容量無線通信を支える次世代エッジクラウドコンピューティング基盤の研究開発」によるものである。ここに記して謝意を示す。

文　献

[1] K. Bilal, O. Khalid, A. Erbad, and S. U. Khan, “Potentials, trends, and prospects in edge technologies: Fog, cloudlet, mobile edge, and micro data centers,” *Computer Networks*, vol. 130, pp. 94–120, Jan. 2018.

[2] S. Han, N. Egi, A. Panda, S. Ratnasamy, G. Shi, and S. Shenker, “Network support for resource disaggregation in next-generation datacenters,” in *Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks*, pp. 1–7, Nov. 2013.

[3] A. Ikoma, Y. Ohsita, and M. Murata, “Impact of remote memory and network performance on execution performance of disaggregated micro data centers,” in *Proceedings of 2021 International Conference on Emerging Technologies for Communications*, pp. C2–2, Dec. 2021.

[4] R. Lin, Y. Cheng, M. D. Andrade, L. Wosinska, and J. Chen, “Disaggregated data centers: Challenges and trade-offs,” *IEEE Communications Magazine*, vol. 58, no. 2, pp. 20–26, 2020.

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

[6] 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.

[7] A. D. Papaioannou, R. Nejabati, and D. Simeonidou, “The benefits of a disaggregated data centre: A resource allocation approach,” in *Proceedings of 2016 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–7, Dec. 2016.

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

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