広島市立大学審査博士学位論文

数理モデルを用いた効果的なネットワーク資源配分の予測に関する研究

Research on Prediction of Effective Network Resource Allocation Using Mathematical Models

2024 年 3 月

広島市立大学大学院
情報科学研究科　博士後期課程
情報科学専攻

奥田　尚樹

概　要

今やインターネットは，人々の暮らしや産業を支える社会インフラとなり，インターネット上の仮想基盤，クラウドを用いたサービス提供も不可欠なものとなっている．安全かつ安定したクラウドサービスを提供するための技術開発が進んでおり，クラウドを支える仮想化技術においても，セキュリティ対策，ネットワーク資源配分，障害復旧，サービス品質など多様な観点で研究開発が必要である．

特に，ネットワーク資源配分の予測では，実際にサービスを監視してデータの収集・分析，システムのモデル化によるシミュレーションや近似解法などでの解析が考えられる．しかしながら，どの方法を選択するかは，時間と精度のトレードオフとなる．

本稿では，安定なクラウドサービスを提供するために，サービス妨害対策とサービスの信頼性に焦点をあて，効果的なネットワーク資源配分の予測のための数理モデルを提案する．

サービス妨害対策では，3 章にて，DDoS 攻撃緩和のための数理モデルを提案する．DDoS 攻撃緩和策は， DDoS 攻撃を受けている間でもサービスを継続するため，攻撃外の正常通信のパケットの損失を防ぐことが目的である．加えて，これを実現するためには，通信帯域やパケットを蓄えるための容量などのネットワーク資源の適切な配分が必要となり，これらを予測することも重要となる．本稿では，DDoS 攻撃を受けた場合でも，サービスを継続するために，拡散型フロー制御を用いる DDoS 攻撃緩和方式の数理モデルを提案する．この方式は，DDoS 攻撃元から攻撃対象までのルータ等をオーバレイネットワークで構成し，ノードのバッファあふれまでの時間（緩和時間）を拡散型フロー制御により，延ばすものである．このとき攻撃トラヒックの転送レートの算出が必要となるが，本稿では既存の算出式を改良することにより，従来できていなかった各ノードのバッファ容量のばらつきがある実ネットワークに近い場合

の緩和時間を延ばすことができることを示す．また，DDoS 攻撃の攻撃規模が増大した場合についても，攻撃規模に応じた適切なネットワーク資源の配分を行うことで，十分に緩和の効果を発揮できることを示す．

一方，サービスの信頼性に関しては，4 章にて，マルチクラウドのサービス提供に必要となるネットワーク資源量の推定のための数理モデルを提案する．マルチクラウドにおいて，オーケストレーションツールやコンテナ仮想化を用いたマイクロサービスベースのアプリケーションの開発やサービス提供が盛んになっている．このような環境で開発されたサービスの管理に関する課題の 1 つとして，ネットワーク資源を効率的に割り当て，管理するかという資源管理の複雑さがある．この課題を解決するために，マルチクラウドのモデル化を行い，マルチクラウドの各クラウド等に必要となる資源量の推定を行う．具体的には，サービス提供に必要となるネットワーク資源に関して，オーケストレーションツールで運用されることを前提に，Pod の故障，再生及び，増減を加味した予測を行う．また，これらの数理モデルを作成するにあたり，補給整備システムの稼働率解析に用いられている数理モデルを参考にした．そして，モンテカルロシミュレーションと近似解法の結果を比較することにより，解析の精度や時間を検証する．加えて，具体的なネットワークを想定し，近似解法の計算結果からそのネットワークに必要となる資源予測・管理における有用性を検証する．

Abstract

The Internet has now become a social infrastructure that supports people's lives and industry. The provision of services using cloud computing, a virtual infrastructure on the Internet, has also become indispensable. Technological development is underway to provide secure and stable cloud services. Research and development in virtualization technologies for cloud computing is also needed from various viewpoints such as information and network security, network resource allocation, failure recovery, and service quality.

In particular, for the prediction of network resource allocation, data collection and analysis by actually monitoring the service, system modeling and analysis by simulation, and analysis by approximate solution method based on the modeling are considered. However, which method to choose is a trade-off between time and accuracy.

In this paper, a mathematical model is proposed to predict effective network resource allocation, focusing on DoS countermeasures and service reliability to provide stable cloud services.

For denial-of-service countermeasures, we propose a mathematical model for DDoS attack mitigation in Chapter 3. DDoS attack mitigation measures are intended to prevent the loss of packets of normal communication outside of the attack in order to continue service during a DDoS attack. In addition, it is important to predict the appropriate allocation of network resources, such as communication bandwidth and packet storage capacity, to achieve this. In this paper, a mathematical model of DDoS attack mitigation scheme using diffuse flow control is proposed to ensure continuity of service in the event of a DDoS attack. In this scheme, routers and other devices from the DDoS attack source to the attack target are configured in an overlay network. At this time, it is necessary to

calculate the transfer rate of the attack traffic. The proposed calculation method in this paper by improving in the existing one can extend the mitigation time in the case of actual networks of which the buffer capacity of each node varies. The paper also shows that even when the attack scale of a DDoS attack increases, our proposed method is useful the paper by allocating appropriate network resources according to the attack scale.

For service reliability, a mathematical model is proposed in Chapter 4 to estimate the amount of network resources required to provide Multi-cloud services. Multi-cloud services developed by Microservices-based applications Multi-cloud environment have become increasingly popular using container virtualization. Here, managing Multi-cloud services is complicated in resource management, such as how to allocate and manage computing resources efficiently. To solve this problem, a multi-cloud service model for resource estimation is proposed. Then, an approximate solution method using this model is presented. Specifically, it is to predict Pod failure, regeneration, and increase/decrease based on the overall state of the system, assuming that the network resources are operated by an orchestration tool. In developing these mathematical models, the mathematical models used in the availability analysis of the replenishment and maintenance system were used as references. The accuracy and time of the analysis are then verified by comparing the results of Monte Carlo simulations and approximate solution methods. In addition, the usefulness of the approximate solution method in predicting and managing the resources required for a specific network is verified from the calculation results of the approximate solution method.

目次

図目次

表目次

第 1 章

序　論

1.1　本稿の研究背景

オンラインアプリケーションやストレージサービスをはじめとした様々なサービスがクラウド上で利用され，複数端末からの接続や場所，時間を問わないクラウドサービス利用に対する需要が高まっている．また，クラウドサービスは，社会インフラの一部になっており，車の自動運転やロボットを使った遠隔手術などでも活用されており，安定したサービスが維持できない場合，生命に大きな影響を与える可能性がある．こうしたクラウドサービスの利用では，インターネットなどの通信やクラウドサービスの品質の違いなどを考慮したサービスの維持が求められる．

インターネットにつながる家電やセンサなど IoT 機器の増加とともに，DDoS（Distributed Denial of Service）攻撃などの通信障害を引き起こすサイバー攻撃が問題となっている．DDoS 攻撃は，クラウドサービスの停止にもつながる可能性があり，利用者が安心安全にサービスを受けるためには，このような脅威に対して対策を検討しておく必要がある．また，DDoS 攻撃は，攻撃トラフィックサイズが増幅する傾向にあり，その規模は数テラ bps に到達している．例えば，Amazon Web Services は，2020 年 2 月に 2.3Tbps の DDoS 攻撃を受けたことを報告している [1]．また，コロナ禍によるオンラインサービスの需要の増加により，標的が増えたことから DDoS 攻撃の件数が 2020 年第 1 四半期は前年同期比で 25% 増加という報告も出ている [2] ．さらに，国内においても，攻撃者が標的とする組織宛に DDoS 攻撃を示唆するメールを送り，仮想通貨による送金を要求する事案が発生しており，この攻撃では，攻撃能力を示すためとして，数十 Gbps から 100Gbps 程度の規模で標的に対して DDoS 攻撃

を行ったことが報告されている [3] ．DDoS 攻撃は，その頻度や影響度も大きくなっており，引き続き，DDoS 攻撃の脅威に対する対策が必要である．

DDoS 攻撃の緩和のために，事前に防御すべき（標的の可能性となる）サーバやネットワークの定常的なトラフィックパターンを調査し，それに基づいて事前に緩和対応策を実施する事前対応型と攻撃を確認した後に緩和対応をする事後対応型がある．事前対応型には，Akamai 社のサービスとして提供されているものもある．一方，事後対応型は一定時間トラフィックの観察や，緩和できるまでの時間がかかるものの，事前対応型のような事前準備は不要である．また，事後対応型では，攻撃の検知，攻撃元から標的までの複数の経路で攻撃トラフィックの排除や分散するなどして，標的への攻撃が及ぶまでの時間を延ばすことが重要となる．そのために，ルーティングアルゴリズム，ブロックチェーン，機械学習などを適用した様々な緩和手法が提案されている [4] [5] [6][7] [8] [9][10].

文献 [11] [12] [13] において，物理現象である拡散現象を指導原理とした自律分散制御を用いて，DDoS 攻撃トラフィックの集中を緩和する事後対応型の手法が提案されている（以下，既存方式と呼ぶ）．既存方式 [13] では，標的までの経路上に大量の攻撃トラフィックを緩和するための装置をおき，標的そのものへのトラフィックを緩和する方式である．この方式による緩和により，標的が攻撃の被害を受ける（サービス妨害が発生する）まで時間に猶予をもたせ，その間に攻撃元の特定とフィルタリングなどの対策を講じることを想定している．そのため，緩和可能な時間（以降，緩和時間）が長い方がよいが，既存方式 [13] では，緩和装置におけるトラフィックの分散パターンやバッファ容量が固定した限定的な制御をしていることから，緩和すべき対象のネットワークのトポロジの構成によっては，経路上での緩和装置のバッファの溢れが早期に起こる．すなわち緩和時間が極端に短くなる．この限定的な制御は，実際の緩和装置や動的なトラフィック変化に即しておらず，より柔軟な制御が必要である．

また，クラウドサービスに関する研究として，最適な資源配分のアルゴリズムが提案されている．文献 [14] では，シングルクラウドサービス向けの最適化アルゴリズムをマルチクラウドサービスに適用するために，探索ベース，ランダム探索，Automated Machine Learning (AutoML) の観点から検証を行っている．文献 [15] は，マルチクラウド環境において，サービス配置のための最適なプロバイダを選択することを可能とし，さらに，そのプラットフォーム上で通信遅延と通信コストを最小化するアルゴリズムを提案してる．また，文献 [16] は，マルチクラウドにおいて，Quality of

Service (QoS) を提供しながら，消費電力を最小化し，収益を最大化できる効果的なアルゴリズムも提案している．文献 [17] は，Kubernetes[18] のスケジューリング機能に関する最近の研究を 5 つの領域に分類し，研究の概要，未解決の課題，今後の研究の方向性についてまとめている．この中で，スケジューリング機能に関する今後の Kubernetes[18] の課題として，各アルゴリズムを比較して，評価するためのデファクトスタンダードとなるオープンソースツールが存在しないことが述べられている．その他，マイクロサービスの異常検知やスケジューリング機能など様々な研究が行われている [19][20][21][22].

また，クラウド提供者・利用者にとって，遊休資産を防ぎ，応答時間を短縮するために，適切なネットワーク資源を選択することは，重要な課題である．そのため，ネットワーク資源の使用量やクラウドサービスに対する処理要求の予測は，統計的な手法，最適化アルゴリズム，機械学習などの様々なアプローチでの研究が行われている [23][24][25][26][27][28]．文献 [29] は，クラウドに必要となる将来のワークロードの需要分布を予測する機械学習のモデル（HBNN）を提案している．特徴としては，CPU とメモリなどのデータの不確実性を捕らえ，将来の需要の分布を予測するものである．また，文献 [30] は，動的で変化の激しいクラウドのワークロードを予測するため，予測方法の違うものを組合せて使用するフレームワークである CloudInsight を提案している．CloudInsight は， ユーザーが選択した任意の異なる予測方法を使用することができ，各ローカル予測器の適切な重み（寄与度）を動的に決定できる．また，Auto Regressive Integrated Moving Average (ARIMA)，Support Vector Machine (SVM)，高速フーリエ変換（FFT），Robust Stepwise Linear Regression（RSLR）と CloudInsight を比較することで性能を評価している．今後の課題として，コンテナのオートスケールをサポートするために CloudInsight を改良することが述べられている．

1.2 本稿の目的

本稿では，安定なクラウドサービスを提供するために，サービス妨害対策とサービスの信頼性に焦点をあて，効果的なネットワーク資源配分の予測のための数理モデルを提案する．

サービス妨害対策としては，第 3 章で，DDoS 攻撃間のサービス継続のために，攻撃外の正常通信のパケット損失を防ぐことを目的として，拡散型フロー制御を用いる DDoS 攻撃緩和方式のための数理モデルを提案する．この方式は，DDoS 攻撃元から攻撃対象までのルータ等をオーバレイネットワークで構成し，ノードのバッファあふれまでの時間（緩和時間）を拡散型フロー制御により，延ばすものである．このとき攻撃トラヒックの転送レートの算出が必要となるが，本稿では，既存の算出式を改良することにより，従来できていなかった各ノードのバッファ容量のばらつきがある実ネットワークに近い場合の緩和時間を伸ばすことができることを示す．また，DDoS 攻撃の攻撃規模が増大した場合についても，攻撃規模に応じた適切なネットワーク資源の配分を行うことで，十分に緩和の効果を発揮できることを示す．

一方，サービスの信頼性については，クラウドサービスの安定した運用のために，様々な状況を想定した各ネットワーク環境への最適な資源配置が重要である．例えば，オーバースペックに資源配置をした場合，多くの費用が必要となり，アンダースペックとなった場合，サービスの遅延や停止などが発生する．具体的には，第 4 章において，複数のクラウド，エッジ，オンプレミス環境からなり，Kubernetes[18] などのオーケストレーションツールによって制御されているマルチクラウドを想定し，必要となるネットワーク資源を予測するための数理モデルを提案する．また，この数理モデルの作成にあたり，補給整備システムの保全性解析に使用された解析手法を参考とした．加えて，具体的なネットワークを想定し，計算結果からネットワークに必要となる資源予測・管理における有用性を検証する．

1.3　本稿の概要

本稿の内容を各章に分けて説明する．

第 2 章において，各章で使用する基礎的概念について述べる．

第 3 章において，拡散型フロー制御を用いる DDoS 攻撃緩和方式の有効性評価について，本稿が提案する数理モデルを述べ，本稿の提案する数理モデルと既存の数理モデルで数値計算を行い結果を比較する．また，DDoS 攻撃の攻撃規模が増大した場合についても，攻撃規模に応じた適切なネットワーク資源の配分を行うことで，十分に緩和の効果を発揮できることを示す．

第 4 章において，マルチクラウドのネットワーク資源を推定するための数理モデルを提案する．提案する数理モデルは，補給整備システムの保全性解析に使用された解析手法を用いる．また，具体的なネットワークを想定し，近似解法の計算結果からそのネットワークに必要となる資源予測・管理における有用性を検証する．

最後の第 5 章にて，得られ結果をまとめた結論として，本稿の成果を述べる．

1.4 本稿の関連研究

本稿の内容は，論文末尾に付している「研究業績一覧」で示す発表論文内で行われた研究に基づいたものである．その関係は次の表のようになっている．

章	「研究業績一覧」における論文番号
第 3 章	[1]
第 4 章	[2],[3],[4]

第 2 章

基礎的概念

本章では，各章で使用する基礎的概念について述べる．

2.1　DDoS 攻撃緩和方式の基礎的概念 (3 章関連)

本節では，3 章の拡散型フロー制御を用いる DDoS 攻撃緩和方式で必要となる用語や基礎的概念を説明する．

2.1.1　拡散型フロー制御

拡散型フロー制御 [31] とは，物理学における拡散現象を指導原理とし，ネットワークの輻輳回避を目的とした自律分散型フロー制御である．エンドホスト間で経由されるネットワーク機器 (以下，ノード) が隣接するノードとの相互作用のみで，自律分散的に転送レートの制御を行い，各ノードのバッファ使用量の平滑化を実現する．

ノードの動作モデルを図 2.1 に示す．送信元エンドホスト側を上流，宛先エンドホスト側を下流とすると拡散型フロー制御におけるノードの動作は以下の 4 つである．

- 上流ノードへ自ノードのバッファ使用量とパケットの転送レートを含んだフィードバック情報を送信
- 下流ノードから送られてきたフィードバック情報を受け取り，その情報と自ノードのバッファ使用量を元にパケットの転送レートを計算
- 上流ノードから送られてきた宛先宛のパケットを受け取りバッファリング
- 下流ノードへ自ノードの転送レートに従って宛先宛パケットを転送

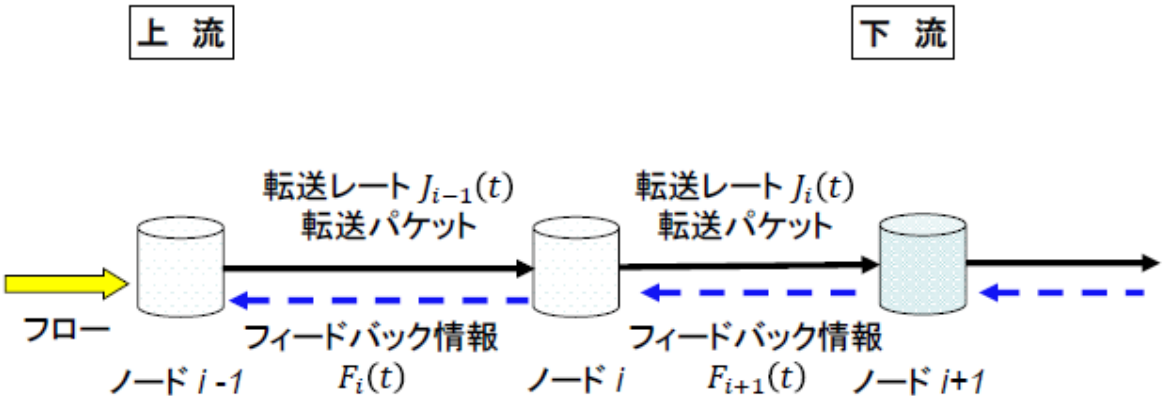

図 2.1　拡散型フロー制御におけるノードの動作モデル

拡散型フロー制御方式における転送レート算出方法について説明する．拡散型フロー制御方式では，ノード i のバッファ使用量 $n_i(t)$ と下流ノードのバッファ使用量 $n_{i+1}(t-d_i)$ の差に応じて，フィードバック情報から求める転送レート $\tilde{J}_i(t)$ は式 (2.1) となる．

$$\tilde{J}_i(t) = r_i(t-d_i) - D_i(n_{i+1}(t-d_i) - n_i(t)) \tag{2.1}$$

ただし，下流ノードから送られてきた転送レート $r_i(t-d_i)$ をフィードバック情報として用い，d_i はノード i とノード $i+1$ 間の伝送遅延時間，D_i は拡散係数である．つまり，ノード i よりも下流ノード $i+1$ の方が混雑している $(n_{i+1}(t-d_i) > n_i(t))$ 場合，転送レートを抑える方向に，一方，下流ノードの方が空いている $(n_{i+1}(t-d_i) < n_i(t))$ 場合，転送レートを上げるといったブレーキとアクセルの役割を果たす．ただし，転送レートは，非負の値であるということや，ノード間で使用できる通信帯域 (L_i) 以下でなければならないことから，実際に利用される転送レート $J_i(t)$ は，式 (2.2) のようになる．

$$J_i(t) = max(0, min(\tilde{J}_i(t), L_i)) \tag{2.2}$$

また，拡散係数 D_i は式 (2.3) で表される．上流ノード数とは，上流側の隣接ノード数である．図 2.1 のような，エンドホスト間で経由するルータをノードとした 1 次元トポロジでは各ノードの上流ノード数は 1 であるため $D_i = 1/2$ となる．

$$D_i = \frac{1}{\text{下流ノード } i+1 \text{ が持つ上流ノード数} + 1} \tag{2.3}$$

ただし，拡散の効果を得るためには，拡散係数は $0 \leq D_i \leq 1/2$ の範囲でなければならない．物理現象としての拡散は，連続媒体中，連続時間で起こるが，実際のネットワークは，離散的 (ノードの空間的配置や制御動作のタイミングが離散的) であり，上記に示した D_i の範囲は離散空間，離散時間の差分計算で求める際に必要な制約条件である．また，D_i が大きいほど拡散する速度が高くなるため，制約条件を考慮すると $D_i = 1/2$ のとき最も早く拡散する (文献 [31] 参考).

2.1.2 オーバレイネットワーク

オーバレイネットワークとは，物理的なネットワークの上に構築された仮想的なネットワークである．オーバーレイネットワーク上のノードは，下位層のネットワークのトポロジーを意識せずに通信することができる．また，下位層にあるネットワーク（例えば物理的なネットワーク）をアンダーレイネットワークと呼ぶ.

オーバーレイネットワークの代表例として，インターネット上に仮想的な閉域ネットワークを構築するインターネット VPN がある．IPsec などのプロトコルを用いて仮想的なネットワークを構築することにより，本来インターネット経由では不可能なプライベート IP アドレスを用いた通信が可能となる.

最近では，ソフトウェアでネットワークを制御する SDN（Software Defined Networking）や SD-WAN においても，オーバーレイネットワークの考え方が採り入れられている.

2.1.3 IX(Internet Exchange Point)

Internet Exchange Point(相互接続点) とは，ISP 事業者やデータセンタ事業者などが相互接続して， 経路情報やトラフィックを交換するための接続点を指し，「IX」または「IXP」と略される．IX に接続すれば，事業者間において個別に回線を準備することなく，複数の事業者と経路交換を行うことが可能となる.

IX では，接続に利用するプロトコル階層として，OSI7 階層参照モデルの第 2 層 (Layer 2; データリンク層) や第 3 層 (Layer 3; ネットワーク層) が利用され，一般的には，第 2 層での相互接続点を IX と呼ぶ．IX を経由した経路交換では，AS 番号を取

得し，BGP により相互の経路情報を交換する．また，安定した相互接続環境を維持するため，接続する機材に一定の要件を課すなど，IX 毎に特徴がある (文献 [32] 参考).

2.2　マルチクラウドの基礎的概念 (4 章関連)

本節では，4 章でのマルチクラウドに必要となる基礎的概念を説明する．

2.2.1　コンテナ型仮想化

コンテナ型仮想化では，物理サーバにインストールされたコンテナ型仮想化ソフトウェアとホスト OS のカーネルの機能によって，仮想化サーバように隔離された実行環境 (コンテナ) を作成し，物理サーバのプロセスの 1 つとして，コンテナが動作する．図 2.2 にコンテナ型仮想化の概要を示す．物理サーバから見たコンテナの実態は，プロセスとして振舞うが，ホスト OS のカーネルの機能によって通常のプロセスより強く環境が隔離されており，他のプロセスとは独立した実行環境を持っている．同じ仮想化技術であるホスト OS 型仮想化 (図 2.3) は OS 単位で仮想化するが，コンテナ型仮想化では，アプリケーション (プロセス) 単位で仮想化するため，軽量な実行環境を実現することができる．コンテナ型仮想化ソフトウェアとしては，Docker や LXC(Linux Container) などがある．

2.2.2　オーケストレーションツール

オーケストレーションツールとは，複数のサーバーに配置されたコンテナなどの仮想化サーバを一括して管理・運用するためのツールである．コンテナには多くのメリットがあるが，複数のサーバーに配置されたコンテナの運用が非常に困難になるというデメリットがある．オーケストレーションツールを使用することで，複数のサーバーに配置されたコンテナの使用による運用負荷を大幅に軽減できる．

主な機能としては，「コンテナの死活監視機能」，「コンテナとホストサーバーの管理機能」，「状況に応じて自動的にコンテナをスケールイン・スケールアウトする機能」，「コンテナの配備に適したサーバーの選定機能」，「障害によりコンテナが停止した場合，そのコンテナを破棄し新たなコンテナを起動する機能」などがある．

また，オーケストレーションツールでは, オープン・ソース・ソフトウェアである Kubernetes[18] が有名である．

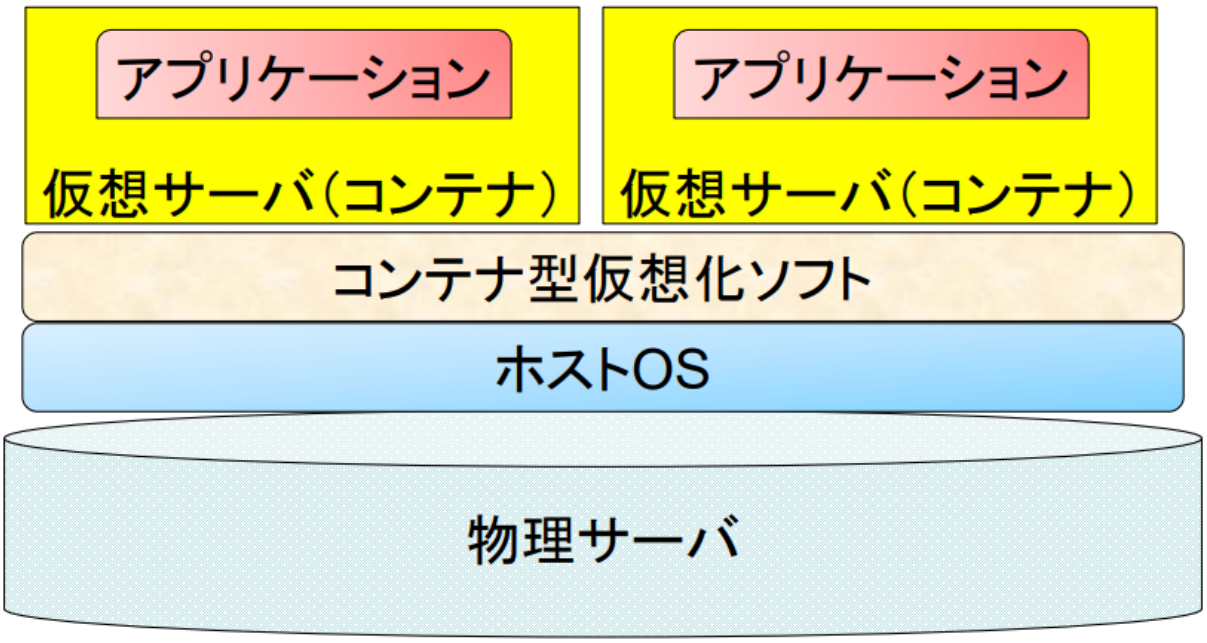

図 2.2　コンテナ型仮想化

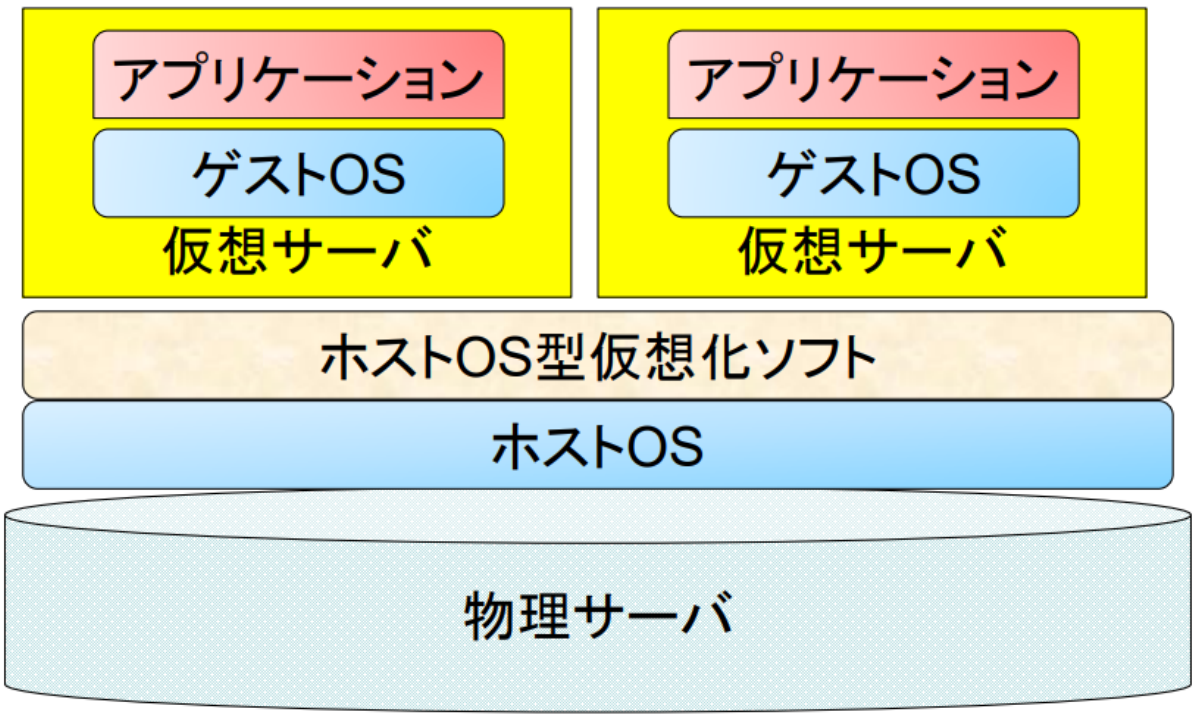

図 2.3　ホスト OS 型仮想化

2.2.3 Kubernetes

Kubernetes[18] は，仮想化技術を用いたシステムを管理するためのオープンソースのオーケストレーションツールである．サーバ（物理及び仮想化サーバ）やコンテナによって構築されてた大規模システムを扱う際，複数のコンテナでデータ共有させつつ，異なるサーバ上に配置したり，過負荷に備えてコンテナを異なるサーバ上に複製し負荷分散を行うなど，コンテナが各サーバ上へ多様に展開される．しかしながら，システムが大規模であるほど，サーバやコンテナの数は膨大となり，それらの接続関係も複雑となるため，管理が困難となる．そこで，Kubernetes[18] を使用することで，コンテナをどのサーバに展開させるか，コンテナ間のデータ共有のさせ方など，複雑なコンテナの管理を指定でき，仮想化技術を用いたシステムを効率的に管理することが可能となる．

また，Kubernetes[18] は，単一または複数のコンテナを 1 つのコンポーネントとして集約した Pod，ノード（仮想化サーバや物理サーバ）の集合体であるクラスタ，Pod を展開するワーカーノード，クラスタを管理するマスターノードの 4 つの単位でシステムを管理する．本稿で使用する Kubernetes[18] の主な管理機能を以下に説明する (文献 [33] 参考).

オートスケーリング

Kubernetes 上にコンテナをデプロイ (ノードに展開すること) する際には，同じコンテナイメージを元にした複数のコンテナ (レプリカ) をデプロイすることで，負荷分散や耐障害性を確保する．また，負荷に応じて，コンテナのレプリカ数を自動的に増減することをオートスケーリングという (図 2.4).

スケジューリング

コンテナをデプロイするノードを決定することをスケジューリングという．例えば，各ノードの CPU の稼働状態などを指標として，コンテナを配置するノードを決定することが可能である (図 2.5).

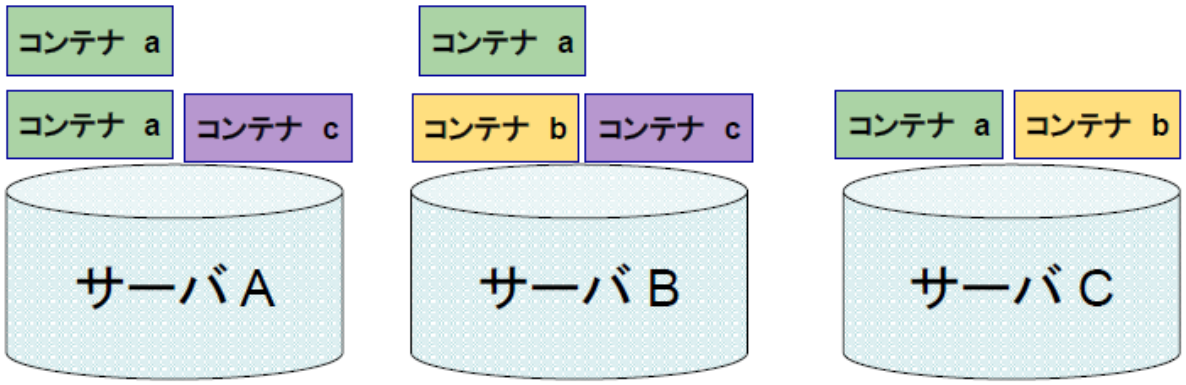

図 2.4　オートスケーリングのイメージ

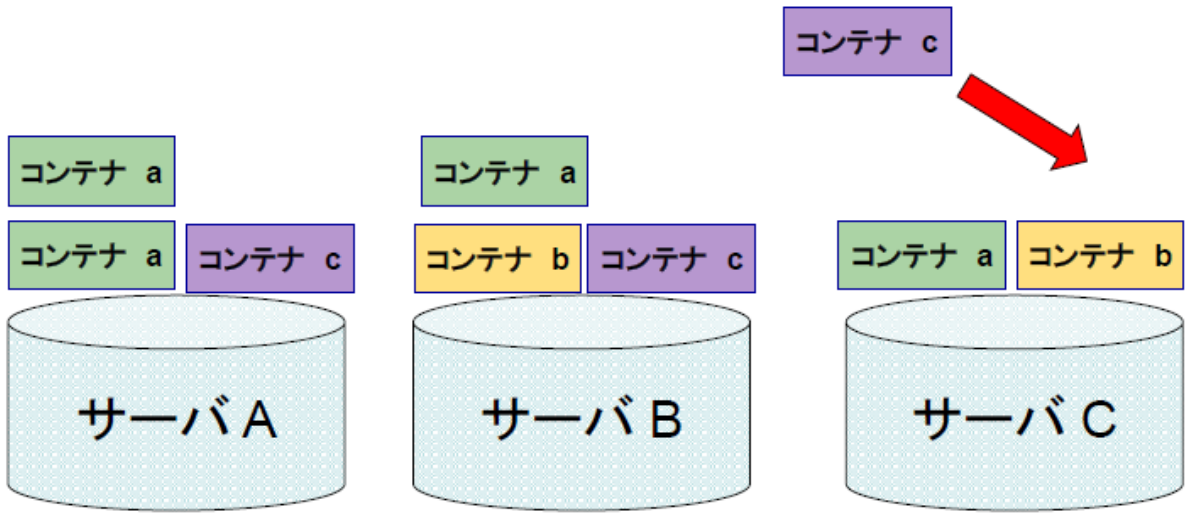

図 2.5　スケジューリングのイメージ

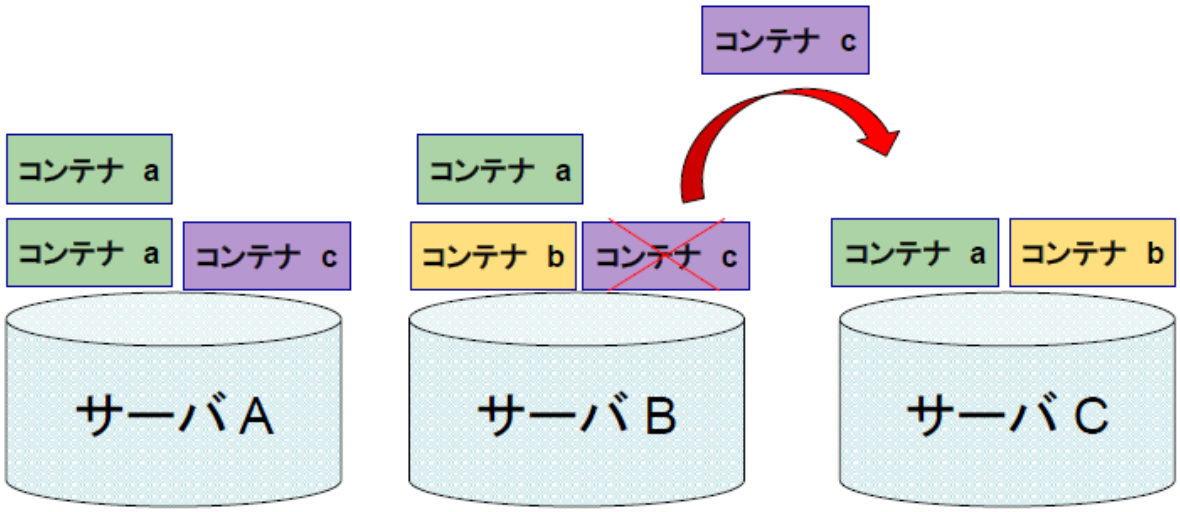

図 2.6　セルフヒーリングのイメージ

セルフヒーリング

耐障害性という観点で，Kubernetes は，コンテナのプロセス監視を行っており，プロセスの停止を検知すると，再度コンテナのスケジューリングを実行することで，自動的にコンテナを再デプロイする．この機能をセルフヒーリングという (図 2.6).

2.3　補給整備システムの解析方法 (4 章関連)

本節では，4 章でのマルチベース及びマルチクラウドのモデル化や近似解法に必要となる基礎的概念を説明する．

2.3.1　概要

複数の運用現場 (ベース) において同一アイテムの運用を行い，その各ベースで予備アイテムの保管，故障アイテムの修理を行うような運用方式は，多くの分野で存在する．例えば，民間航空機や発電所，製造業，鉄道，コンピューターシステムなどの各種補給整備システムがある．このような運用方式で運用されるアイテムの一部には，アイテムに故障が発生した場合，故障アイテムを廃棄するのではなく，故障アイテムを修理したうえで再度アイテムを利用する部品，すなわち修理可能部品が多く含まれる．また，修理可能部品は再利用を前提とするため，一般的に高価なものである．このような修理可能部品を取り扱った補給整備システムのコストの削減や稼働率の向上などの運用効率向上は，重要な問題である．修理可能部品を取り扱った補給整備システムでは，運用効率向上のための代表的な方策として，以下の 2 点が挙げられる（図 2.7）．

１点目の方策として，故障アイテムの修理をアイテムの故障状態で階層的に分け，その各階層でそれぞれの担当する修理を行う方策，すなわち，多段階整備方式がある．例えば，軽微な故障アイテムは運用現場で修理を行い，重度の故障アイテムは中央の修理施設で修理を行う．このような，多段階整備方式で運用される補給整備システムは，修理工程を多段階に分けることにより整備体制の集約を行うことができ，それにより修理人の人件費や整備施設の投資費用を抑え，効率的な整備体制を執ることができる．

もう一方の方策として，ベース間補給方式，すなわち，ベース間でのアイテムのやり取りを行う方式がある．例えば，あるベースでアイテムの故障が発生し，このベースに使用可能な予備アイテムが存在しない場合，他のベースに存在する予備アイテムをこのベースに移送しアイテムを運用する．この方式は，アイテムをベース間で移送させるだけという少ない投資で，システム全体の稼働率の向上に大きく貢献する．また，各ベースでの故障アイテム数の大小というリスクを分散することができ，システ

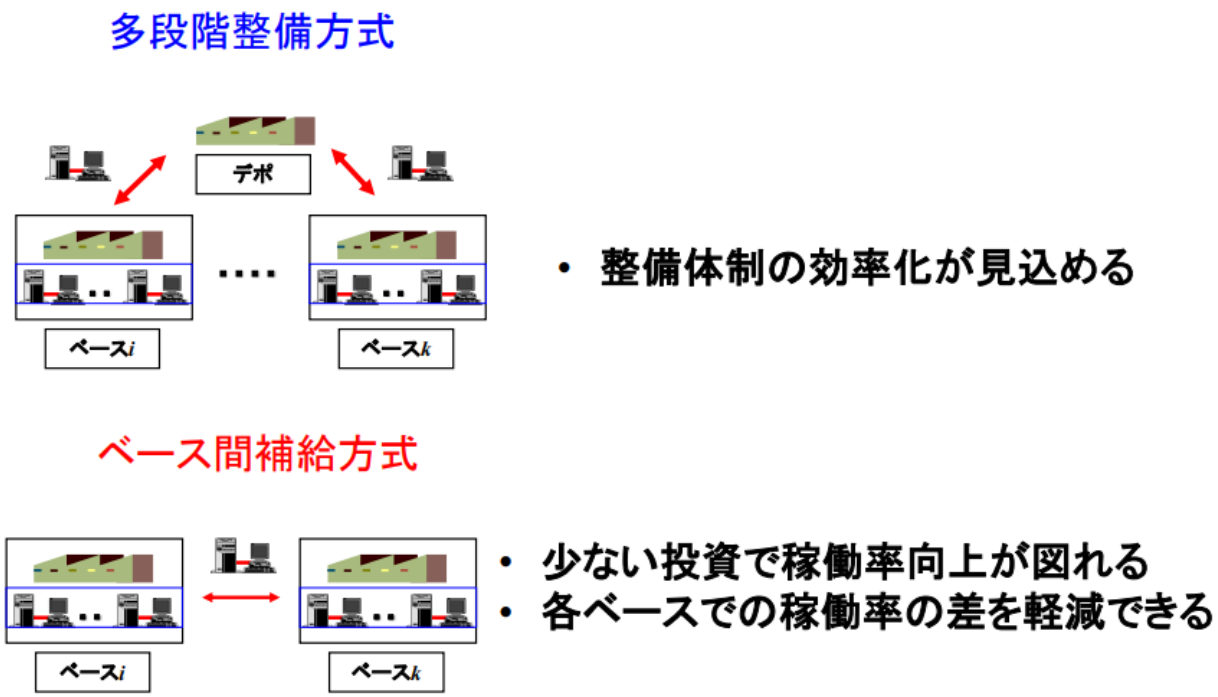

図 2.7　運用効率向上のための代表的な手法

ム全体の運用効率向上に大きく貢献する．

この両者のモデルについて，デポ及びベースに対して，修理施設の規模や予備アイテム数を調整することにより，稼働率の向上やコスト最適化を図ることが可能となる．

2.3.2　研究背景

前項のような補給整備システムを解析するための数理モデルの研究も盛んに行われている．

多段階整備方式における代表的な解析手法として Sherbrooke が提案した METRIC モデル [34] がある．このモデルでは，繰り越し発注量とコストを評価尺度として，繰り越し発注量を最小にすることにより，最適な予備アイテムの在庫数を決定している．以後，METRIC モデル関する研究やそれらを発展させたモデルの研究が行われた [35][36]．その中で，文献 [37] はデポ及び各ベースの同時修理可能アイテム数を有限としたモデルを提案し，大規模なモデルに対する解析を可能とした．しかし，一

定の故障発生率の下で保全性解析を行っているため，METRIC モデル同様に稼働率解析には適用できなかった．文献 [38] は文献 [37] のモデルを発展させ，各ベースにおける平均稼働アイテム数及び稼働率を求める近似解析手法を提案した．また，文献 [38] を発展させた研究として文献 [39] や [40] がある．

一方，ベース間補給方式に関するモデルの解析に関しても，現在までに数々の報告がなされている [41][42]．その中で，文献 [43][44] は対象モデルとして，民間航空機の補給整備システムを参考としたモデルの解析を行った．具体的には，複数ベースで同一アイテムが運用，修理，保管が行われ，各ベース間でのアイテムのやり取りを行うマルチベース補給システムに関して，文献 [45] や文献 [46] の解析方法を発展させた解析方法を提案した．また，現在までにこのモデルに基づく様々な研究が行われている [47]．しかし，文献 [43][44] の解析方法では，各ベースに存在するアイテム数が十分大と仮定していることや，故障アイテムの平均動作不可能時間が他の故障アイテム数によらず一定としていること，加えて，マルコフモデルによる解析を行っているため，ベース数，アイテム数の増加によって考慮すべき状態数が指数的に増大し，大規模モデルの解析は困難となることなど，いくつかの問題点が挙げられる．

2.3.3　解析手法概要

この項では，4 章でのマルチベース及びマルチクラウドの解析方法として用いた，単一アイテムでの 2 段階整備方式の定常状態での解析を行った文献 [37] の手法及び文献 [38] の解析手法を示す．

動作不可能アイテム数の分布決定法

文献 [37] は，２段階整備方式における各ベースで運用されるアイテムの故障発生率が一定，即ち故障発生がポアソン分布に従うと仮定した．この仮定により文献 [37] は，以下のような各ベースに所属する動作不可能アイテム数の分布決定法を提案した (図 2.8).

(1)　ベース修理施設にある故障アイテム数の分布
(2)　デポにおける故障アイテム数の分布
(3)　デポ・ベース間で移送中のアイテム数の分布
(4)　あるベースにおける動作不可能アイテム総数の分布

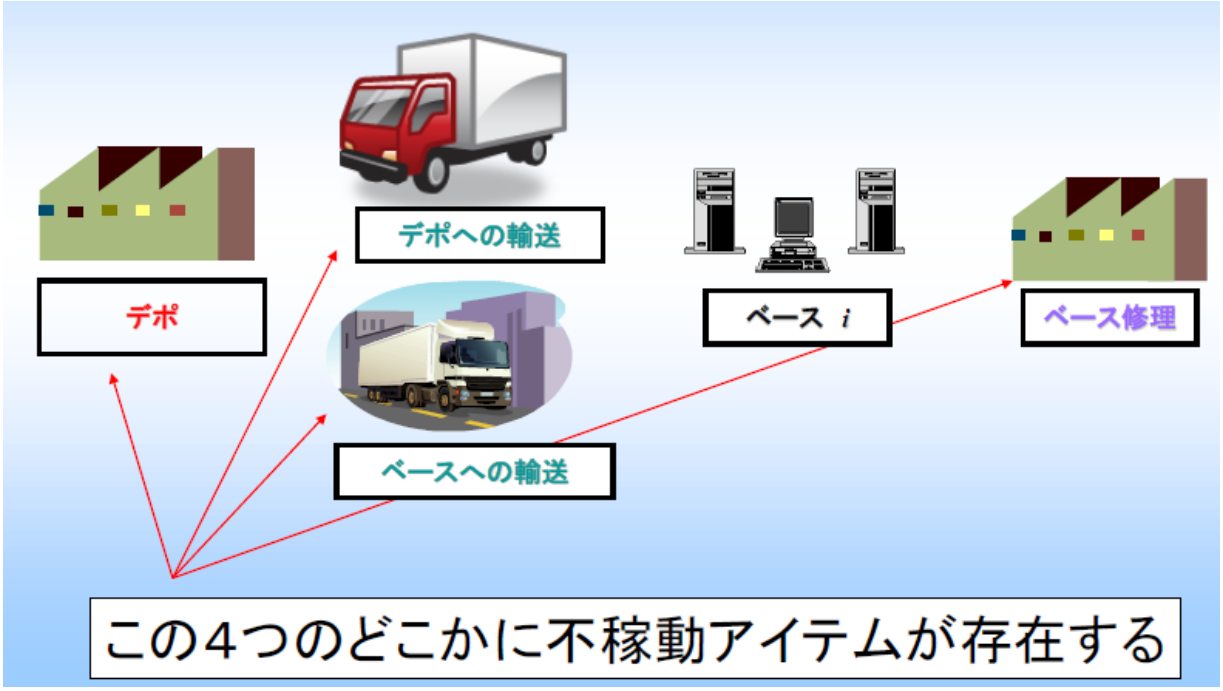

図 2.8　不稼動アイテムが存在する場所

これらを用いて各ベースでの平均故障数，予備品切れ確率等をもとめ最適予備アイテム数問題を解決している．

反復法による稼働率解析

文献 [38] は，大規模な 2 段階整備システムの稼働率解析を可能とする近似解析手法を提案した．この手法は，各ベースで運用される平均稼働アイテム数を未知パラメータとして導入し，故障発生率 (＝各ベースでのアイテムの故障率×平均稼アイテム数) を一定とおいた．そして，文献 [37] の動作不可能アイテム数の分布決定法を利用し，以下のような計算アルゴリズムを反復法により繰り返す．この反復計算により収束した各ベースの平均稼働アイテム数を解とし，2 段階整備方式の稼働率解析を可能とした．

各ベースにおいて

(0)　平均稼働アイテム数の初期値として適当な値を設定する

(1)　平均稼働アイテム数を用いて故障発生率を一定と考え，動作不可能アイテム

数の分布決定手法を利用し，各ベース所属の動作不可能アイテムの数の分布を計算する

(2)　(1) を用いてベースの平均稼働アイテム数の再計算を行う

(3)　(1) で使用した平均稼働アイテム数と (2) で再計算された平均稼働アイテム数を比較する

(4)　(3) の差が小さくなるまで (2) の平均稼働アイテム数を新たな平均稼働アイテム数とし，(1)〜(3) を繰り返す

2.3.4　モンテカルロシミュレーション

4 章で用いたモンテカルロ法について説明する．モンテカルロ法とは大数の法則に基づいて，乱数を用い何度も試行を繰り返し，その結果得られる測定値の平均で真の値を推定しようとする方法のことである．一般に，モンテカルロ法においては，試行回数による統計的誤差が問題となる．この誤差は標準偏差で評価され，試行回数を n とすると標準偏差に比例する．したがって，1 桁精度を上げるためには，試行回数 $1/\sqrt{n}$ を 100 倍にしなければならない．

第 3 章

拡散型フロー制御を用いる DDoS 攻撃緩和方式の有効性評価

3.1 研究背景

サイバー空間に脅威を与えるサイバー攻撃の一つ，DDoS（Distributed Denial of Service）攻撃は，攻撃トラフィックサイズが増幅する傾向にあり，その規模は数テラ bps に到達している．例えば，Amazon Web Services は 2020 年 2 月に 2.3Tbps の DDoS 攻撃を受けたことを報告している [1]．また，コロナ禍によるオンラインサービスの需要の増加により，標的が増えたことから DDoS 攻撃の件数が 2020 年第 1 四半期は前年同期比で 25% 増加という報告も出ている [2] ．さらに，国内においても，攻撃者が標的とする組織宛に DDoS 攻撃を示唆するメールを送り，仮想通貨による送金を要求する事案が発生しており，この攻撃では，攻撃能力を示すためとして，数十 Gbps から 100Gbps 程度の規模で標的に対して DDoS 攻撃を行ったことが報告されている [3] ．DDoS 攻撃は，その頻度や影響度も大きくなっており，引き続き，DDoS 攻撃の脅威に対する対策が必要である．

DDoS 攻撃の緩和のために，事前に防御すべき（標的の可能性となる）サーバやネットワークの定常的なトラフィックパターンを調査し，それに基づいて事前に緩和対応策を実施する事前対応型と攻撃を確認した後に緩和対応をする事後対応型がある．事前対応型には Akamai 社のサービスとして提供されているものもある．事後対応型は一定時間トラフィックの観察や，緩和できるまでの時間がかかるものの，事前対応型のような事前準備は不要である．事後対応型では，攻撃の検知，攻撃元から

標的までの複数の経路で攻撃トラフィックを排除や分散するなどして，標的への攻撃が及ぶまでの時間を延ばすことが重要となる．そのために，ルーティングアルゴリズム，ブロックチェーン，機械学習などを適用した様々な緩和手法が提案されている [4] [5] [6][7] [8] [9][10].

文献 [11] [12] [13] において，物理現象である拡散現象を指導原理とした自律分散制御を用いて DDoS 攻撃トラフィックの集中を緩和する，事後対応型の手法が提案されている（以下，既存方式と呼ぶ）．既存方式 [13] では，標的までの経路上に大量の攻撃トラフィックを緩和するための装置をおき，標的そのものへのトラフィックを緩和する方式である．この方式による緩和により，標的が攻撃の被害を受ける（サービス妨害が発生する）まで時間に猶予をもたせ，その間に攻撃元の特定とフィルタリングなどの対策を講じることを想定している．そのため，緩和可能な時間（以降，緩和時間）が長い方がよいが，既存方式 [13] では緩和装置におけるトラフィックの分散パターンやバッファ容量が固定した限定的な制御をしていることから，緩和すべき対象のネットワークのトポロジの構成によっては，経路上での緩和装置のバッファの溢れが早期に起こる．すなわち緩和時間が極端に短くなる．この限定的な制御は，実際の緩和装置や動的なトラフィック変化に即しておらず，より柔軟な制御が必要である．

そこで，本章では偏りのあるトポロジ（例えば，完全 2 分木のような子ノード数や葉ノードの深さが全て揃っているものではなく，子ノード数や葉ノードの深さに違いがあるようなもの）やバッファ容量のばらつきがある場合にも，より長い緩和時間を確保することを目的として，既存方式 [13] を改良する．具体的には，バッファリングノードで形成するオーバーレイネットワークのトポロジのタイプやバッファ容量等の偏りのあるトポロジやバッファ容量のばらつきなど，ネットワーク状態が不均衡である場合にも，より安定した性能を確保することを目的として，ノード間のパケットの転送を制御している転送レート算出式を改良することで，より長時間パケット損失を防ぐことを可能とする．また，既存方式 [13] と本章で提案する方式について，シミュレーションを実施し，その結果を比較することで提案方式の有効性を評価する．本章の構成は，以下の通りである．3.2 節にて関連研究について，3.3 節にて提案方式について述べる．3.4 節にて改良方式を用いた DDoS 攻撃緩和システムについて，3.5 節でシミュレーションの評価について説明を行う．3.6 節にてシミュレーション結果より考察を行い，最後に 3.7 で本章のまとめと今後の課題について述べる．

3.2 関連研究

3.2.1 DDoS 攻撃の検知・対策

最近の DDoS 攻撃緩和（ミチゲーション）に関する研究は，以下のようなものがある．文献 [8] では，攻撃を受けている疑いのある宛先への通信について，DDoS ミチゲーション装置を経由させることで，不正なトラフィックを検知し排除することを目的とし，ルーティングにより，「ミチゲーション装置」への経由をより効率的に実施するアルゴリズムが提案されている．また，文献 [9] では，分散された緩和装置間でブロックチェーンを用いたスマートコントラクトによるブラックリストやホワイトリストの制御情報を自動化し，共有することで攻撃を検知する手法が提案されている．さらに，文献 [10] で提案されている手法は，トラフィックの監視を行い攻撃検知するもので，その検知方法に過去のデータから，決定木アルゴリズム等の機械学習を用いて学習を行うことであり，誤検知を減らすとともに，未知の攻撃の検知も試みるものである．

これらの研究は，主に攻撃の検知を効果的に行うことに焦点をあてて，標的への攻撃を回避する提案で，本章や本章の先行研究である文献 [13] のように DDoS 攻撃の緩和により検知から対策までの緩和時間を確保して，標的の攻撃を回避するというアプローチとは異なる．

文献 [13] では，攻撃対象の拡散型フロー制御 [31] を用いて，隣接するノードとの相互作用のみで自律分散的に転送レートの制御を行い，DDoS 攻撃検知後から攻撃対象となる標的サーバ側に近い下流でのネットワーク負荷を回避しながら，正規パケットの損失を防ぐことを可能とする手法が提案されている．

3.2.2 拡散型フロー制御の DDoS 攻撃緩和への適応

拡散型フロー制御 [31] とは，物理学における拡散現象を指導原理とし，ネットワークの輻輳回避を目的とした自律分散型フロー制御である．エンドホスト間で経由されるネットワーク機器（以下，ノード）が隣接するノードとの相互作用のみで自律分散的に転送レートの制御を行い，転送パケットのネットワーク機器間での平滑化を実現する (2.1.1 項参照).

既存方式 [13] では，DDoS 攻撃の標的となるサーバに向かう DDoS 攻撃トラフィッ

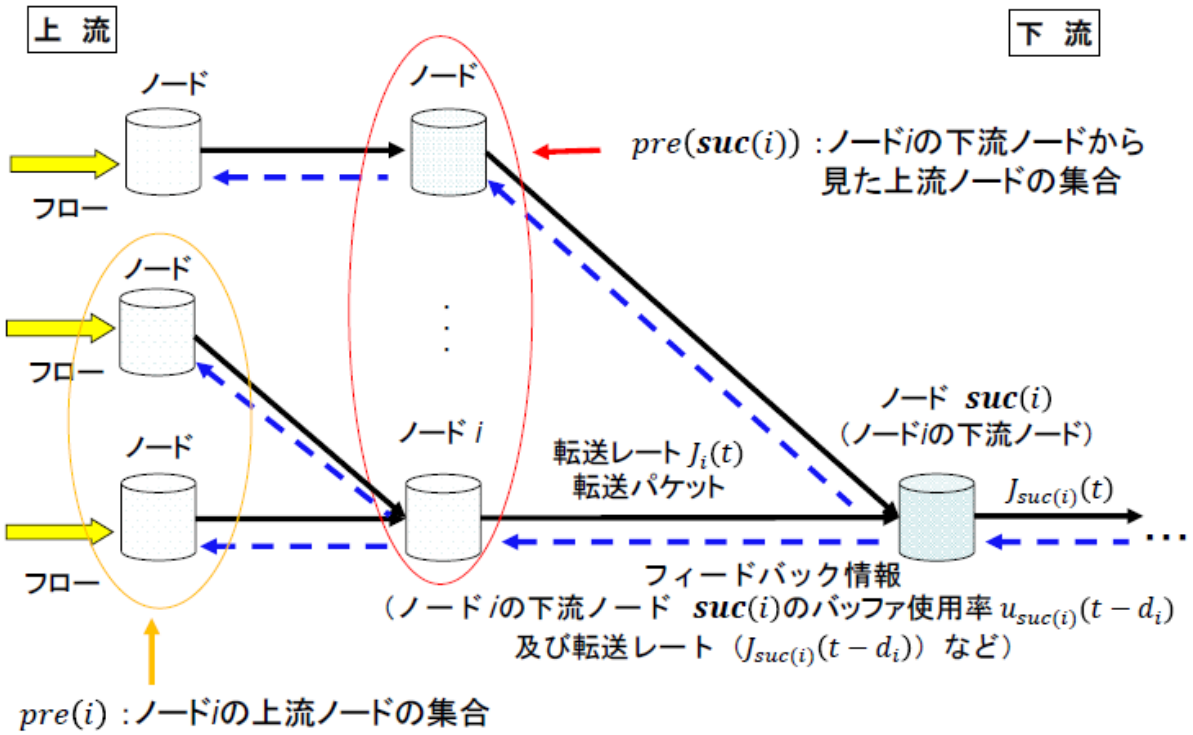

図 3.1　ノードの動作モデル

クの経路上のノード（ルータ等）が，攻撃トラフィックの転送レートを制御するための大きなバッファをもつノード（緩和装置）となることを前提とする．転送レートを制御してバッファ使用率を平滑化することで，標的サーバに向かう DDoS 攻撃トラフィックの緩和をする手法である．各ノードのバッファ容量の平滑化に前述の拡散型フロー制御 [31] による自律分散的な転送レート制御を用いる．

ノードの動作モデルを図 3.1 に示す．図 3.1 では，ネットワークに存在する n 個のノードの識別子を $i(=0,1,2,\cdots,n-1)$ とし，ノード i の下流ノードを $suc(i)$, 上流ノードの集合を $pre(i)$ と定義している．このモデルでは DDoS 攻撃の攻撃トラフィックは上流ノードから流入し，最下流ノードの先にある標的サーバに向かっていくことを表す．攻撃元は複数あることから拡散型フロー制御 [31] の動作モデルで想定されているノードが直列につながったトポロジ (図 3.2) ではなく，木構造のトポロジを採用している既存方式 [13] では，各ノードで構成されるオーバーレイネットワークとなるため木構造のトポロジ (図 3.1) である．このトポロジ上において，上流ノード向

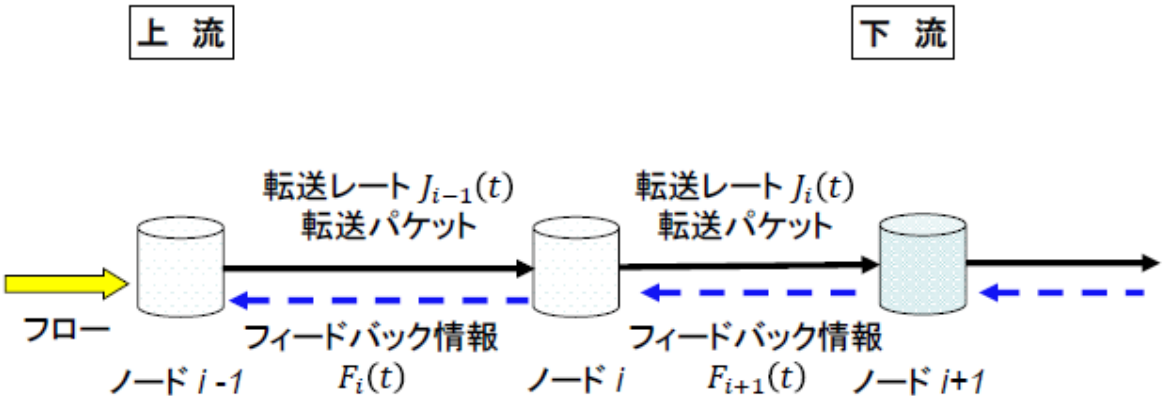

図 3.2　拡散型フロー制御におけるノードの動作モデル

きにフィードバック情報（自ノードのバッファ使用率とパケットの転送レート）が送信される．また，フィードバック情報は，各ノードで一定時間毎（フィードバック情報送信間隔）に送信しており，かつ各ノード間の伝搬遅延の時間に連動してフィードバック情報を同期している．

各ノードは，下流からの全フィードバック情報をもとに転送レートを計算して，それに応じて上流ノードからのパケットを自ノードでバッファリングするとともに必要量を下流へ転送する．ノード i に流れるフローの時刻 t の転送レート $J_i(t)$ は式 (3.1) のように算出する．

$$J_i(t) = D_i' J_{suc(i)}(t - d_i) - D_i S_i(u_{suc(i)}(t - d_i) - u_i(t)) \tag{3.1}$$

ただし，$0 \leq J_i(t) \leq L(i)$ とする．

$$D_i' = \frac{1}{|pre(suc(i))|} \tag{3.2}$$

$$D_i = \frac{1}{|pre(suc(i)) + 1|} \tag{3.3}$$

ただし

$suc(k)$：ノード k の下流ノード

$pre(k)$：ノード k の上流ノードの集合

($|pre(k)|$ は，$pre(k)$ の要素数を示す)

ここで，式 (3.1) の右辺第一項をドリフト項，第二項を拡散項とよぶ．d_i はノード i とノード $i+1$ の間の伝送遅延時間（ただし，上り下りとも同じ値），$D_i^{'}$ が分散係数，D_i が拡散係数，S_i はスケール係数である．$J_i(t)$ は，下流ノードから送られてきた転送レート $J_{suc(i)}(t-d_i)$，バッファ使用率 $u_i(t)$，下流ノードから送られてきたバッファ使用率 $u_{suc(i)}(t-d_i)$，各ノード間の使用可能帯域 $L(i)$ より算出される．ただし，スケール係数 S_i はノード i のバッファ容量とする．

3.3 提案方式

既存方式 [13] では，分散係数 (式 3.2) の算出式のように，各ノードの下流向け転送レートの決定時に上流ノードの数で均等割を行なっている．しかし，実際の DDoS 攻撃緩和システムで動作するノードの接続構成も様々なため上流ノード向けのリンク数も様々である．また，各ノードのバッファ容量も均一でない場合もある．これらが考慮されてないため，特定ノードでのバッファの溢れが早期に起るという問題点がある．以降でこの問題を詳述し，転送レートの算出式の見直しについて述べる．

3.3.1 容量のばらつきを考慮した拡散項の改良

既存方式 [13] では，バッファ使用率を平滑化することを目的として，式 (3.1) を定義している．しかし，各ノードのバッファ容量が異なるとき，式 (3.1) はバッファ使用率の平滑化を行うため，各ノードのバッファ容量の違いによる残容量の平滑化が達成されない場合がある．例えば，自身のノードとその下流のノードのバッファ容量がそれぞれ 10GB と 20GB で，どちらもバッファ使用率が 0.5 の場合，それぞれのバッファ使用量は，5GB と 10GB である．この場合，既存方式 [13] では自身のノードの式 (3.1) の拡散項のスケール係数 ×（バッファ使用率の差）は 10GB× (0.5 – 0.5) = 0GB となり，実際のバッファ使用量のノード間の差 20GB× 0.5 − $10GB \times 0.5$ =5GB を示せない．これにより，本来の目的である，各ノードの偏りを解消することができない問題が生じる (図 3.3).

このことを解決するため，既存方式 [13] で使用しているバッファ使用率 $u_i(t)$ を，ノードの利用可能な残りのバッファ容量をスケールとするバッファ残容量 $\tilde{U}_i(t)$ に変更する．これにより，各ノードのバッファ容量に対する残りの容量を平滑化するように，拡散現象が起こるため，より効率的にバッファを使用できる．つまり，各ノードのバッファ容量の違いによる既存方式の問題点を解決する．

改良方式として，各ノードのバッファ容量のばらつきを考慮した転送レート算出を式 (3.1) の拡散項を式 (3.4) のとおり変更する．

$$J_i(t) = D_i^{'} J_{suc(i)}(t - d_i) - D_i(\tilde{U}_{suc(i)}(t - d_i) - \tilde{U}_i(t)) \tag{3.4}$$

なお，各ノードのバッファ容量が同一の場合，改良方式と既存方式 [13] は同値と

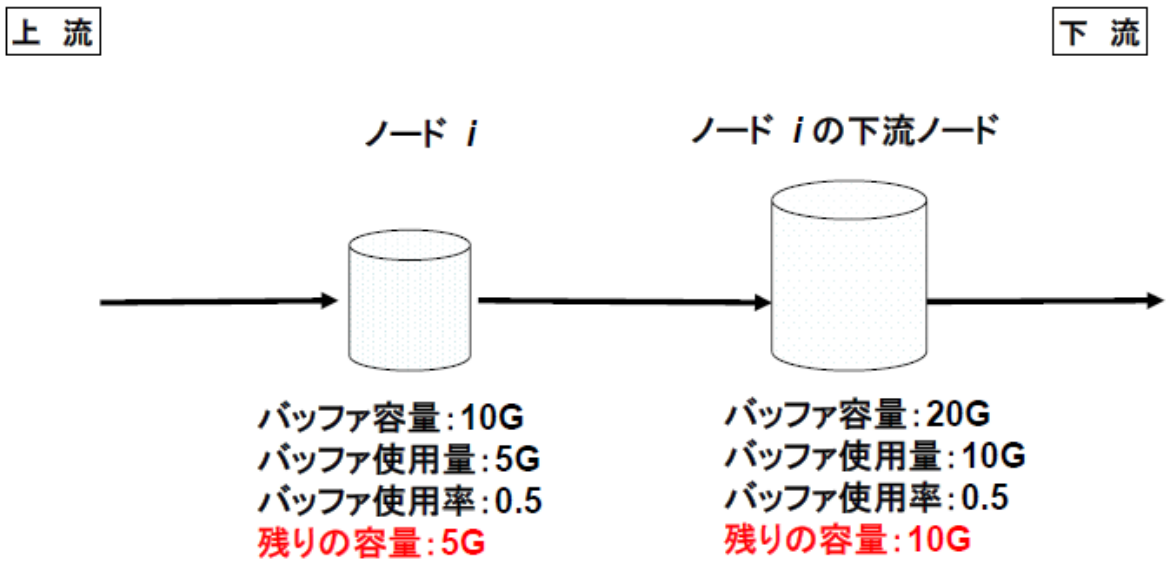

図 3.3　拡散項の修正

なる．

3.3.2　トポロジ分岐を考慮したドリフト項の改良

本節では，式 (3.1) で定義されているドリフト項にかかる分散係数の修正を行う．

分散係数 D_i' は，下流ノードから上流ノードに送られてくる転送レートを分配するための係数である．既存方式 [13] では，DDoS 攻撃の送信元が図 3.1 のように木構造になっており，下流ノードに対して上流ノードは 1 対多となる場合がある．そのため，既存方式 [13] では，下流ノードに対する上流ノードの数を均等割し，下流ノードから送られてくる転送レートを 式 (3.2) の分散係数で分配する．

しかし，ノードによって各上流ノードからのパケットの流入量が異なるので，既存方式 [13] のように均等割した場合，パケットの流出量の多い上流ノードでは，パケットを下流ノードに転送できず早期にバッファ溢れが発生することが予想される．そこで，分散係数 D_i' の定義である式 (3.2) について，時点 t での上流ノードからの下流ノードに流入するパケットの流入量を各上流ノードの流入量で分配して重み付けする手法に変更することで前述の問題の発生の改良をする．この改良により，各上流ノードからのパケットの流入量の違いを吸収する (図 3.4).

具体的には，式 (3.2) を式 (3.5) に変更する．

$$D_i^{'} = \begin{cases} \dfrac{J_i(t-2d_i)}{\sum_{k \in pre(suc(i))} J_k(t-2d_i)} & \text{(分母} > 0 \text{ の場合)} \\ 0 & \text{(分母} = 0 \text{ の場合)} \end{cases} \tag{3.5}$$

式 (3.5) では，流入量を単位時間当たりの流入量，つまり，転送レート $J_i(t)$ としている．また，$t-2d_i$ となっているのは，ノード間で遅延が往復分となるためである．

このドリフト項の改良に伴いフィードバック情報に新たに定義した分散係数 $D_i^{'}$ を追加する．これにより，上流ノードに対して，下流ノードから新たな分散係数が定期的に伝えられ，順次上流ノードの転送レート算出式のドリフト項が更新される．これにより各上流ノードのパケットの流入量の変化に対して柔軟に対応できる．

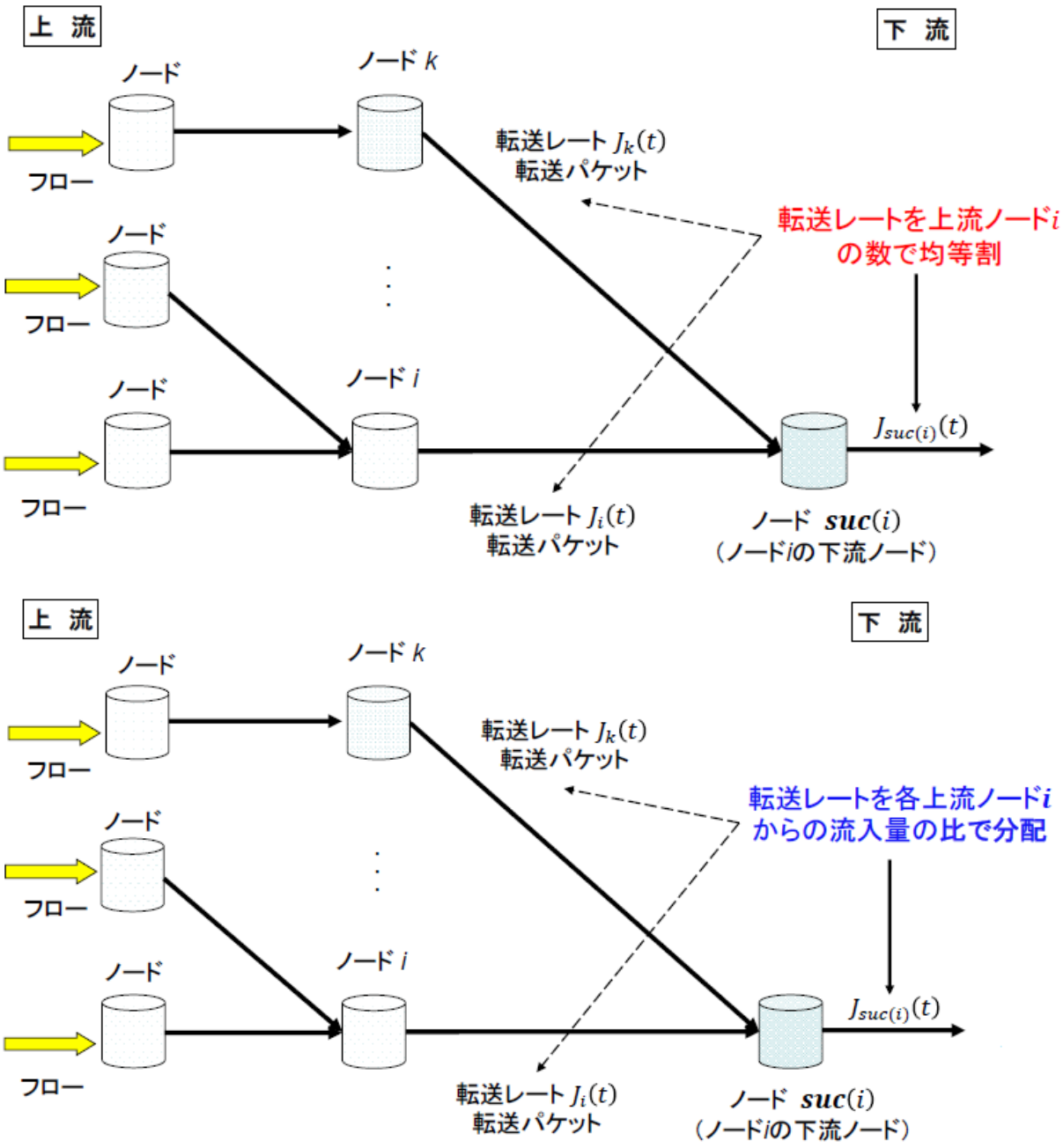

図 3.4　ドリフト項の修正

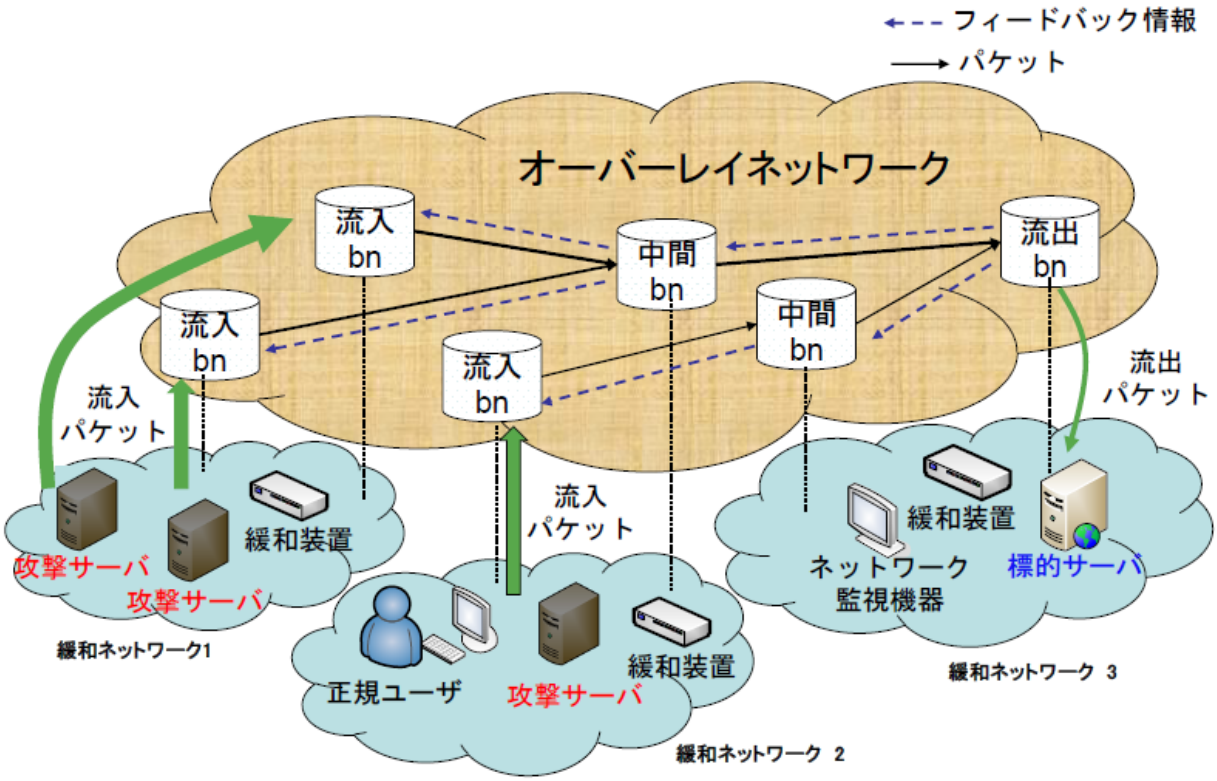

図 3.5　システムの全体構成

3.4　改良方式による DDoS 攻撃緩和システム

3.3 節で述べた改良方式を用いた DDoS 攻撃緩和システムの構成は文献 [13] と同様に図 3.5 になる．DDoS 攻撃緩和の対象となるネットワーク（以降，緩和ネットワーク）内のノードは，パケットが蓄積できるバッファリングノード（以下，bn）であると伴に，ミチゲーションを行う．システムとしては，「常時ネットワークを監視し DDoS 攻撃を検知するネットワーク監視機器」，「bn と連携してトラフィックを学習しフィルタリングルールを生成する緩和装置」から構成される．各緩和ネットワークの bn 同士はオーバーレイネットワークを構成し，このノードが図 3.1 のノードに対応する．bn のうち送信元からのパケットを受信するノードを流入 bn，最下流に位置する標的サーバにパケットを転送するノードを流出 bn，それ以外のノードを中間 bn と呼ぶ．ネットワーク監視機器が異常トラフィックを検知することで DDoS 攻撃

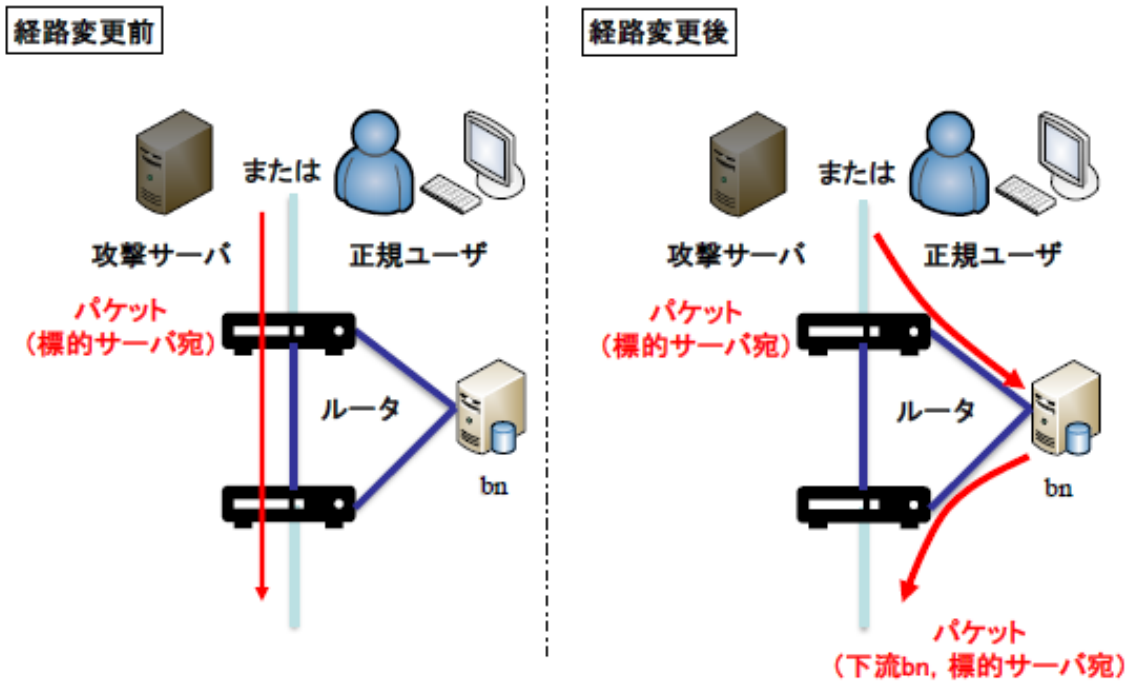

図 3.6　経路変更前後のパケットの流れ

を検知し，流出 bn にミチゲーションを要求する．ミチゲーションが始まると，隣接 bn 間が連携して転送レートを制御し，下流にパケットを転送やバッファリングを行う (図 3.6)．動作の詳細は以下となる．

1. 各 bn は，自ノードのバッファ残容量，転送レート及び分散係数をフィードバック情報として上流ノードに送信する．
2. 下流ノードから送られてきたフィードバック情報と自ノードのバッファ残容量を元にパケットの転送レートを計算する．
3. 下流ノードへ自ノードの転送レートに従って宛先宛パケットの転送を行うと伴に，上流ノードから送られてきた宛先宛のパケットを受け取りバッファリングする．
4. ミチゲーション終了の通知が届くまでは，新たなフィードバック情報を受け取った場合は，上記を再度 1〜3 を実施する．

上記の動作と同時に，攻撃パケットの解析を行い，フィルタリングルールを生成

し，緩和装置が生成したフィルタリングルールを用いて bn が攻撃パケットを遮断する（バッファにある攻撃パケットは廃棄する）.

表 3.1　シミュレーションのパラメータ 1

パラメータの種類	変数値			
bn 数	60			
流入 bn 数	20			
シナリオ	(a)	(b)	(c)	(d)
転送レートパターン	固定	固定	ランダム	ランダム
バッファ容量パターン	固定	ランダム	固定	ランダム
流出 bn 送信レート [pps]	10×10^4			
流入トラフィック送信レート [pps]	1.0×10^5		$0.5 \sim 1.5 \times 10^5$	
bn のバッファ容量 [packet]	2.0×10^6	$1.0 \sim 3.0 \times 10^6$	2.0×10^6	$1.0 \sim 3.0 \times 10^6$
bn 間使用可能帯域 [pps]	10×10^6			
bn 間の伝搬遅延時間 [ms]	10			
フィードバック情報送信間隔 [ms]	10			

3.5　シミュレーションによる評価

3.5.1　提案方式の評価方法について

改良方式を評価するため，標的サーバ宛の DDoS 攻撃を開始した後，各方式で用いる転送レートの算出式を用いて緩和をするシミュレーションを行う．シミュレータは文献 [13] のシミュレータを一部変更して用いた．具体的には，3.4 節の DDoS 攻撃緩和システムをシミュレータで再現し，攻撃パケットの発生，ノード間のパケットの転送などの各事象に関しては，一定の時間で状態の変化がおきる時間駆動型とし，各事象の発生によってシステムの状態を変化させる離散事象シミュレーションで数値計算を行った．また，シミュレーションで使用するパラメータや bn で構成されるオーバーレイネットワークのトポロジも文献 [13] で使用したものを用いた．パラメータの一覧を表 3.1 に示す．

bn 数を 60 とした理由は，以下のような想定による．bn 数は，提案システムの適用するオーバーレイネットワークを構成するノード数であり，言い換えると連携して提案システムを用いる緩和ネットワーク数に相当する．本章では，連携して提案システムを用いる集まりを IX (Internet Exchange Point) に接続するもの（ISP 事業者，

CDN 事業者，クラウド事業者，データセンタ事業者など）と想定し，実際に存在する IX にどれくらいの接続があるかを調べた．2021 年 10 月現在， 世界中に存在する IX への接続情報を共有するデータベースである Peering DB[48] には，約 900 の IX が登録されており，国内の登録数は 14 である．また，国内の各 IX に接続されているネットワーク数は，1〜230 程度である．例えば，IX に接続されたネットワークのうち DDoS 攻撃緩和システムに参加するネットワーク数を 4 分の 1 程度と仮定すると最大で 60 程度の緩和ネットワーク数が見込まれる (図 3.7)．これにより bn 数を 60 とした．また，各 bn で構成されるオーバーレイネットワークのトポロジの木の深さについては 7 から 10 としている．この深さについては，既存方式 [13] と差異を比較するため同じ深さとした．

他のパラメータは，

1. 緩和ネットワークよりパケットが流れ込む bn の数である流入 bn 数
2. オーバーレイネットワークより標的サーバへの流れ込む流出パケットである流出 bn 送信レート
3. 標的サーバ宛に緩和するネットワークよりオーバーレイネットワークに流れ込む流入パケット（正規及び攻撃）の流入トラフィック送信レート
4. 各 bn のバッファ容量
5. 各ノード間の使用可能帯である bn 間使用可能帯域
6. ノード間の伝送遅延時間である bn 間の伝搬遅延時間
7. ノード間のフィードバック情報の送信間隔となるフィードバック情報送信間隔

である．

評価基準としては，文献 [13] の評価基準でもある，トポロジを構成するいずれかの bn のバッファからパケットがあふれ出す最初の時刻（パケット損失発生時刻）までの時間とする．これを緩和時間（mitigation time）と呼ぶ．

3.5.2 実験方法

前述の評価方法に基づき，以下の 4 方式に対して，トポロジや各種パラメータが緩和時間に与える影響を調べるシミュレーション実験を実施した．

- 既存) ：既存方式
- 改良 1)：拡散項改良方式（3.3.1）
- 改良 2)：ドリフト項改良方式 (3.3.2）
- 改良 3)：拡散項・ドリフト項改良方式（3.3.1，3.3.2）

3.5.3　既存方式との比較に関する実験

シミュレーション実験では表 3.1 のシナリオ (a) から (d) のように各パラメータの値を設定し，トラフィック送信レートとバッファ容量がそれぞれ固定とランダムの 4 つの組み合わせで，50 パターンのトポロジ (図 3.8) を用いてシミュレーションを実施した．また，以下に変数値のシナリオを記述する．

(a)　バッファ容量及び宛先トラフィック一定
(b)　バッファ容量ランダム・宛先トラフィック一定
(c)　バッファ容量一定・宛先トラフィックランダム
(d)　バッファ容量及び宛先トラフィックランダム

ただし，各トポロジで使うランダムな値について，同じトポロジでは同一とした．

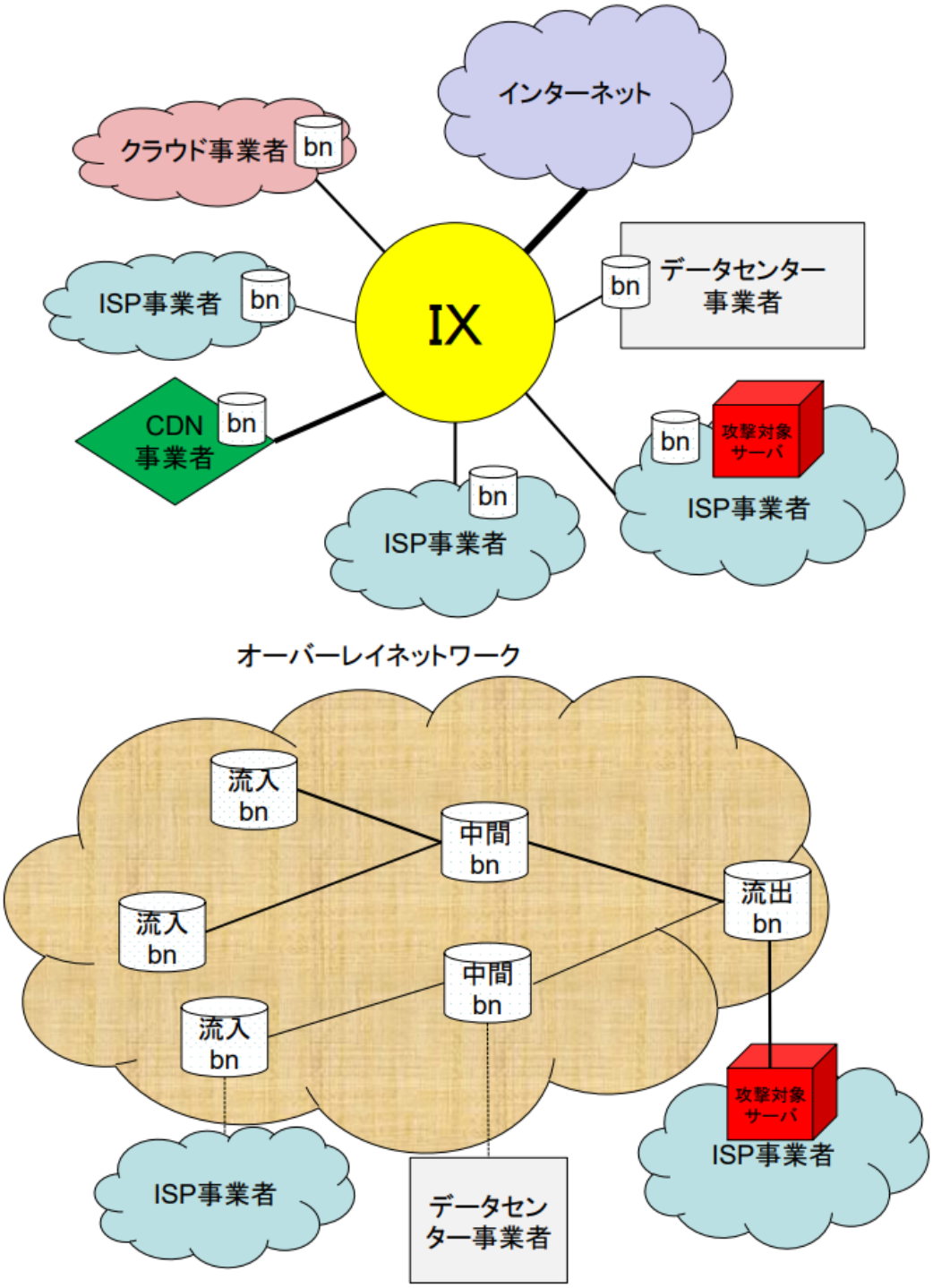
インターネット
クラウド事業者
bn
IX
ISP事業者
bn
データセンター
事業者
bn
CDN
事業者
bn
攻撃対象
サーバ
bn
ISP事業者
bn
ISP事業者
オーバーレイネットワーク
流入
bn
中間
bn
流出
bn
流入
bn
中間
bn
流入
bn
攻撃対象
サーバ
ISP事業者
ISP事業者
データセン
ター事業者

図 3.7　IX の構成

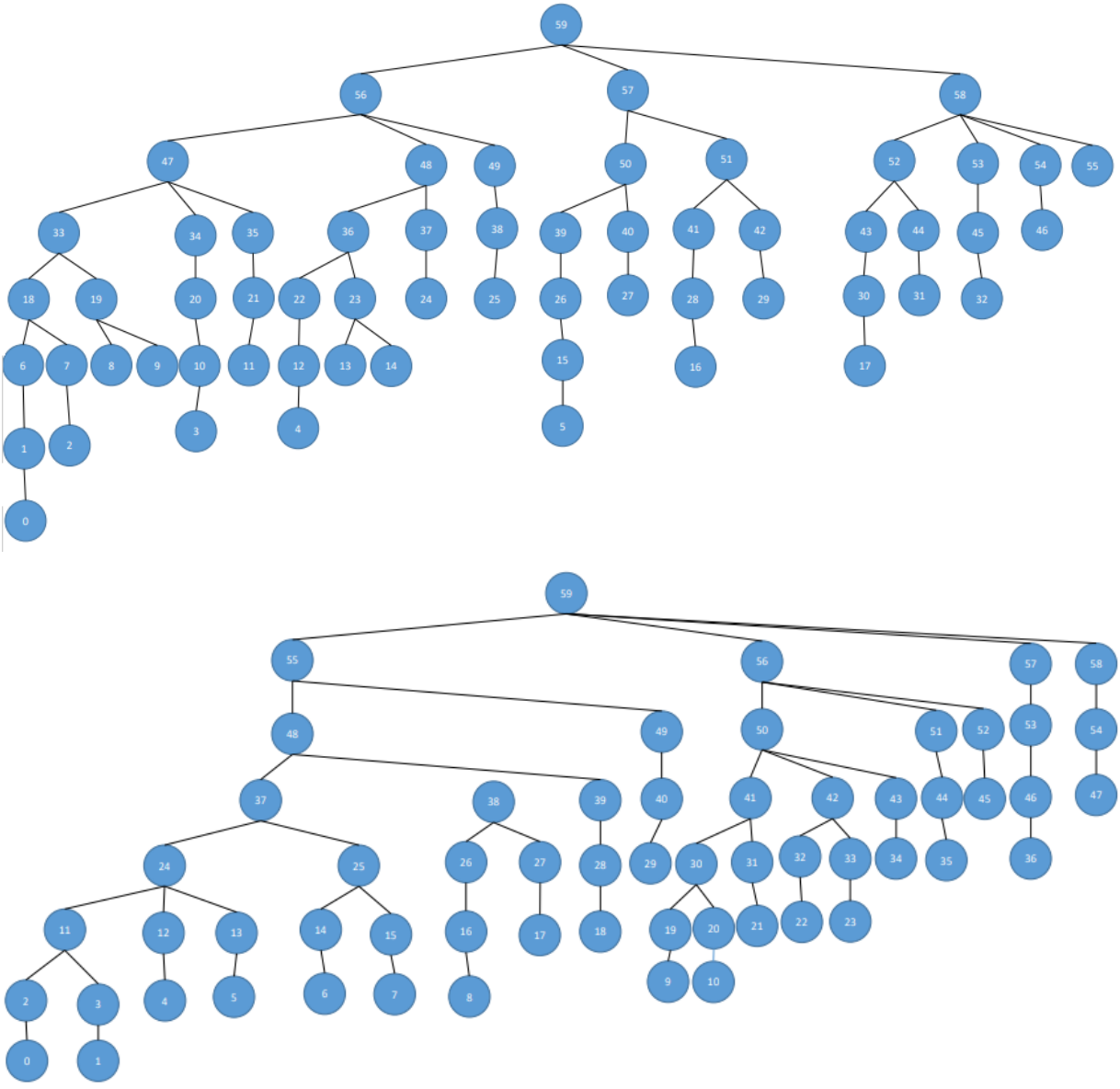
59
56
57
58
47
48
49
50
51
52
53
54
55
33
34
35
36
37
38
39
40
41
42
43
44
45
46
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
6
7
8
9
10
11
12
13
14
15
16
17
1
2
3
4
5
0
59
55
56
57
58
48
49
50
51
52
53
54
37
38
39
40
41
42
43
44
45
46
47
24
25
26
27
28
29
30
31
32
33
34
35
36
11
12
13
14
15
16
17
18
19
20
21
22
23
2
3
4
5
6
7
8
9
10
0
1

図 3.8　トポロジの例

表 3.2 シミュレーションのパラメータ 2

パラメータの種類	変数値
bn 数	60
流入 bn 数	20
シナリオ	(e)
転送レートパターン	ランダム
バッファ容量パターン	ランダム
流出 bn 送信レート [pps]	1.0×10^6
流入トラフィック送信レート [pps]	$0.5 \sim 1.5 \times 10^7$
bn のバッファ容量 [packet]	$1.0 \sim 3.0 \times 10^8$
bn 間使用可能帯域 [pps]	1.0×10^8
bn 間の伝搬遅延時間 [ms]	10
フィードバック情報送信間隔 [MS]	10

3.5.4 攻撃規模の増大の影響に関する実験

DDoS 攻撃の規模は年々増加傾向にあり，例えば，Amazon Web Services は 2020 年 2 月に 2.3Tbps の DDoS 攻撃を受けたことが報告されている [1]．そこで，攻撃規模の増大による本システムへの影響について考察する．表 3.2 の攻撃トラフィックに相当する流入トラフィック送信レートが増加した場合についての実験を実施する．具体的には，表 3.1 のシナリオ (d) における流入トラフィック送信レート，流出 bn 送信レート及び，bn のバッファ容量を 100 倍としたシナリオ (e) を作成し，攻撃規模を増大させた変数について，改良 3) で 5 パターンのトポロジを用いて，各トポロジでの緩和時間及び緩和時間を超える直前の全ノードのバッファ容量に対してどれくらいバッファリングしているかを示すバッファ使用率を確認する．

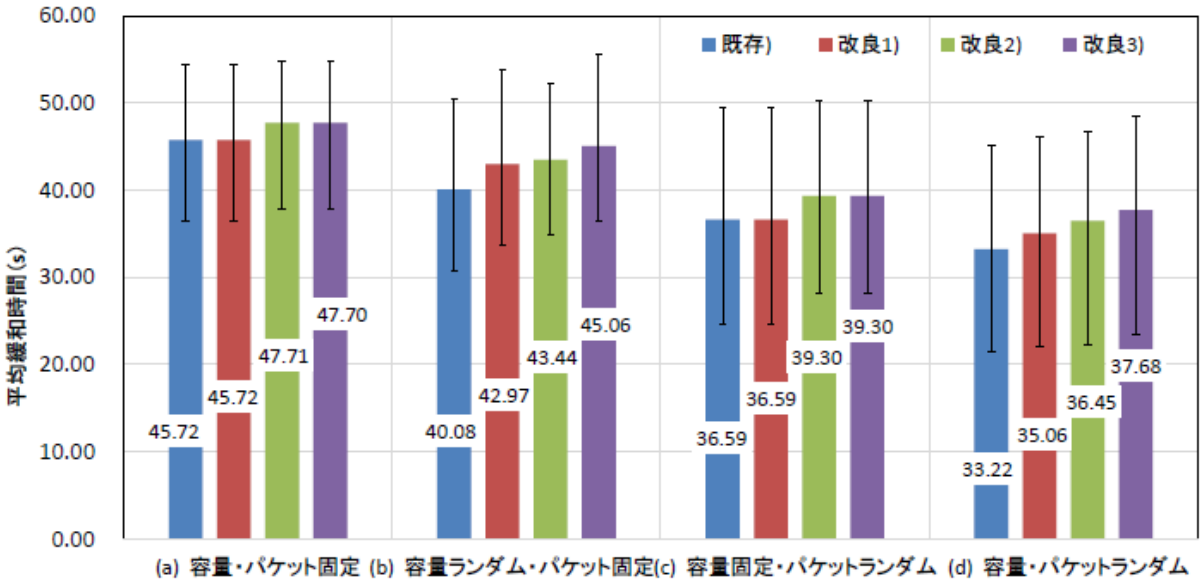

図 3.9　平均の緩和時間の変化

3.6　結果と考察

3.6.1　改良方式の有効性

3.5.3 の実験における 50 パターンのトポロジの結果の平均緩和時間（s）を図 3.9 に示す．また，図 3.10 にシナリオ (d) 転送レートとバッファ容量がどちらもランダムに変わる場合，トポロジによって既存方式 [13] に比べて，3 つの改良方式の緩和時間がどう変化するかを示す．縦軸は既存方式 [13] の緩和時間を 1 としたときの改良方式の緩和時間の比率を示す．横軸の 1〜40 の番号に相当するトポロジは，50 パターンのトポロジのうちから，既存方式 [13] と改良 3) の緩和時間の差が最小となるもの（番号 1〜20），緩和時間の差が最大のもの（番号 21〜40）である．

図 3.9 のとおり，平均緩和時間は，既存) と比較した場合，全てのシナリオで改良 3) が上回る結果となった．また，パラメータやトポロジのタイプによっては，図 3.10 の番号 40 のように，約 1.7 倍の緩和時間を延ばすことができている．加えて，最大と最小の緩和時間も (a) から (d) の全てのシナリオにおいて改良 3) が既存) を上回る．シナリオ (a) と (c) のようなバッファ容量一定の場合には，既存) と改良 1) または改

表 3.3 各トポロジでの緩和時間

シナリオ	(a) バッファ容量一定・攻撃パケット一定			
方式	既存)	改良 1)	改良 2)	改良 3)
平均バッファあふれ時間 (s)	45.72	45.72	47.71	47.70
最小バッファあふれ時間 (s)	36.47	36.47	37.74	37.74
最大バッファあふれ時間 (s)	54.31	54.31	54.75	54.75
既存) との比較 (平均)	−	1.00	1.04	1.04
既存) との比較 (最小)	−	1.00	1.00	1.00
既存) との比較 (最大)	−	1.00	1.13	1.13

シナリオ	(c) バッファ容量一定・攻撃パケットランダム			
方式	既存)	改良 1)	改良 2)	改良 3)
平均バッファあふれ時間 (s)	36.59	36.59	39.30	39.30
最小バッファあふれ時間 (s)	24.92	24.92	28.21	28.21
最大バッファあふれ時間 (s)	49.53	49.53	50.23	50.23
既存) との比較 (平均)	−	1.00	1.08	1.08
既存) との比較 (最小)	−	1.00	1.00	1.00
既存) との比較 (最大)	−	1.00	1.28	1.28

良 2) と改良 3) のような組み合わせで，各トポロジでの緩和時間が概ね一致した．これは，バッファ容量が一定の場合，3.3.1 項の提案方式と既存方式 [13] は，理論的に同じ条件（同じ結果となる）となるためであり，実験と理論が一致した結果を示している (表 3.3).

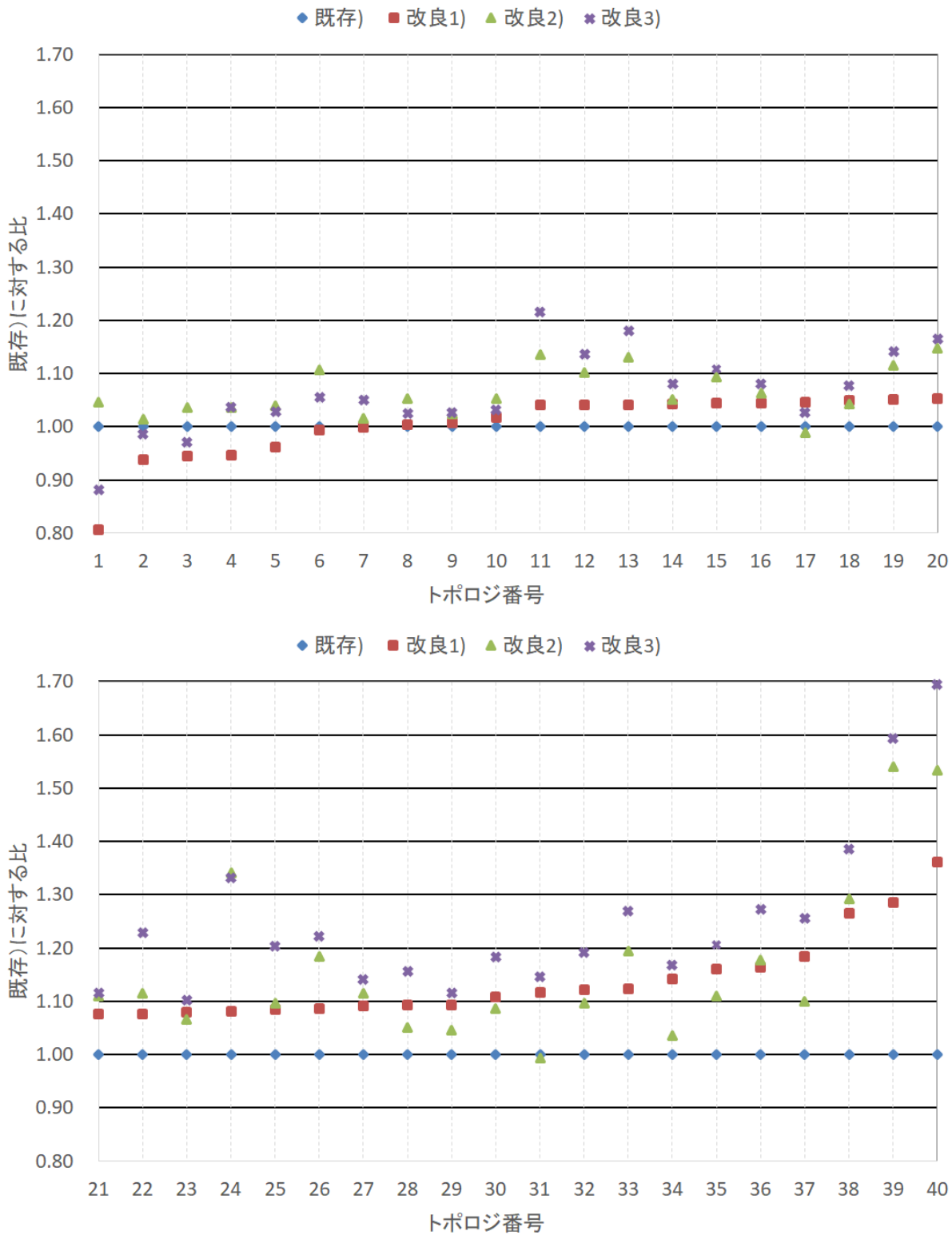

図 3.10　シナリオ (d) のトポロジによる各提案方式

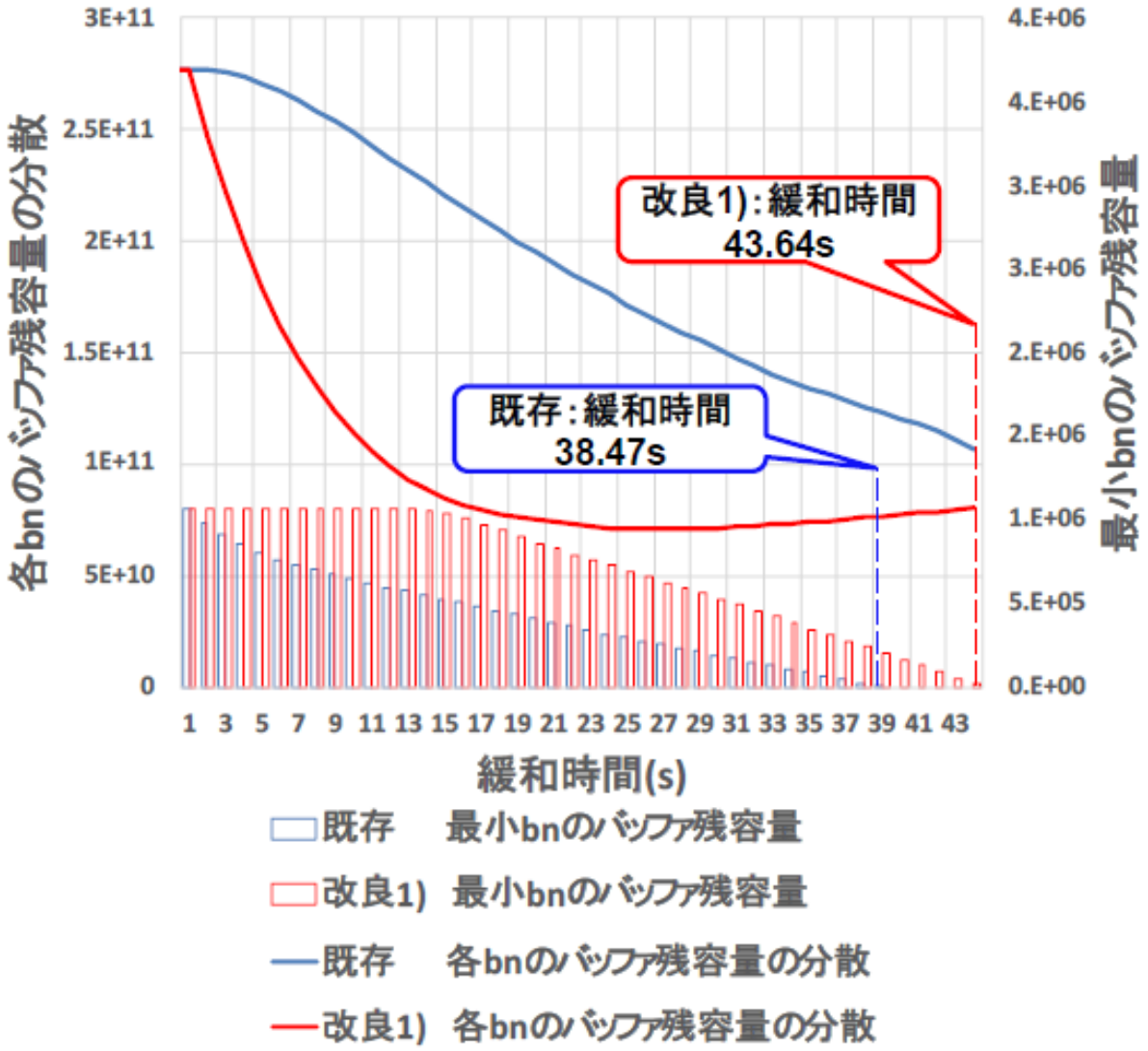

図 3.11　バッファ残容量および分散の時間変化と緩和時間

既存と改良 1) でバッファ残容量が緩和時間に与える影響を比較するために，シナリオ（b）において，「緩和時間」，「バッファ残容量の各 bn の分散の時間変化」及び「最小 bn のバッファ残容量の時間変化」についての追加検証を実施した．結果として図 3.11 のとおり，改良 1) は既存に比べ，バッファ残容量の各 bn の分散が小さくなることがわかった．このことは，各 bn のバッファ残容量の平滑化を実現していると考えることができ，各 bn のバッファ残容量を平滑化することで，バッファからパケットが溢れる緩和時間を引き延ばすことができる（図 3.11 では，既存が 38.47s に対し

表 3.4 送信間隔の差による比較

シナリオ	(e) バッファ容量ランダム・攻撃パケットランダム		
方式	改良 3)		
フィードバック情報間隔 (ms)	10ms(1 倍)	100ms(10 倍)	1000ms(100 倍)
トポロジ平均緩和時間 (s)	36.83	36.66	29.49
トポロジ平均バッファ使用率	0.66	0.66	0.51

て改良 1）が 43.64s となった）．また，bn のバッファ残容量が少なくなると，拡散の効果が小さくなるため，例えば，図 3.11 において 15 秒経過以降のように拡散効果が機能しなくなり，最終的に bn からのバッファ溢れを引き起こしていることもわかる．これらの現象ついては，他のトポロジでも同様の結果であることを確認している．

各シミュレーション単位で観察すると，改良 2) は既存) に対して，シナリオ (a) から (d) のパターンの全てのトポロジで，緩和時間が上回る結果となった．これは，ドリフト項改良手法（3.3.2 項）がパラメータやトポロジの違いを柔軟に吸収し，拡散効果を発揮している結果だと考えられる．しかし，改良 1) については，シナリオ（b）及び (d) で，少数ではあるが緩和時間が既存方式 [13] を下回る場合も確認した．

本方式は拡散現象を利用したものであり，各 bn の状態の情報が遅れるとその効果が発揮できなくなる．そのため，本章では，既存研究の想定と同じ 10ms としている．また，送信間隔について考察するために，フィードバック情報送信間隔を 100, 1000ms という長い間隔とし，シミュレーションを実施した結果，送信間隔が長くなると拡散効果が効きづらくなることも確認した．ただし，10ms と 100ms では大きな違いはなかった (表 3.4).

以上より，改良 3) の方式がトポロジや複雑なパラメータ設定に対して，より柔軟に拡散効果を発揮し，既存方式 [13] の改良として効果が大きいことがわかる．一方，(a) から (d) の全てのシナリオにおいて，トポロジの違いによる緩和時間が異なることが観測できた．このことは，既存方式 [13] においても指摘されたことである．これらについては，トポロジ及びバッファ容量，宛先パケット等のパラメータの設定に要因があると推測されるが，現時点で解析できておらず今後の課題とする．

表 3.5 攻撃規模の増大による影響

シナリオ		(d)	(e)
トポロジ 1	緩和時間	38.75	36.76
	使用率	0.705	0.580
トポロジ 2	緩和時間	39.71	37.96
	使用率	0.623	0.679
トポロジ 3	緩和時間	48.53	42.94
	使用率	0.818	0.735
トポロジ 4	緩和時間	34.99	29.77
	使用率	0.544	0.634
トポロジ 5	緩和時間	41.50	36.74
	使用率	0.652	0.694
合計	緩和時間	40.70	36.83
	使用率	0.668	0.664

3.6.2 攻撃規模の増大による影響

3.5.4 の実験結果を表 3.5 に示す．パラメータについて，1 パケット 1450byte と仮定した場合，(d) のシナリオでは，流入トラフィック送信レートが 0.5〜1.5 × 10^5 (pps), 送信元（流入 bn 数）が 20 なので, 総流入トラフィックレートは 0.5〜1.5 × 10^5 × 1450 × 8 × 20 ≈ 23Gbps である．また，各 bn のバッファ容量は 2.0 × 10^6 パケット分なので，1.0〜3.0 × 106 × 1450 ≈ 2.9GB となる．一方，(e) のシナリオでは，流入トラフィック送信レート 0.5〜1.5 × 10^7（pps），送信元は 20，総流入トラフィックレートは 0.5〜1.5 × 10^7× 1450 × 8 × 20 ≈ 2.3Tbps である．また，各 bn のバッファ容量は 2.0 × 10^8 パケット分であるため，1.0〜3.0 × 10^8× 1450 ≈ 290GB となる．

表 3.5 のとおり，平均緩和時間については，10% 程度下回る結果となり，各トポロジにおいても（e）が (d) を下回る結果となった．だだし，バッファ使用率については，トポロジ 2，4，5 で（e）が (d) を上回る結果となった．なぜ，このような逆転現

象が起きたのかは解明できなかった．しかしながら，オーバーレイネットワークに流入するトラフィック総量を 100 倍にしても，適切に各 bn のバッファ容量や bn 間使用可能帯域などのネットワーク資源を調整することで，十分に緩和の効果を発揮した結果となった．また，各 bn に必要なバッファ容量は，流入量＝流出量＋蓄えられる量となるため，本システムでは「流出量」が少ないため「流入量 ≈ 蓄えられた量」の比例関係になる．すなわち，DDoS の攻撃量が増えても，それに比例してバッファ容量や bn の数及び，bn 間使用可能帯域を調整することで，オーバーレイネットワーク全体のネットワーク資源を攻撃規模に比例させ調整することで，十分に緩和の効果を発揮することが可能となる．

本実験では，文献 [1] で報告された，2.3Tbps の DDoS 攻撃を受けた事例を想定し，攻撃規模を約 2.3Tbps とした．更に，各 bn のバッファ容量についても，平均で約 290GB と現実的な設定とし，実験を実施した．結果として，十分な効果が発揮できており，想定した緩和が可能であると考える．

ただし，本システムで確保する緩和時間が攻撃パケットを検知し，フィルタリングルール適用・排除までに十分かどうかの判断は，標的となるサーバのサービス品質やオーバーレイネットワークの構成形態によって異なる．これらのことを考慮した上で，緩和時間を延ばすための方法は今後の課題とする．

3.7　まとめ

本章では，バッファリングノードで形成するオーバーレイネットワークのトポロジのタイプやバッファ容量等の偏りのあるトポロジやバッファ容量のばらつきがある場合にも，安定した性能を確保することを目的として，ノード間のパケットの転送を制御している転送レート算出式を改良し，長時間パケット損失を防ぐことを可能とした．また，シミュレーションにより既存方式 [13] と提案方式の緩和性能を評価した．実験の結果，本章の提案方式が既存方式 [13] に対して，平均緩和時間がすべて上回る結果となった．加えて，バッファ容量等のパラメータを複雑に設定したものほど，本章の提案方式が既存方式 [13] に対して，緩和時間を引き延ばすことも確認した．これらのことから本章提案方式がトポロジや複雑なパラメータの設定に対して，柔軟に拡散効果を発揮し，バッファあふれによる緩和時間が延長されたと考えられる．ただし，トポロジ及びバッファ容量等のパラメータの設定によっては，本章の提案方式が既存方式 [13] に対して下回る結果も観測されており，これら要因について解明できなかった．

また，DDoS 攻撃の規模は年々増加傾向にあり，DDoS 攻撃の攻撃規模が増大した場合について，本システムへの影響を考察した．結果として，攻撃規模に応じた適切なネットワーク資源の配分を行うことで，攻撃規模が増大した場合でも，十分にミチゲーションの効果を発揮できる結果となった．

第 4 章

マルチクラウドのネットワーク資源の推定について

4.1 研究背景

複数のクラウド，エッジ，オンプレミス環境から構成される分散処理ネットワーク（以下，マルチクラウド）の効率的な運用を実現するためには，様々な状況を想定したネットワーク資源の予測が必要である．しかしながら，ネットワーク資源の予測は，クラウドの処理負荷やネットワーク状態，資源利用のリクエストなど変動する要因が多く，困難な問題である．また，資源の予測や配分の方法としては，様々な数理モデルが提案されているが，必ずしもマルチクラウドを対象としたものでなく，マルチクラウドに対する適切なモデル化が必須となる．

本章で想定するマルチクラウドは，複数のクラウド，エッジ，オンプレミス環境からなり，ノード（サーバー）がマルチクラウド上に点在している．また，ノードには，複数の Pod が展開されており，Kubernetes [18] などのオーケストレーションツールによって制御されているものとする（図 4.1）．本章の目的は，マルチクラウドでサービスを展開しているサービス提供者が必要とするネットワーク資源をオーケストレーションツールで運用されることを前提に，Pod の故障，再生及び，増減を加味した予測を行うことである (2.2 節参照)．すなわち，与えられた条件下でマルチクラウドのネットワーク資源を予測するための数理モデルの提案である．

なお，本章で想定するマルチクラウドの数理モデルは，2.3 節で説明した補給整備システムの解析手法に基づく．具体的には，補給整備システムの中でもベース間補給

方式 (以下，マルチベース) の数理モデルを提案し，その数理モデルを用いてマルチクラウドの数理モデルを提案する．マルチベースは，ベース間でアイテムを融通させることで，遊休資産を使い，システム全体の稼働率を上げる方式である（図 4.2)．このことが，オーケストレーションツールを使用して，ノード間でネットワーク資源を融通させるマルチクラウドの運用方式に類似していることから，補給整備システムで研究されてきた数理モデルがマルチクラウドに適用できると考えたためである．

本章の構成は，補給整備システムで使用されているマルチベースの数理モデルとマルチクラウドの数理モデルを比較する構成としている．具体的には，次のとおりである．4.2 節で関連研究を説明する． 4.3 節でマルチベース，4.4 節でマルチクラウドのモデルの仮定について述べる．4.5 節でマルチベース，4.6 節でマルチクラウドの近似解法を提案する．また，4.7 節では，マルチベース，4.8 節でマルチクラウドの数値計算と結果の考察について述べる．最後に，4.9 節で結論と今後の課題について述べる．

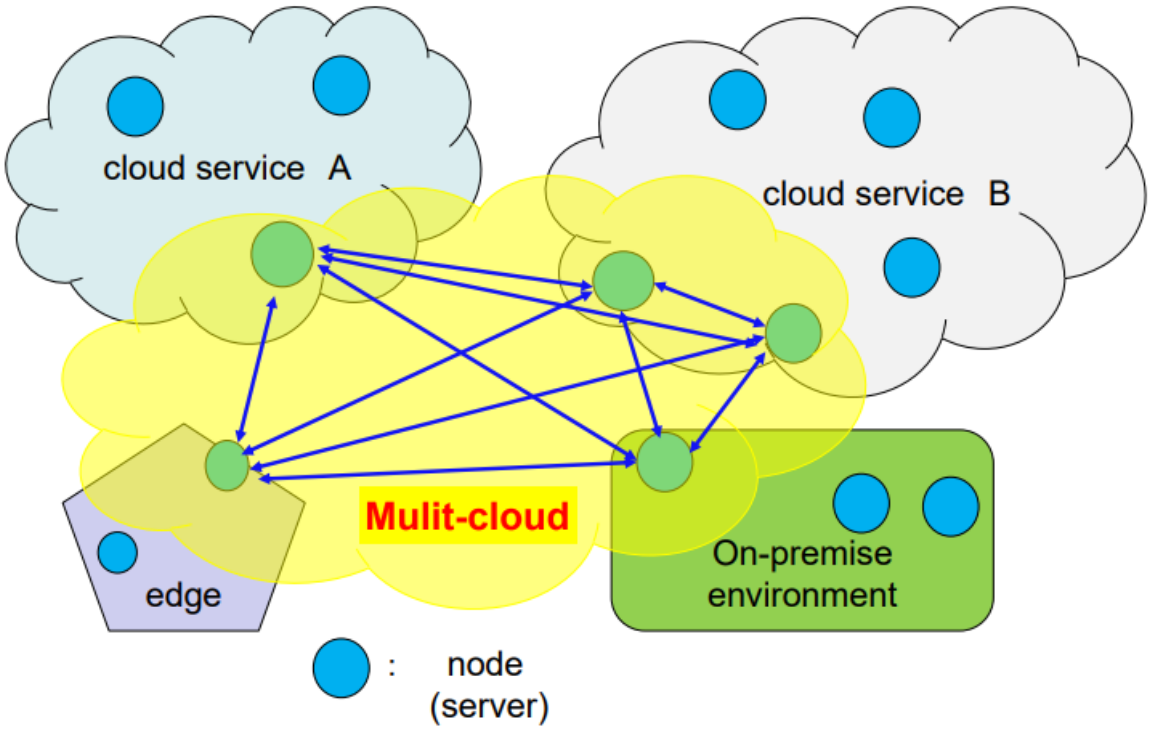

図 4.1　想定するマルチクラウド

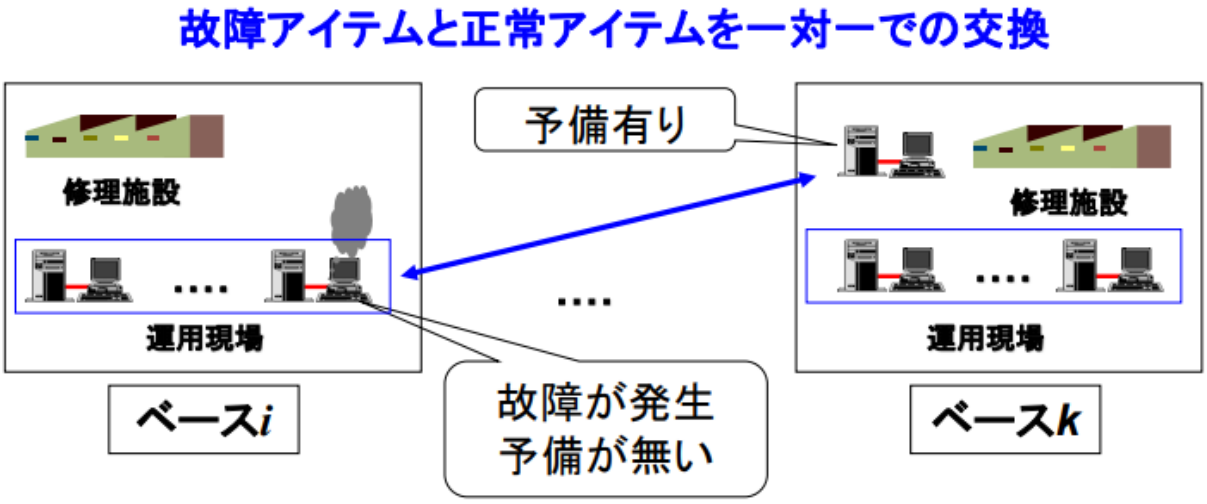

図 4.2　マルチベース

4.2　関連研究

4.2.1　マルチベース

ベース間補給のモデルの解析に関する研究は，現在までに数々の報告がなされている [41][42]．その中で，文献 [43][44] は対象モデルとして，民間航空機の補給整備システムを参考としたモデルの解析を行った．具体的には，複数ベースで同一アイテムが運用，修理，保管が行われ，各ベース間でのアイテムのやり取りを行うマルチベースに関して，文献 [45] や文献 [46] の解析方法を発展させた解析方法を提案した．また，現在までにこのモデルに基づく様々な研究が行われている [47].

しかし，文献 [43][44] の解析方法では，各ベースに存在するアイテム数が十分大と仮定していることや，故障アイテムの平均動作不可能時間が他の故障アイテム数によらず一定としていること，加えて，マルコフモデルによる解析を行っているため，ベース数，アイテム数の増加によって考慮すべき状態数が指数的に増大し，大規模モデルの解析は困難となることなど，いくつかの問題点が挙げられる．

本章では，従来研究の問題点である拠点数，物品数の増加によって考慮すべき状態数が指数的に増大し，大規模モデルの解析が困難となる課題を解決するため，文献 [37] の手法及び文献 [38] によって提案された反復法を利用し，マルチベースの稼働率解析を可能とする近似解析手法を提案する．

4.2.2　マルチクラウド

マルチクラウドに関する研究として，最適な資源配分のアルゴリズムが提案されている．文献 [14] では，シングルクラウドサービス向けの最適化アルゴリズムをマルチクラウドサービスに適用するために，探索ベース，ランダム探索，Automated Machine Learning (AutoML) の観点から検証を行っている．文献 [15] は，マルチクラウド環境において，サービス配置のための最適なプロバイダを選択することができ，さらに，そのプラットフォーム上で通信遅延と通信コストを最小化するアルゴリズムを提案している．また，文献 [16] は，マルチクラウドにおいて，Quality of Service (QoS) を提供しながら，消費電力を最小化し，収益を最大化できる効果的なアルゴリズムも提案している．文献 [17] は，Kubernetes[18] のスケジューリング機能に関する最近の研究を 5 つの領域に分類し，研究の概要，未解決の課題，今後の研

究の方向性についてまとめている．この中で，スケジューリング機能に関する今後のKubernetes[18] の課題として，各アルゴリズムを比較して，評価するためのデファクトスタンダードとなるオープンソースツールが存在しないことが述べられている．その他，マイクロサービスの異常検知やスケジューリング機能など様々な研究が行われている [19][20][21][22].

また，クラウド提供者・利用者にとって，遊休資産を防ぎ，応答時間を短縮するために，適切なネットワーク資源を選択することは，重要な課題である．そのため，ネットワーク資源の使用量やクラウドサービスに対する処理要求の予測は，統計的な手法，最適化アルゴリズム，機械学習などの様々なアプローチでの研究が行われている [23][24][25][26][27][28]．文献 [29] は，クラウドに必要となる将来のワークロードの需要分布を予測する機械学習のモデル（HBNN）を提案している．特徴としては，CPU とメモリなどのデータの不確実性を捕らえ，将来の需要の分布を予測するものである．また，文献 [30] は，動的で変化の激しいクラウドのワークロードを予測するため，予測方法の違うものを組合せて使用するフレームワークである CloudInsight を提案している．CloudInsight は， ユーザーが選択した任意の異なる予測方法を使用することができ，各ローカル予測器の適切な重み（寄与度）を動的に決定できる．また，Auto Regressive Integrated Moving Average (ARIMA)，Support Vector Machine (SVM)，高速フーリエ変換（FFT），Robust Stepwise Linear Regression（RSLR）と CloudInsight を比較することで性能を評価している．今後の課題として，コンテナのオートスケールをサポートするために CloudInsight を改良することが述べられている．

本章では，オーケストレーションツールで制御されたマルチクラウドを利用して，サービスを展開しているサービス提供者が必要とするネットワーク資源をオーケストレーションツールで運用されることを前提に，Pod の故障，再生及び，システムの全体の状態から増減を加味した予測を行うことである．この前提は，先行研究が解析しているものと異なる．加えて，実際に，サービスを提供する事業者の運用実績から推定したパラメータで，本章の数理モデルを使用した計算結果から，ネットワーク資源の予測に関しての有用性を検証する．

このネットワーク資源の予測を行うことで，ベンダロックインの回避，リスク分散，リソースコストの削減に寄与すると考える．

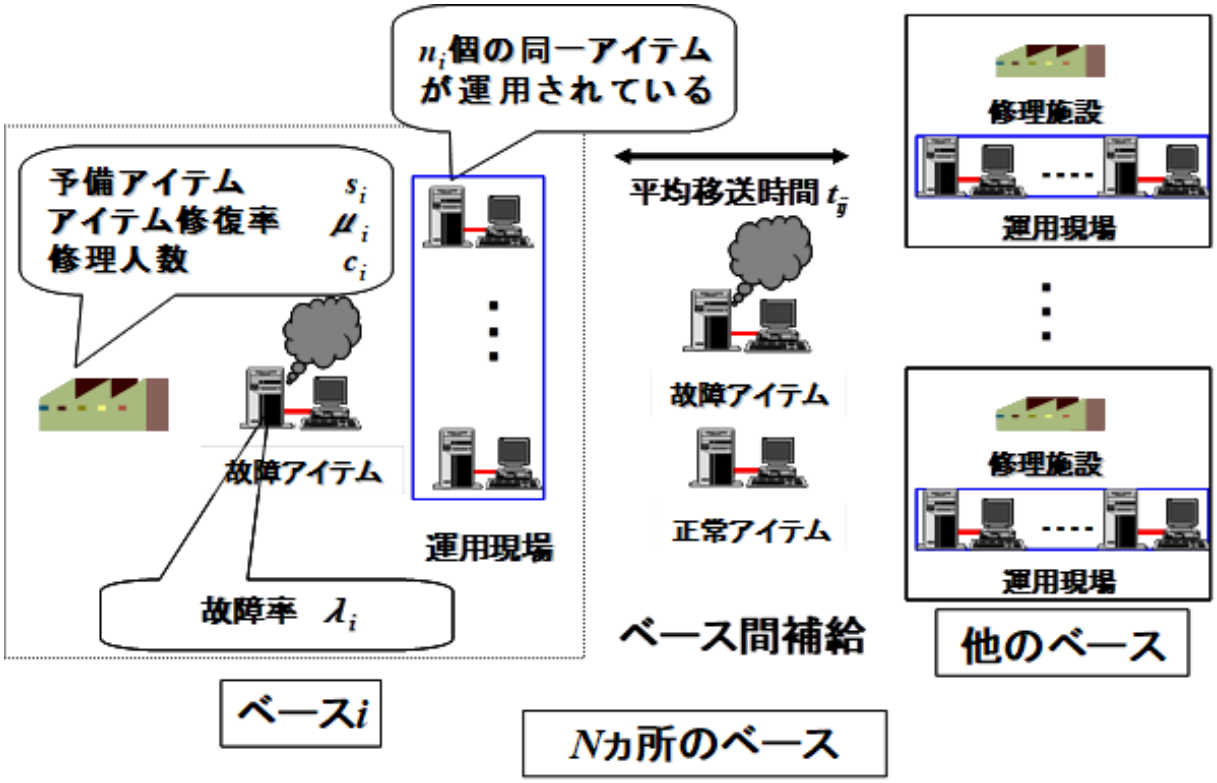

図 4.3　マルチベースの概要

4.3　マルチベースのモデルの仮定

2.3 節に示した補給整備システムは，複数の運用現場 (ベース) において，運用されているアイテムの再利用を前提としたシステムである．また，このようなシステムは，大規模なコンピュータシステムや通信設備など情報通信分野でも多く存在する．例えば，ベース（運用現場）をデータセンターやオンプレ環境，アイテムを物理サーバや通信機器等と考えることができる．本章のマルチベースでは，ベースをデータセンターやオンプレ環境，アイテムを物理サーバや通信機器と考えモデル化を行っている．次項でマルチベースに関するモデルの仮定を示す．

4.3.1 モデルの仮定

(1) モデルは，複数ベース及びそのベースで運用されるアイテム，修理施設から構成される．

(2) 修理可能な同一種類のアイテムが N ヵ所のベースで運用されている．

(3) ベース i ($i = 1, 2, ..., N$) では n_i 個の運用アイテムと s_i 個の予備アイテムが配置されている．

(4) ベース i での各アイテムの故障率は λ_i とする．

(5) ベース i でアイテムの故障が発生した時，以下のいずれかの処置がとられる．

(a) ベース i に予備アイテムが存在した場合，直ちに予備アイテムと交換され，故障アイテムは，ベース i 内の修理施設に送られる．

(b) ベース i に予備アイテムが存在せず，かつ他のベースに予備アイテムが存在する場合，ベース間補給が行われる．(仮定 (6) 参照)

(c) ベース i に予備アイテムが存在せず，かつ他のベースに予備アイテムが存在しない場合，故障アイテムは，ベース i 内の修理施設に送られ，予備アイテム待ちとなる．

(6) ベース間補給は，利用可能な予備アイテムが存在しないベース i で故障が発生し，かつ他のベースに予備アイテムが存在するときに実施される．また，同時に予備アイテムの補給を受けたベースから故障アイテムが補給元のベースの修理施設に移送される．

(7) ベース間補給が生じた場合，複数のベースに予備アイテムが存在していたならば，複数のベースから利用可能な予備アイテム数に比例した確率によって選ばれたベースより補給を受ける．

(8) ベース i での修理施設の修理人１人当りの修復率は μ_i であり，また修理人の数を c_i とする．

(9) ベース i で修理が完成した場合，ベース i で正常アイテム数が n_i を越えなければ運用アイテムとして利用され，越えればベース i の予備アイテムとなる．

(10) ベース内での故障・修理完了アイテムの移送時間は十分短く，無視できる．

(11) ベース i，ベース j 間のアイテム平均移送時間を t_{ij} とする．

図 4.3 に概要図を示す．

4.3.2　モデルの仮定の表記

N　ベースの数

i　ベース番号（$i = 1, 2, ..., N$）

n_i　ベース における運用アイテム数

λ_i　ベース i におけるアイテムの故障率

μ_i　ベース i におけるアイテムの修復率

c_i　ベース i の修理施設における修理人数

s_i　ベース i における予備アイテム数

t_{ij}　ベース i とベース j 間の平均移送時間

ξ_i　ベース i での平均稼働アイテム数

γ_i　ベース i に予備アイテムが存在する確率

α_{kj}　ベース j に予備アイテムが存在する条件下で，ベース k に対してベース j から正常アイテムが送られる確率

β_{kj}　ベース k に対してベース j から正常アイテムを移送する事象の発生率 (ベース j に対してベース k から故障アイテムを移送する事象の発生率)

R_i　ベース間補給によってベース i に対し，移送中の故障アイテム数の期待値

Q_i　ベース間補給によってベース i に対し，移送中の正常アイテム数の期待値

Rx_i　ベース i に予備アイテムが存在する場合，他のベースで故障が発生し，かつその故障アイテムが他のベースから輸送された事象が発生する発生率

Qx_i　ベース i に予備アイテムが存在しない場合，ベース i で故障が発生し，かつその故障アイテムが他のベースへ輸送される事象が発生する発生率

$P_{tif}(d)$　ベース i に向けてベース間を移送中である故障アイテムの総数が d である確率

$P_{tin}(m)$　ベース i に向けてベース間を移送中である正常アイテムの総数が m である確率

$P_i(w)$　ベース i に属する動作不可能アイテムの総数が w である確率

4.4 マルチクラウドのモデルの仮定

4.4.1 モデル化の概念

本節では，前節のマルチベースのモデル化を参考に，マルチクラウドのモデル化を行う．マルチベースは，遊休資産を減らす目的として，修理可能なアイテムをベース間で融通させる方式である．

一方，本章で想定するマルチクラウドは，遊休資産を減らすためオーケストレーションツールにより，各ノード間を再生可能な Pod が移動する方式となる．具体的には，Pod の故障，再生及び，増減を加味したモデルとなる．すなわち，ベースをノード，アイテムを Pod と考えると，マルチベースとマルチクラウドは，概ね同じ動きとなる．ただし，マルチベースとの違いは，マルチクラウド全体の稼働状況に依存して，Pod の数が増減する点がある．この点を考慮して，次項でマルチクラウドのモデルの仮定を示す．

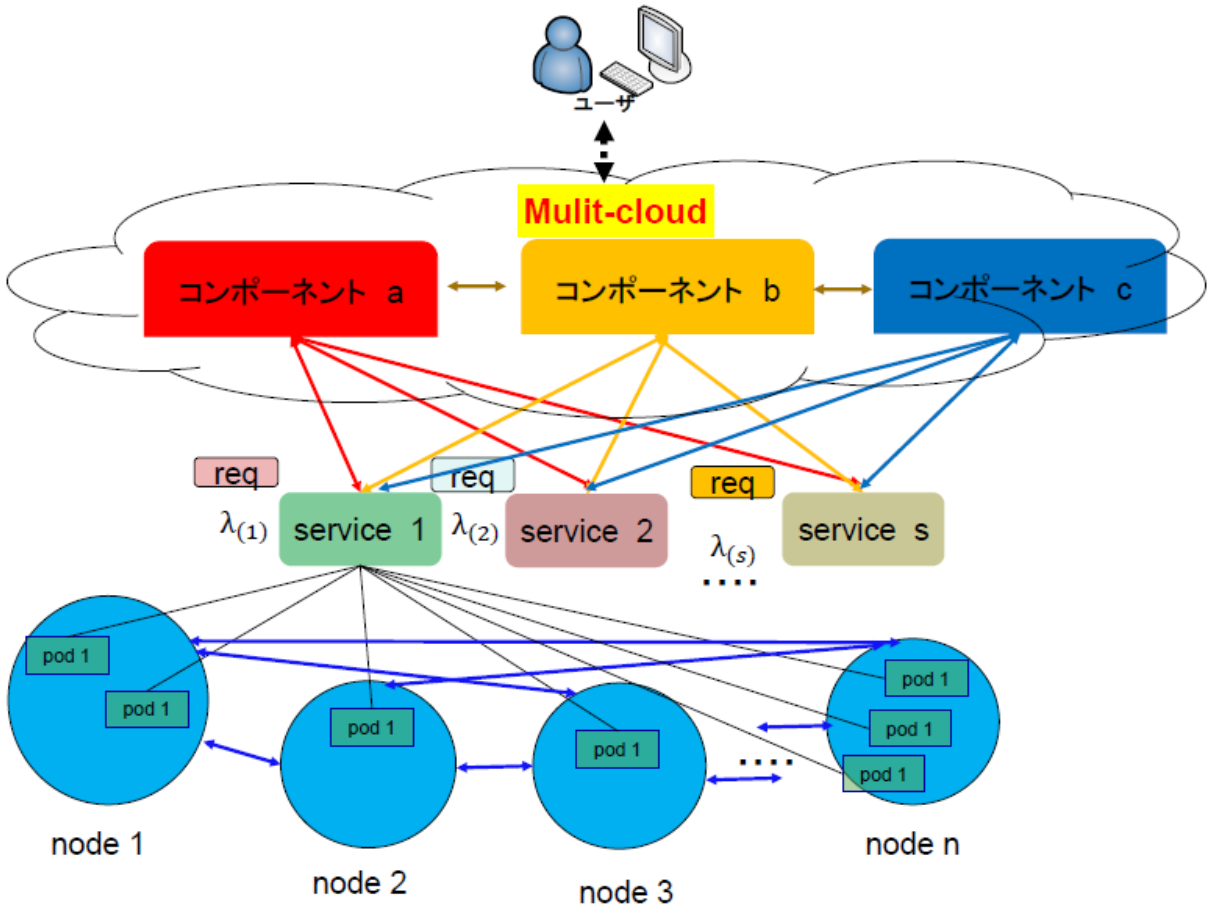

図 4.4　マルチクラウドの概要

4.4.2　モデルの仮定

マルチクラウド (図 4.4)

想定するネットワークは複数のクラウド，エッジ，オンプレミス環境から構成される分散処理ネットワーク（マルチクラウド）である．マルチクラウドには，ノード n $(n = 1, 2, \cdots, N)$ が存在する．

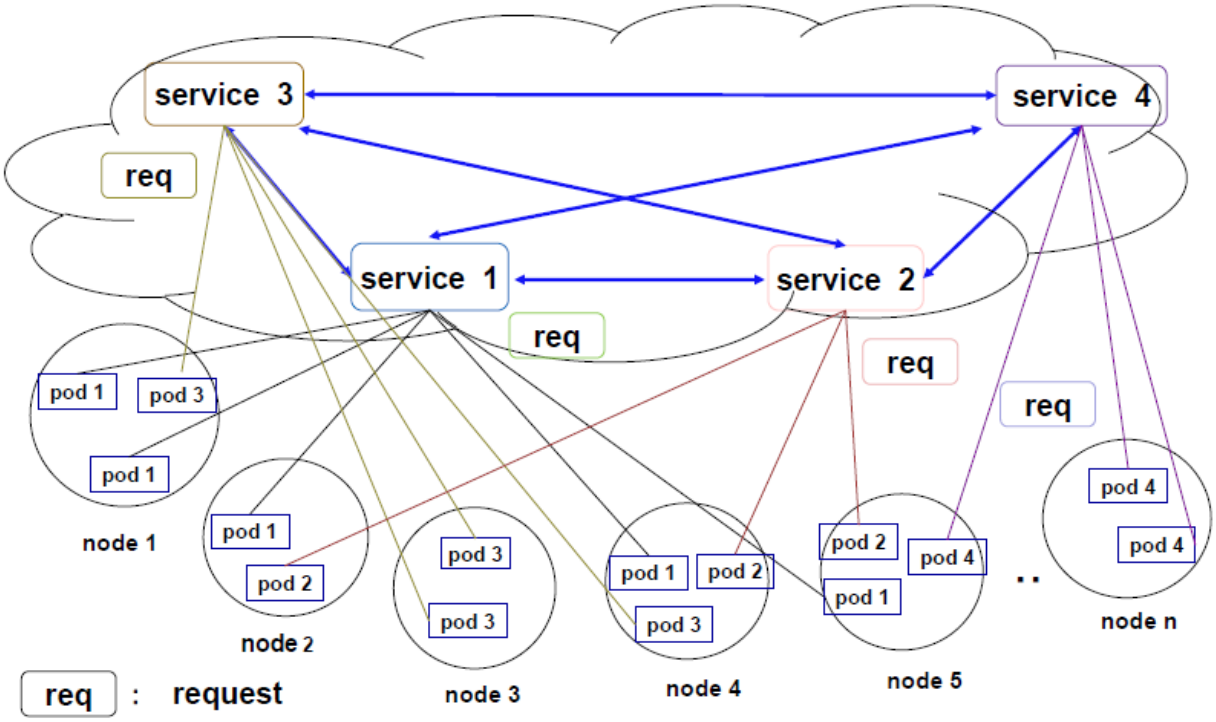

図 4.5　サービスやノードの定義

サービスやノードの定義 (図 4.5)

各機能単位で request を実行するサービス s ($s = 1, 2, \cdots, S$) が存在する．サービス s は，各ノードに点在している同一機能（性能については各ノードに依存）の Pod の集合である．また，Pod とは，オーケストレーションツールで実行できるアプリケーションの最小単位と定義する．

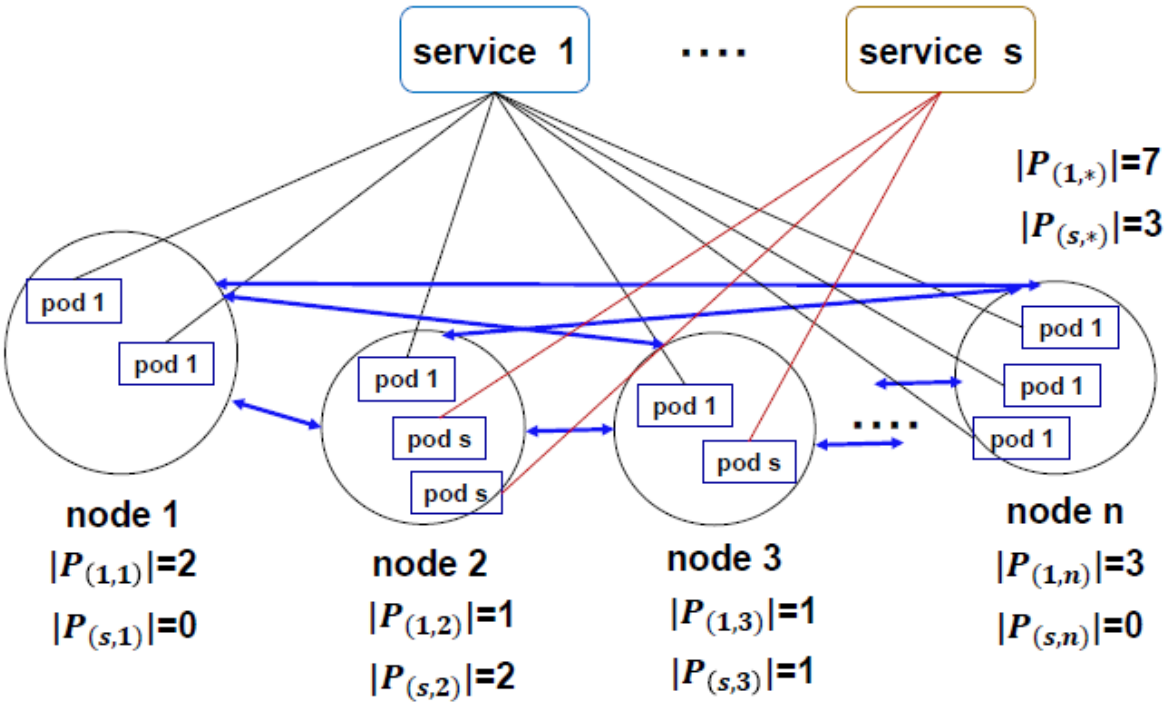

図 4.6　サービスマネジメント

サービスマネジメント (図 4.6)

各ノードには，複数の Pod が展開されており，ノード n にあるサービス s の機能を持つ Pod の集合を $P_{(s,n)}$ とする．

$P_{(s,*)}$ を全ノードのサービス s の Pod の集合とする ($P_{(s,n)} \subset P_{(s,*)}$)．図 4.6 において，$|P_{(s,n)}|$ は，ノード n にあるサービス s の機能を持つ Pod の要素数を表す．また，各ノードの Pod 数には上限はないものとし，特定のサービスに対応する Pod が存在しないノードもある．

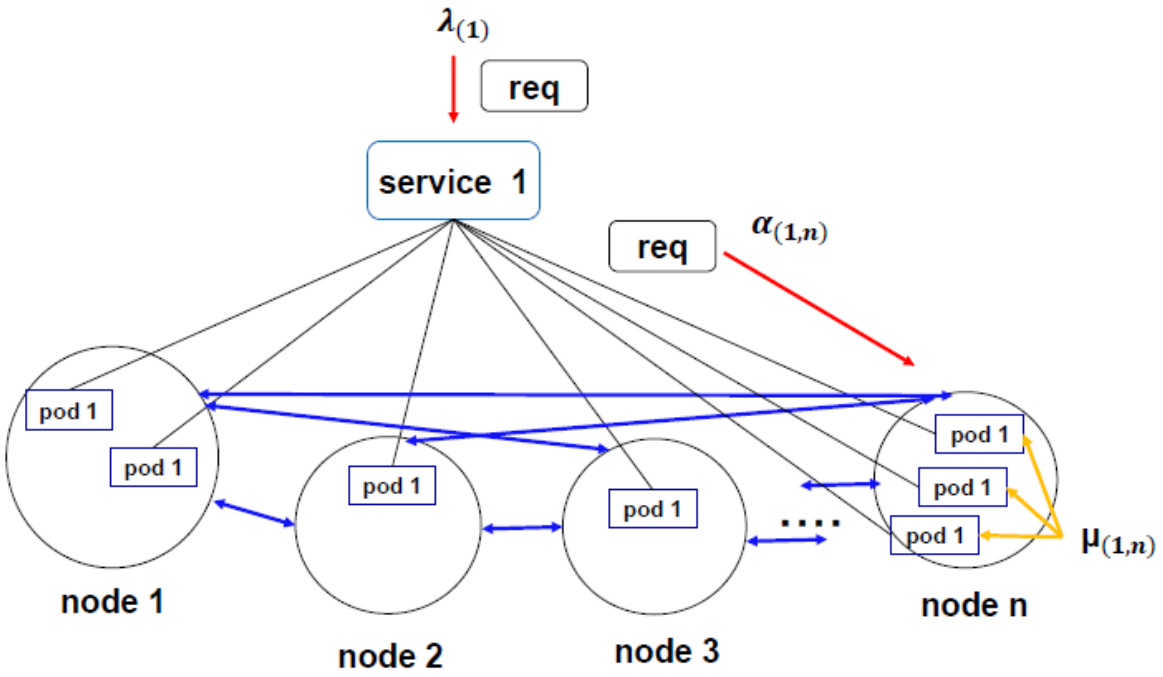

図 4.7　ノードへの request の割当

ノードへの request の割当 (図 4.7)

サービス s への request は，平均到着率 $\lambda_{(s)}$ となる．

また，$P_{(s,n)}$ に対しての request は，サービス s への request から確率 $\alpha_{(s,n)}$ で振り分けられ，$P_{(s,n)}$ の 稼働している要素（Pod）に均等に到着する．

各 Pod に到着した request は，各ノードに依存した一つの Pod の平均処理率 $\mu_{(s,n)}$ で処理される．

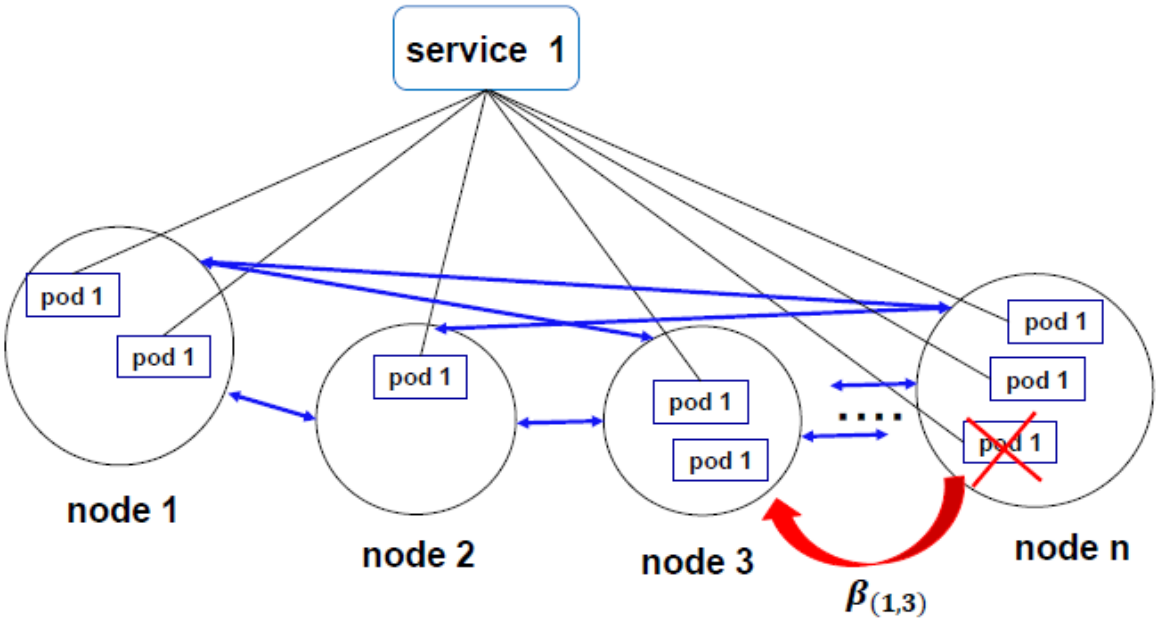

図 4.8　故障発生時の Pod の移動

故障発生時の Pod の移動 (図 4.8)

ノード n にあるサービス s の機能を持つ Pod は，平均故障時間 $T_{f(s,n)}$ で故障（Pod が処理機能を失う）する．その後，Pod はノード i において，確率 $\beta_{(s,i)}$ で $P_{(s,i)}$ の要素として展開される．また，展開から稼働が開始されるまでの平均展開待ち時間は，$T_{d(s,i)}$ となる．

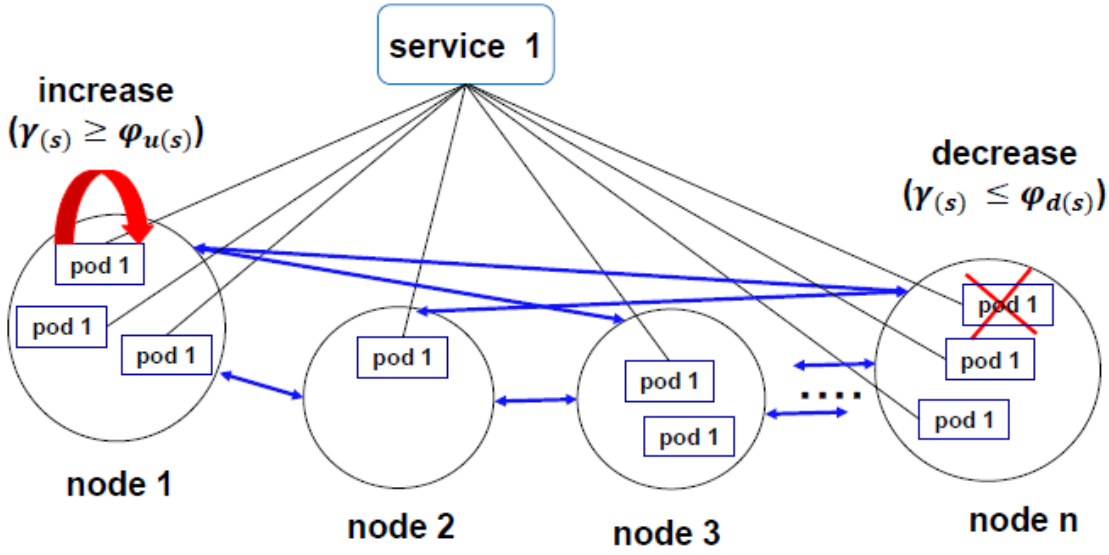

図 4.9　Pod 数の調整

Pod 数の調整 (図 4.9)

サービス s は，一定時間間隔 $T_{o(s)}$ で， $|P_{(s,*)}|$ (要素数) の見直しを行う．見直しには，サービス s の平均利用率 $\gamma_{(s)}$, により，Pod を 1 つ増やすための基準平均利用率 $\phi_u(s)$ と Pod を 1 つの減らすための基準平均利用率 $\phi_d(s)$ を用いて，以下のように実行される．

$$|P_{(s,*)}| = \begin{cases} |P_{(s,*)}| + 1 & (\gamma_{(s)} \geq \phi_{u(s)}), \\ |P_{(s,*)}| - 1 & (\gamma_{(s)} \leq \phi_{d(s)}). \end{cases} \tag{4.1}$$

ただし， 増減を行う ノード n は確率 $\beta_{(s,n)}$ で決定される． また，ノード n の増加に必要な待機時間は $T_{d(s,n)}$ とする．

すなわち，Pod の数を 1 つ増やす条件は，障害が発生したときと，平均利用率が閾値を超えたときの 2 つとなる．

4.4.3　モデルの仮定の表記

モデル仮定で使用される表記をまとめると，以下のようになる．

N　ノードの数
n　ノードの番号 ($n = 1, 2, \cdots, N$)
S　サービスの数
s　サービスの番号 ($s = 1, 2, \cdots, S$)
$P_{(s,n)}$　ノード n にあるサービス s の Pod の集合
$\lambda_{(s)}$　$P_{(s,*)}$ に送られてくる request の平均到着率
$\mu_{(s,n)}$　$^{\forall}p \in P_{(s,n)}$ が request を実行する平均処理率
$\alpha_{(s,n)}$　$P_{(s,n)}$ に対して request が振り分けられる確率
$\beta_{(s,n)}$　$^{\forall}p \in P_{(s,n)}$ が故障時及び全体の数の調整時に待機となるノードが n である確率
$\gamma_{(s)}$　サービス s の平均利用率
$\phi_{u(s)}$　$|P_{(s,*)}|$ を 1 つ増やすための基準平均利用率
$\phi_{d(s)}$　$|P_{(s,*)}|$ を 1 つ減らすための基準平均利用率
$T_{f(s,n)}$　$^{\forall}p \in P_{(s,n)}$ の平均故障時間
$T_{d(s,n)}$　$^{\forall}p \in P_{(s,n)}$ の平均展開時間
$T_{o(s)}$　Pod の増減を判断する時間間隔

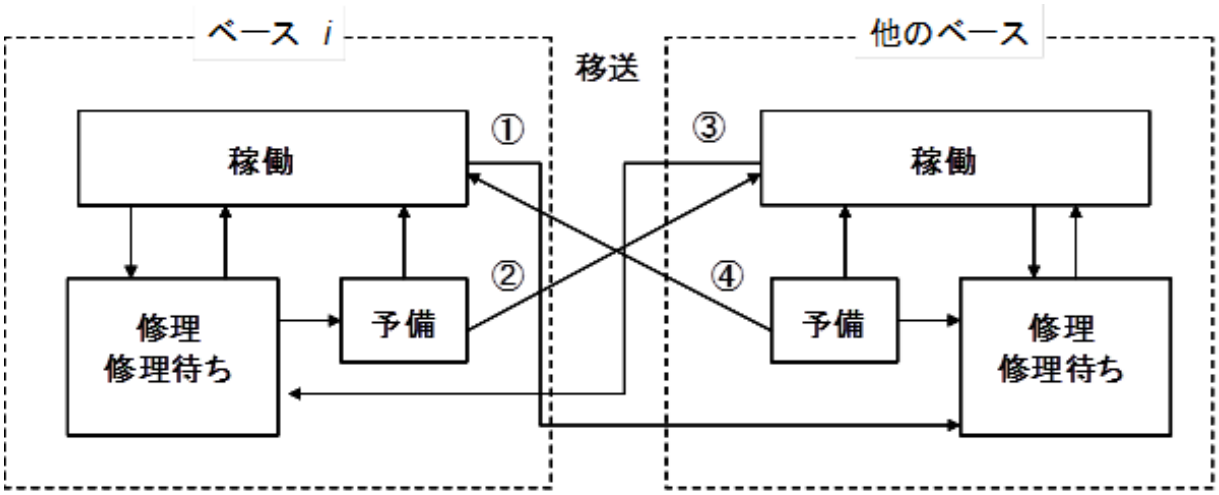

図 4.10　ベース間補給によるアイテムのフロー

4.5　マルチベースの近似解法

この節では，4.3 節に示したマルチベースの近似解法を提案する．

4.5.1　近似解法の概要

動作不可能アイテムの状態

2.3 節に示した 2 段階整備方式における文献 [38] の解析手法は，平均稼働アイテム数を未知パラメータとし，各ベースでのアイテムの故障発生率を一定とした．これにより，文献 [37] の動作不可能アイテム数の分布決定法を利用し，各ベースでの稼働率解析を可能とした．

2.3 節の手法をマルチベースへ適用するために重要となることは，各ベースに所属する動作不可能アイテムがベース間補給を考慮した場合，どのような状態が存在するかである．

例えば，ベース間補給を考慮しない場合，各ベースのアイテムは運用，整備，保管のいずれかの状態にある．一方，ベース間補給を考慮した場合，それに加えてアイテムのベース間での移送も考慮しなければならない．

ここで，ベース間補給による正常アイテムと故障アイテムの移送の状態は，以下の

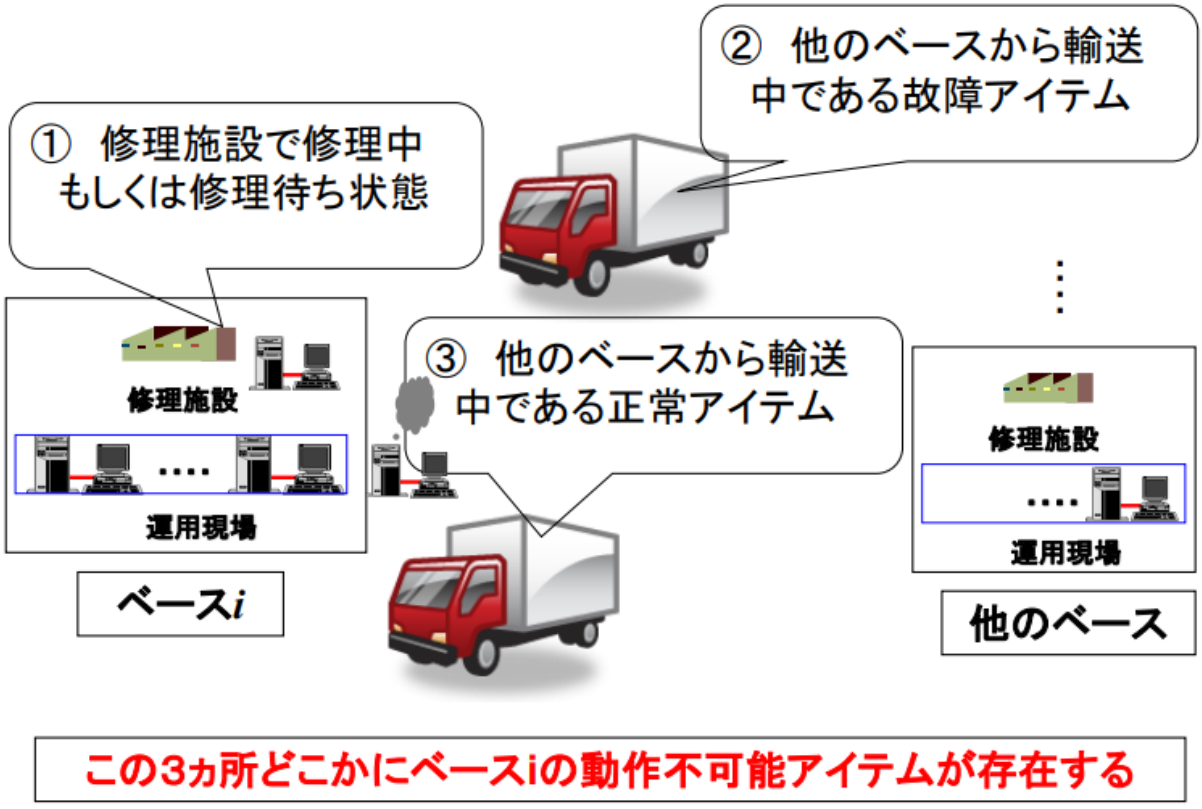

図 4.11　動作不可能アイテムが存在する場所

4 つに分類される．(図 4.10)

(1) ベース i での故障発生によりベース間補給が成立，ベース i から他のベースに対して故障アイテムを移送中の状態
(2) 他のベースでの故障発生によりベース i との間でベース間補給が成立，ベース i から他のベースに対して正常アイテムを移送中の状態
(3) 他のベースでの故障発生によりベース i との間でベース間補給が成立，他のベースからベース i に対して故障アイテムを移送中の状態
(4) ベース i での故障発生によりベース間補給が成立，他のベースからベース i に対して正常アイテムを移送中の状態

(1) と (4) 及び (2) と (3) は，ベース間補給として同時に発生する事象である．ここで，ベース間補給が成立した場合，成立したベース間において，交換アイテム (すな

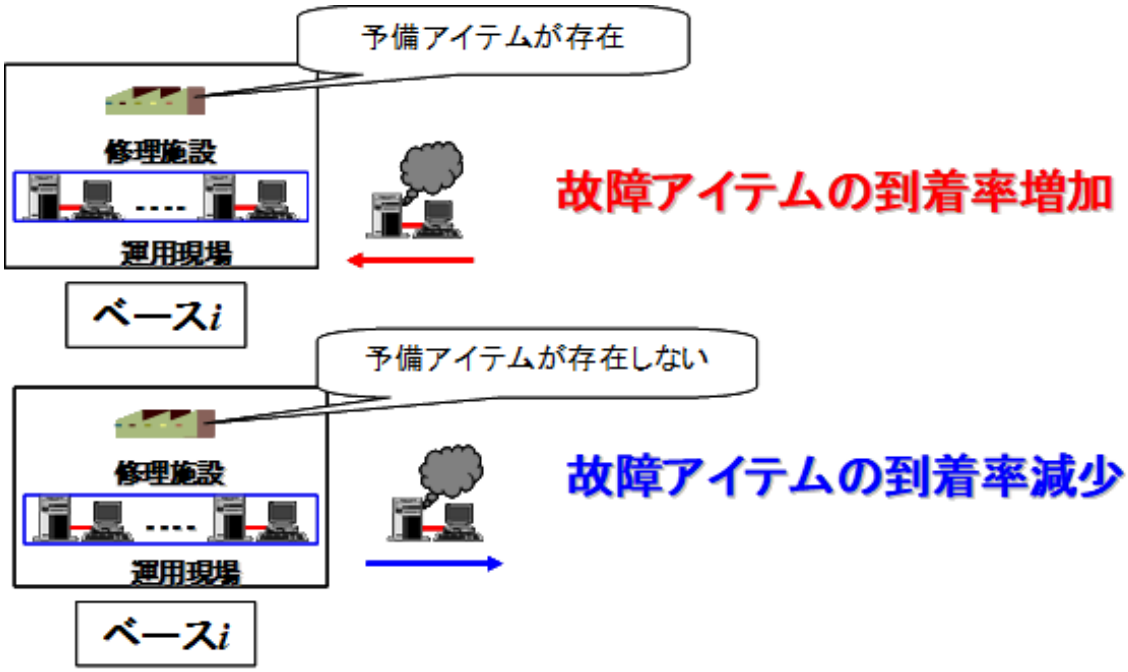

図 4.12　ベース修理施設への故障アイテムの到着率

わち故障アイテムと正常アイテム) の所属ベースを入れ替えると考える．このことで，ベース i に所属する動作不可能アイテムの存在する状態は以下の 3 つと考えることができる (図 4.11).

(a) ベース i へ故障アイテムが移送中
(b) ベース i へ正常アイテムが移送中
(c) ベース i の修理施設での修理，修理待ち

ベース修理施設への到着率

ベース間補給を考慮したモデルでは，ベースの予備アイテムの有無により，修理施設へ故障アイテムの到着率が異なる．

例えば，ベース間補給を考慮しない場合，ベースでの故障発生率が修理施設への到着率となる．一方，ベース間補給を考慮した場合，ベースに予備アイテムが存在したならば，他のベースでの故障の発生により，その故障アイテムをベース間の補給によって受け入れる可能性ある．すなわち，ベースの修理施設への故障アイテムの到着

率が増加する．

逆に，ベースに予備アイテムが存在しない場合，ベース間補給の成立により，他のベースに対して故障アイテムを移送することになる．すなわち，故障アイテムの到着率が減少する (図 4.12)．このような現象をベース修理施設の故障アイテムの生起率に考慮しなければならない．

4.5.2　近似解法

前項で示した考えに基づき，マルチベースのモデルに対する近似解析法は，以下のようになる．ただし，ベース i の平均稼働アイテム数を未知パラメータ ξ_i として導入すると，ベース i での故障発生は，発生率 $(\xi_i\lambda_i)$ のポアソン過程に従うと考えことができる．

変数の定義

ベース i に予備アイテムが存在する確率 γ_i を用いて，ベース j に予備アイテムが存在する条件下で，ベース k に対してベース j から正常アイテムが送られる確率 α_{kj} は以下のように近似する．

$$\alpha_{kj} = \frac{1}{1 + \sum_{s=1,s\neq k,j}^{N} \gamma_s} \tag{4.2}$$

上式を用いると，ベース k に対してベース j から正常アイテムを移送する事象の発生率 β_{kj} (ベース j に対してベース k から故障アイテムを移送する事象の発生率) は，以下のようになる．

$$\beta_{kj} = \xi_k\lambda_k(1-\gamma_k)\gamma_j\alpha_{kj} \tag{4.3}$$

ベース間で移送中のアイテム数の分布

ベース間補給によってベース i に対して，移送中である故障アイテム数の期待値 R_i は，β_{ij} を用いて以下のようになる．

$$R_i = \sum_{k=1,k\neq i}^{N} \beta_{ki}t_{ki} \tag{4.4}$$

また，同様にベース間補給によってベース i に対して，移送中である正常アイテム数の期待値 Q_i は，β_{kj} を用いて以下のようになる．

$$Q_i = \sum_{j=1, j\neq i}^{N} \beta_{ij} t_{ij} \tag{4.5}$$

以上より他のベースからベース i に移送中である故障アイテム数が d である確率 $P_{tif}(d)$ 及び他のベースからベース i に移送中である正常アイテム数が m である確率 $P_{tin}(m)$ は次式で表される．

$$P_{tif}(d) = (R_i)^d \frac{\exp(-R_i)}{d!} \tag{4.6}$$

$$P_{tin}(m) = (Q_i)^m \frac{\exp(-Q_i)}{m!} \tag{4.7}$$

ベース修理施設に存在する故障アイテム数の分布

ベース i に予備アイテムが存在する場合，他のベースに予備アイテムを移送することに伴い，そのベースからベース i に対して故障アイテムを移送してくる場合が考えられる．そのため，ベース i の修理施設への故障アイテムの到着は，ベース i の故障発生による場合と，他のベースの故障発生による 2 通りがあり，後者について，その発生率 Rx_i は以下となる．

$$Rx_i = \sum_{k=1, k\neq i}^{N} \alpha_{ki} \xi_k \lambda_k (1 - \gamma_k) \tag{4.8}$$

また，ベース i に予備アイテムが存在しない場合，ベース i で故障が発生し，かつその故障アイテムが他のベースへ輸送される事象が発生する発生率 Qx_i は，以下となる．

$$Qx_i = \sum_{j=1, j\neq i}^{N} \alpha_{ij} \xi_i \lambda_i \gamma_j \tag{4.9}$$

以上を用いてベース i 内に予備アイテムが存在する場合の到着率 λ_i^A，存在しない場合の故障アイテムの修理施設への到着率 λ_i^* は，それぞれ以下のようになる．

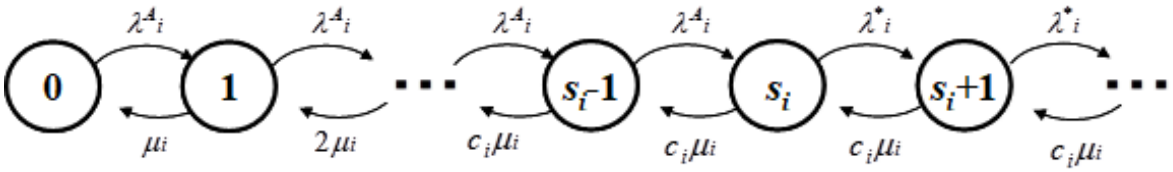

図 4.13　ベース修理施設の出生死滅過程 ($c_i \leq s_i$).

$$\lambda_i^A = \xi_i \lambda_i + R x_i \tag{4.10}$$

$$\lambda_i^* = \xi_i \lambda_i - Q x_i \tag{4.11}$$

以上を用いて 2 つの到着率を一定とし, 図 4.13 に示す 1 次元出生死滅過程 ($M/M/c_i$) を用いてベース修理施設に故障アイテムが存在する確率 $P_{ib}(k)$ は次式で表される.

$c_i \leq s_i$ の場合

$$P_{ib}(k) = \begin{cases} \dfrac{\rho_{iA}^k}{k!} P_{ib}(0) & (1 \leq k < c_i) \\ \dfrac{\rho_{iA}^{c_i}}{c_i!} u_{ic}^{k-c_i} P_{ib}(0) & (c_i \leq k < s_i) \\ \dfrac{\rho_{iA}^{c_i} u_{ic}^{s_i-c_i} \rho_{ix}^{k-s_i}}{c_i!} P_{ib}(0) & (s_i \leq k) \end{cases} \tag{4.12}$$

$$P_{ib}(0) = [\sum_{x=0}^{c_i-1} \frac{\rho_{iA}^{x-1}}{x!} + \frac{\rho_{iA}^{c_i}}{c_i!} (\sum_{x=0}^{s_i-c_i-1} u_{ic}^x + \frac{u_{ic}^{s_i-c_i}}{1-\rho_{ix}})]^{-1} \tag{4.13}$$

$s_i < c_i$ の場合

$$P_{ib}(k) = \begin{cases} \dfrac{\rho_{iA}^{k}}{k!} P_{ib}(0) & (1 \le k < s_i) \\ \dfrac{\rho_{iA}^{s_i}}{k!} \rho_{is}^{k-s_i} P_{ib}(0) & (s_i \le k < c_i) \\ \dfrac{\rho_{iA}^{s_i} \rho_{is}^{c_i-s_i} \rho_{ix}^{k-c_i}}{c_i!} P_{ib}(0) & (c_i \le k) \end{cases} \tag{4.14}$$

$$P_{ib}(0) = [\sum_{x=0}^{s_i-1} \frac{\rho_{iA}^{x-1}}{x!} + \rho_{iA}^{c_i} (\sum_{x=s_i}^{c_i-1} \frac{\rho_{is}^{x}}{x!} + \frac{\rho_{is}^{s_i-c_i}}{c_i!(1-\rho_{ix})})]^{-1} \tag{4.15}$$

ただし，$\sum_{x=0}^{-1} u_{ic}^{x} = 0$,$\rho_{iA} = \dfrac{\lambda_i^A}{\mu_i}$，$u_{ic} = \dfrac{\lambda_i^A}{c_i \mu_i}$，$\rho_{is} = \dfrac{\lambda_i^*}{\mu_i}$，$\rho_{ix} = \dfrac{\lambda_i^*}{c_i \mu_i}$，$\rho_{ix} < 1$ とする．

ベース i に所属する動作不可能アイテムの総数の分布

定常状態において，ベース i で運用される動作不可能アイテム数は，ベース修理施設にあるアイテム数，移送中のアイテム数の合計となる．従って，ベース i に所属する動作不可能アイテムの総数が w である確率 $P_i(w)$ は式 (4.6)，式 (4.7)，式 (4.12) 及び式 (4.14) より次式で表される．

$$P_i(w) = \sum_{m} \sum_{d} P_{tin}(m) P_{tif}(d) P_{ib}(w-d-m) \tag{4.16}$$

また，式 (4.16) よりベース i に予備アイテムが存在する確率 γ_i，平均稼働アイテム数 ξ_i は，以下となる．

$$\gamma_i = \sum_{w=0}^{s_i-1} P_i(\mathrm{w}) \tag{4.17}$$

$$\xi_i = \sum_{w} min(n_i, n_i + s_i - w) P_i(w) \tag{4.18}$$

以上よりベース i でのシステムの稼働率を A_i とすると，A_i は次式で得られる．

$$A_i = \frac{\xi_i}{n_i} \tag{4.19}$$

4.5.3　計算アルゴリズム

以下の反復アルゴリズムより，各ベースでの稼働率解析を行う．

Step0 ベース i の平均稼働アイテム数 ξ_i，予備アイテム存在確率 γ_i に，適当な初期値を与える

Step1 $\rho_{ix} < 1$ ならば Step2 へ．もし，この条件を満たさなければ計算を終了する (解析不可能).

Step2 ベース i に関して各状態でのアイテム数の分布などを式 (4.2)〜式 (4.15) より計算する．

Step3 Step2 の結果より，ベース i に所属する動作不可能アイテムの総数が w である確率 $P_i(w)$ を式 (4.16) より計算する．

Step4 Step3 での計算結果をもとに，各ベースでの平均稼働アイテム数を式 (4.18) より計算し，その値 ξ_i^* をとする．

Step5 以下の式を満たすなら Step6 へ，そうでなければ $\xi_i^* \to \xi_i$ かつ γ_i を式 (4.17) より再計算し Step1 に戻る．

$$\left| \frac{\xi_i - \xi_i^*}{\xi_i} \right| = \epsilon \tag{4.20}$$

ただし $\epsilon(> 0)$ は十分に小さい値とする．

Step6 ベース i のアイテムの稼働率 A_i を式 (4.19) より計算し，終了する．

4.6 マルチクラウドの近似解法

マルチクラウドの解析では，オーケストレーションツールで制御されたマルチクラウドを利用して，サービスを展開しているサービス提供者が必要とするネットワーク資源をオーケストレーションツールで運用されることを前提に，Pod の故障，再生及び，増減を加味した予測を行うことである．

本節では，前節のマルチベースでの稼働率解析方法を用いて，与えられた条件下における定常状態でのマルチクラウドの稼働状況（平均稼働 Pod 数）を解析するための近似解法を提案する．

4.6.1 近似解法の概要

前節では，平均稼働アイテム数を未知パラメータとし，未知パラメータに適当な初期値を与えることで，各ベースのアイテムの故障発生率を一定と考え，不稼働アイテムの生起率を考慮し，各状態でのアイテム数の分布を計算している．そして，この分布関数を使用して，新たな平均稼働アイテム数を計算する．計算に使用した未知パラメータと計算した平均稼働アイテム数の差が小さくなるまで，反復法を使用して再計算する．

一方，マルチクラウドでは，不稼働アイテムが各ノードでの展開待ちの Pod となる．また，各ノードの稼働数や各サービスの利用率によって，各ノードで展開待ちとなる Pod の生起率に違いが生じる (図 4.14)．以下に近似解法の概要を示す．

(1) 各ノードの平均稼働 Pod 数を未知パラメータとし，未知パラメータに適当な初期値を与え，各ノードの故障発生を一定と考える．
(2) 各ノードの展開待ちや故障の Pod の数の分布関数を求める．
(3) 分布関数から，各ノードの平均稼働 Pod 数を計算する．
(4) (1) で設定した未知パラメータと (3) で計算した平均稼働 Pod 数の差がすべてのノードで一定以下の場合は，計算した平均稼働 Pod 数を解とする．また，差が一定以上の場合は，計算した平均稼働 Pod 数を用いて，未知パラメータを修正し，(1) から再計算する．

次項に詳細な解析方法を示す．

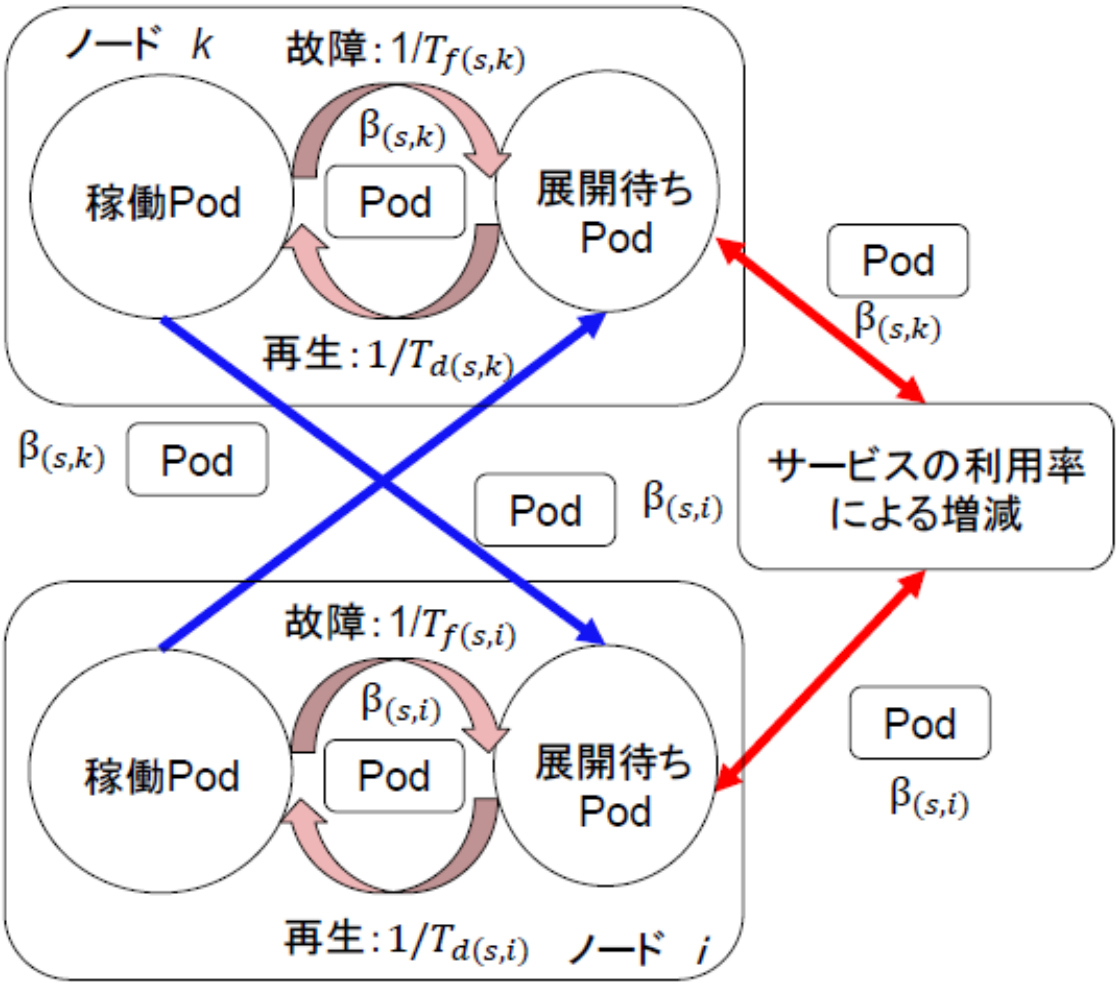

図 4.14　Pod のノード間での動き

4.6.2 近似解法

定常状態において $P_{(s,n)}$ で稼働している Pod 数（平均稼働 Pod 数）を未知パラメータ $\xi_{(s,n)}$ として導入する．$1/T_{f(s,n)}$ が障害発生率を表すため， $P_{(s,n)}$ の障害発生数は $\xi_{(s,n)} \times 1/T_{f(s,n)}$ となる．また，入力，出力，障害発生がランダムに発生すると仮定し，このモデルをポアソン過程と考える．

各ノード n への request が分配される確率を $\alpha_{(s,n)}$ と定義すると，サービス s，ノード n の平均利用率 $\rho_{(s,n)}$ は，以下のように表すことができる．

$$\rho_{(s,n)} = \frac{\alpha_{(s,n)}\,\lambda_{(s)}}{\xi_{(s,n)}\,\mu_{(s,n)}} \tag{4.21}$$

ただし，$\xi_{(s,n)}\,\mu_{(s,n)} = 0$ の場合は，$\rho_{(s,n)} = 0$ とする．

サービス s の各ノード n への Pod が展開される確率 $\beta_{(s,n)}$ を以下のように定義する．

$$\beta_{(s,n)} = \frac{\rho_{(s,n)}}{\sum_{l=1}^{N} \rho_{(s,l)}} \tag{4.22}$$

この定義は，平均利用率が高いノードに Pod が展開されることを意味する．また，サービス s に対する平均利用率 $\gamma_{(s)}$ を以下のように定義する．

$$\gamma_{(s)} = \frac{\sum_{n=1}^{N} \rho_{(s,n)}}{N} \tag{4.23}$$

各ノードの展開待ち Pod 数の分布

サービス s に対するノード k の Pod に障害が発生し，ノード i に新しい Pod が展開される単位時間あたりの個数 $\delta_{(s,k,i)}$ は，次のとおり定義する．

$$\delta_{(s,k,i)} = \frac{\xi_{(s,k)}\,\beta_{(s,i)}}{T_{f(s,k)}} \tag{4.24}$$

また，ノード k で障害が発生した後，平均展開時間 $T_{d(s,i)}$ の間にノード i で展開を待つ Pod 数の期待値 $E_{f(s,i)}$ は，次のようになる．

$$E_{f(s,i)} = \sum_{k=1}^{N} \delta_{(s,k,i)}\,T_{d(s,i)} \tag{4.25}$$

一方，Pod 数の調整において $|P_{(s,i)}|$ の単位時間当たりの増減数 $\epsilon_{(s,i)}$ は，以下のように定義する．

$$\epsilon_{(s,i)} = \begin{cases} \dfrac{\beta_{(s,i)}}{T_{o(s)}} & (\text{if } \gamma_{(s)} \geq \phi_u(s) \text{ or } \gamma_{(s)} \leq \phi_d(s)) \\ 0 & (\text{else}) \end{cases} \tag{4.26}$$

平均展開時間 $T_{d(s,i)}$ の間に $P_{(s,i)}$ の展開を待つ Pod 数の増減数の期待値 $E_{d(s,i)}$ は，以下のとおりである．

$$E_{d(s,i)} = \begin{cases} \epsilon_{(s,i)}\, T_{d(s,i)} & (\gamma_{(s)} \geq \phi_u(s)) \\ -\epsilon_{(s,i)}\, T_{d(s,i)} & (\gamma_{(s)} \leq \phi_d(s)) \end{cases} \tag{4.27}$$

式 (4.25) と式 (4.27) から，サービス s に対してノード i で展開待ちの Pod 数の期待値 $E_{a\ (s,i)}$ は，以下のように定義される．

$$E_{a(s,i)} = E_{f(s,i)} + E_{d(s,i)} \tag{4.28}$$

したがって，ノード i でサービス s の展開を待つ Pod 数が w である確率分布 $Pr_{(s,i)(w)}$ は，ポアソン過程とみなし，次式で表される．

$$Pr_{(s,i)(w)} = \frac{(E_{a(s,i)})^w \exp(-E_{a(s,i)})}{w!} \tag{4.29}$$

各ノードの Pod の故障数の分布

ノード i のサービス s の Pod の故障により，展開待ちとなる Pod 数の期待値 $\hat{E}_{f(s,i)}$ は，式 (4.25) と同様に以下となる．

$$\hat{E}_{f(s,i)} = \sum_{k=1}^{N} \delta_{(s,i,k)} T_{d(s,i)} \tag{4.30}$$

以上，ノード i のサービス s の Pod の故障により，展開待ちとなる数が m である確率 $\hat{Pr}_{(s,i)(m)}$ は次式で表される．

$$\hat{Pr}_{(s,i)(m)} = \frac{(\hat{E}_{f(s,i)})^m \exp(-\hat{E}_{f(s,i)})}{m!} \tag{4.31}$$

各ノードの稼働 Pod 数の再計算

定常状態のパラメータを定点とみなし，式 (4.29) と 式 (4.31) から，サービス s , ノード i の 新たな平均稼働 Pod 数 $\xi^*_{(s,i)}$ は，以下となる．

$$\xi^*_{(s,i)} = \xi_{(s,i)} + \sum_{w=0}\sum_{m=0}(w-m)Pr_{(s,i)(w)}\hat{Pr}_{(s,i)(m)} \tag{4.32}$$

式 (4.32) を用いて，4.4.1 節に記述した近似解法概要の (4) において，近似解となる平均稼働 Pod 数 $\xi^*_{(s,i)}$ を具体的に求める方法は，以下である．

$\xi_{(s,i)}$ と $\xi^*_{(s,i)}$ の差がすべてのサービス s 及びノード i で式 (4.33) を満たせば，$\xi^*_{(s,i)}$ を収束解とする．また，本章の数値計算では，$\tau = 0.01$ としている．

$$|\xi_{(s,i)} - \xi^*_{(s,i)}| < \tau \tag{4.33}$$

ただし，$\tau(> 0)$ は十分に小さな値とする．

満たさない場合は，緩和法を用いて式 (4.34) で $\xi_{(s,i)}$ を修正し，4.6.1 の近似解法概要の (1) から再計算する．

$$\xi_{(s,i)} = (1-\omega)\xi_{(s,i)} + \omega\xi^*_{(s,i)} \tag{4.34}$$

また，本章の数値計算では，ω =1/2 としている．ただし，ω は $0 < \omega < 1$ の適当な値とする．

4.6.3　近似解法の表記

近似解法で使用される表記をまとめると以下のようになる.

$\xi_{(s,n)}$　$P_{(s,n)}$ の平均稼働 Pod 数

$\rho_{(s,n)}$　サービス s，ノード n に対する利用率

$\delta_{(s,k,j)}$　$^{\forall}p \in P_{(s,k)}$ で故障が発生し，$P_{(s,j)}$ で待機となる事象の発生率

$\epsilon_{(s,k)}$　$|P_{(s,k)}|$ が 1 つ増減する事象の発生率

$\mathrm{E}_{f(s,i)}$　サービス s，ノード i で故障が発生したことによる $T_{d(s,i)}$ の展開待ち数の期待値

$\mathrm{E}_{d(s,i)}$　$P_{(s,i)}$ が増減する事象の発生よる $T_{d(s,i)}$ で展開待ち数の期待値

$\mathrm{E}_{a(s,i)}$　サービス s，ノード i の展開待ち数の期待値

$\hat{\mathrm{E}}_{f(s,i)}$　$^{\forall}p \in P_{(s,i)}$ で故障により，ノード i で展開待ち数の期待値

$\mathrm{P}_{(s,i)(w)}$　$P_{(s,i)}$ の展開待ち数が w である確率

$\hat{\mathrm{P}}\mathrm{r}_{(s,i)(m)}$　$^{\forall}p \in P_{(s,i)}$ が故障により展開待ち数が m である確率

4.7　マルチベースの数値計算

本節では，4.5 節で示した近似解法を用いた数値計算及びモンテカルロシミュレーションの数値計算結果を比較する．そして，本章で提案した近似解法の近似精度について検証する．また，表 4.1 に使用したパラメータ及び各種数値計算の実施結果を示す．

パラメータの選定は，既存の研究で使用された値や想定する現実のモデルを考慮した数値を利用した．最も注意する点は，各ベースにおける平均修理時間に対してベース間のアイテムの平均移送時間が十分に小さくなければならないことである．もし，平均修理時間と平均移送時間の間に差がない，もしくは上回った場合，ベース間補給を考える意味がなくなる．近似解法とシミュレーションの相対誤差は % 表示とする．各ベース間の平均移送時間は，t_{ij}=1 とした

なお，シミュレーションは，各ケースについて終了時間を 100 万時間としたイベント法を実施している．

表 4.1　パラメータ及び各種数値計算の実施結果

case1

base	n_i	λ_i	μ_i	s_i	c_i	システム稼働率		相対誤差 (%)
						simulation	近似解	
1	15	0.01	0.17	1	1	0.952	0.959	0.709
2	12	0.01	0.13	2	1	0.941	0.946	0.521
3	10	0.01	0.12	3	1	0.942	0.945	0.293
4	8	0.01	0.09	1	1	0.928	0.927	0.048
5	7	0.01	0.08	1	1	0.923	0.920	0.256

case2

base	n_i	λ_i	μ_i	s_i	c_i	システム稼働率		相対誤差 (%)
						simulation	近似解	
1	30	0.01	0.31	2	1	0.946	0.951	0.552
2	25	0.01	0.26	1	1	0.942	0.946	0.491
3	20	0.01	0.21	2	1	0.933	0.935	0.219
4	15	0.01	0.16	1	1	0.923	0.922	0.104
5	10	0.01	0.12	1	1	0.916	0.910	0.597

case3

base	n_i	λ_i	μ_i	s_i	c_i	システム稼働率		相対誤差 (%)
						simulation	近似解	
1	90	0.005	0.12	4	4	0.978	0.975	0.303
2	80	0.005	0.15	3	3	0.979	0.976	0.323
3	90	0.005	0.11	4	4	0.973	0.969	0.405
4	70	0.005	0.18	4	2	0.972	0.966	0.520
5	75	0.005	0.18	4	2	0.968	0.962	0.630
6	70	0.005	0.18	4	2	0.972	0.966	0.521
7	85	0.005	0.15	3	3	0.977	0.974	0.353
8	85	0.005	0.14	3	3	0.972	0.968	0.426

表 4.1 パラメータ及び各種数値計算の実施結果の続き

case4

base	n_i	λ_i	μ_i	s_i	c_i	システム稼働率		相対誤差 (%)
						simulation	近似解	
1	70	0.01	0.25	4	3	0.970	0.968	0.235
2	65	0.01	0.22	3	3	0.965	0.962	0.310
3	60	0.01	0.21	4	3	0.965	0.962	0.352
4	55	0.01	0.19	3	3	0.962	0.958	0.423
5	50	0.01	0.17	3	3	0.959	0.953	0.557

case5

base	n_i	λ_i	μ_i	s_i	c_i	システム稼働率		相対誤差 (%)
						simulation	近似解	
1	50	0.01	0.17	3	3	0.969	0.964	0.537
2	30	0.01	0.16	3	2	0.958	0.949	0.913
3	50	0.01	0.18	3	3	0.972	0.968	0.470
4	40	0.01	0.14	3	3	0.966	0.959	0.716
5	30	0.01	0.11	3	3	0.962	0.952	0.982
6	30	0.01	0.16	3	2	0.958	0.949	0.956
7	50	0.01	0.26	3	2	0.970	0.965	0.510

case6

base	n_i	λ_i	μ_i	s_i	c_i	システム稼働率		相対誤差 (%)
						simulation	近似解	
1	54	0.01	0.18	3	3	0.958	0.954	0.380
2	52	0.01	0.17	3	3	0.954	0.950	0.444
3	50	0.01	0.26	2	2	0.959	0.956	0.396
4	48	0.01	0.25	2	2	0.958	0.954	0.426
5	46	0.01	0.24	3	2	0.957	0.953	0.464
6	44	0.01	0.23	2	2	0.956	0.951	0.488
7	42	0.01	0.22	2	2	0.955	0.950	0.545
8	40	0.01	0.21	3	2	0.954	0.948	0.607

表 4.2　パラメータの範囲 (4.8.1)

Parameter	Variable Values
N	5
λ	1 ~ 20
μ	0.1 ~ 4.0
ϕ_u	0.30
ϕ_d	0.28
T_o	60
T_f	1500 ~ 5000
T_d	100 ~ 300
α_n	0.1 ~ 0.4

4.8　マルチクラウドの数値計算

本節では，4.6 節で示した近似解法と 4.4 節で示したモデルの仮定に従うモンテカルロシミュレーションによる数値計算を実施する．数値計算結果を比較することで，近似精度の評価や各方法での計算時間について考察する．また，近似解法を使用して，KaaS でサービスを展開しているサービス提供者が必要とするネットワーク資源の推定を行い，必要となる資源予測・資源管理における有用性を検証する．なお，数値計算には，Google 社の提供する Colaboratory[49] の Python を使用した．

4.8.1　近似精度の計算時間の検証

本項では，想定するマルチクラウドの条件に沿ったモデルの仮定を定義するために，4.4 節に示したモデルの仮定に従った，モンテカルロシミュレーションによる数値計算を実施する．また，Pod のノード間の移動（β）や Pod の増減を判断するサービスの平均利用率 (γ) に関しては，4.6 節の定義式 (4.21) ，(4.22)，(4.23) に従う．このことを考慮したモンテカルロシミュレーションと近似解法の数値計算の結果を比較することで，近似解法の精度を評価する．

表 4.3　データセットの例

N	λ	ϕ_u	ϕ_d	T_o
5	5	0.30	0.28	60.0
case 1				
Node ID	μ	T_f	T_d	α_n
1	0.1	3000	100	0.1
2	0.4	5000	120	0.1
3	0.6	3000	150	0.2
4	1.0	2000	200	0.4
5	0.2	1500	300	0.2
case 2				
Node ID	μ	T_f	T_d	α_n
1	0.1	3000	100	0.2
2	0.4	5000	120	0.2
3	0.6	3000	150	0.2
4	1.0	2000	200	0.2
5	0.2	1500	300	0.2

パラメータの選定

本節の数値計算に使用するパラメータの選定にあたっては，想定するネットワークを考慮した 10 個のデータセットを表 4.2 のパラメータの範囲から選んだ．表 4.3 にデータセットの例を示す．

近似解法の近似精度を確認するため，モンテカルロシミュレーションの回数を一つのデータセットに対して 10 回実施した．1 回のモンテカルロシミュレーションは，最大 200 万時間のイベント法で実施している．また，モンテカルロシミュレーションの開始から平均稼働 Pod 数を観測し，一定期間（2000 時間）での変化を計測した．一定期間での変化が近似解法の計算の終了条件と同じ，全てのノードでの平均稼働 Pod 数の誤差が 0.01 以下となった場合は，シミュレーションを停止して，その値を結果とした．

近似精度検証

表 4.4 は表 4.3 のパラメータを使用した計算結果となる．各ノードの列が平均稼働 Pod 数，最終列が計算時間である．モンテカルロシミュレーションについては，複数回しているため, 平均値，最大値, 最小値を記載している．

表 4.4 からすべてのデータセットにおいて，モンテカルロシミュレーションの結果の範囲内に近似解法の結果が収まっていることがわかる．残りの 8 個のデータセットにおいても，同様の数値計算を実施し，データセット 10（5 つのノード）の合計 50 個中 48 個でモンテカルロシミュレーションの結果の範囲内に近似解法の結果が収まった (図 4.15〜 4.19)．また，近似解法とモンテカルロシミュレーションの平計算時間は，近似解法が 0.49s，モンテカルロシミュレーションが 161.89s となり，近似解法がシミュレーションに比べ，100 倍以上速い結果となった（表 4.5）．

表 4.4　表 4.3 のデータセットでの結果

case1

		Node					Computing Time(s)
		1	***2***	***3***	***4***	***5***	
Approximate solution		13.79	8.75	8.09	7.19	9.81	0.39
Monte Carlo	Mean.	13.94	9.34	8.29	7.38	9.91	219.05
	Min.	13.41	8.41	7.83	7.04	9.49	96.84
	Max.	14.83	10.23	8.90	7.90	10.43	336.67

case 2

		Node					Computing Time(s)
		1	***2***	***3***	***4***	***5***	
Approximate solution		21.45	13.67	8.80	5.54	10.71	0.54
Monte Carlo	Mean.	21.60	14.35	9.25	5.97	11.19	187.10
	Min.	20.37	13.62	8.69	5.53	10.51	93.16
	Max.	22.29	14.91	9.61	6.27	11.57	236.70

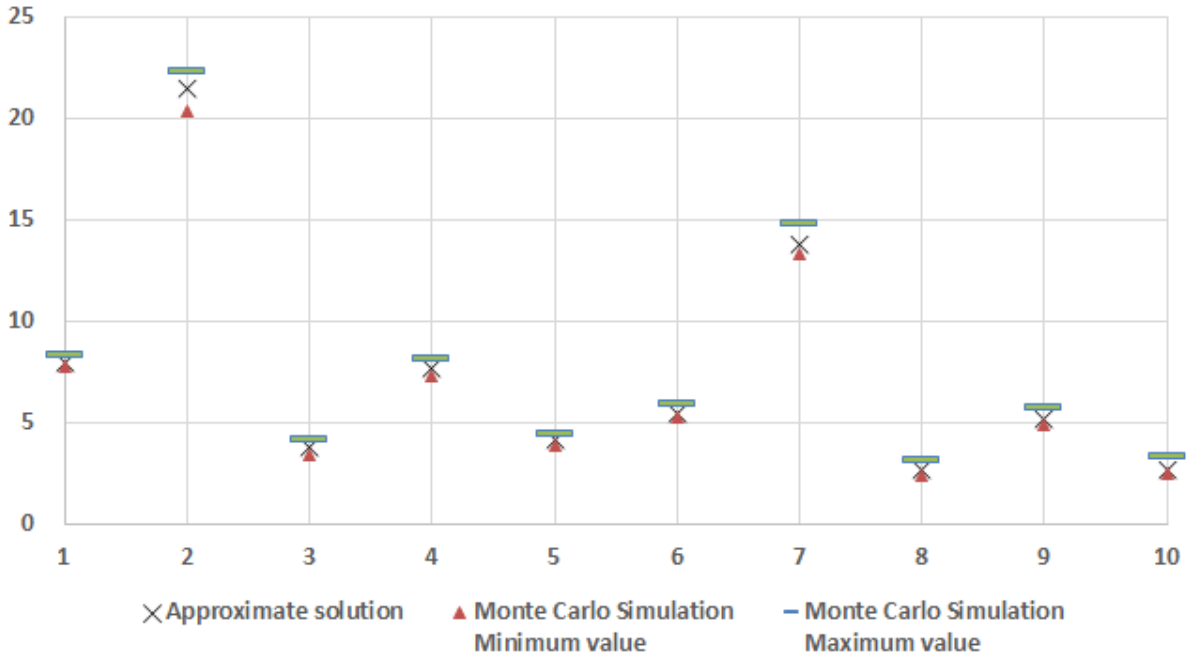

図 4.15　ノード 1 の平均稼働 Pod 数

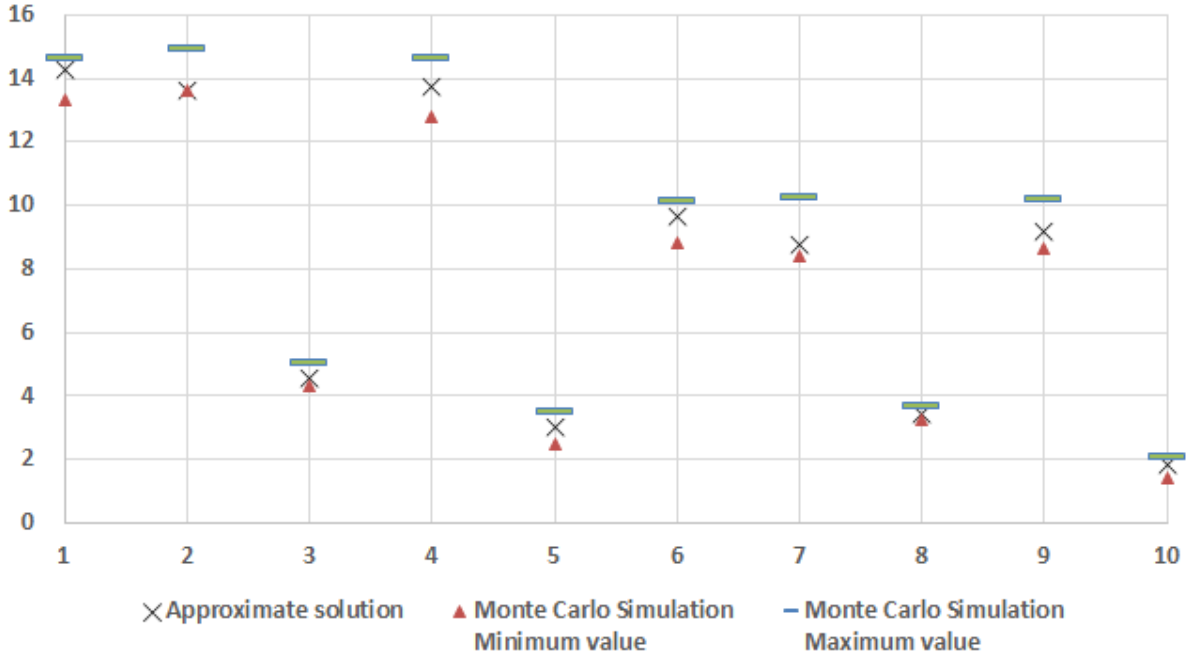

図 4.16　ノード 2 の平均稼働 Pod 数

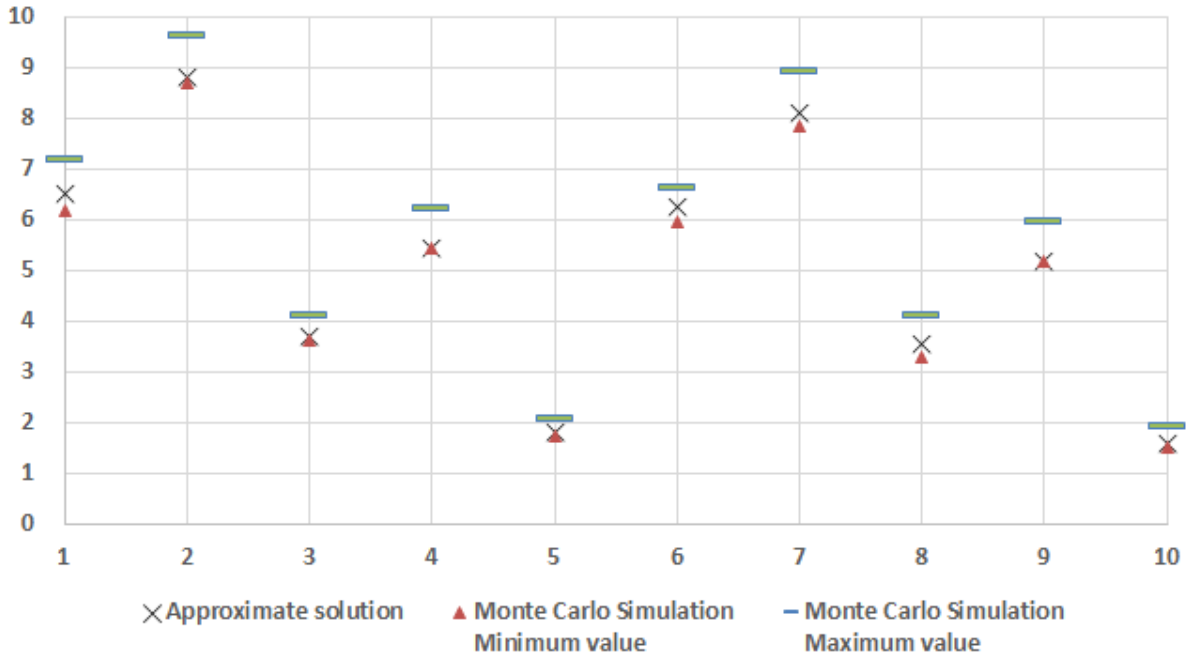

図 4.17　ノード 3 の平均稼働 Pod 数

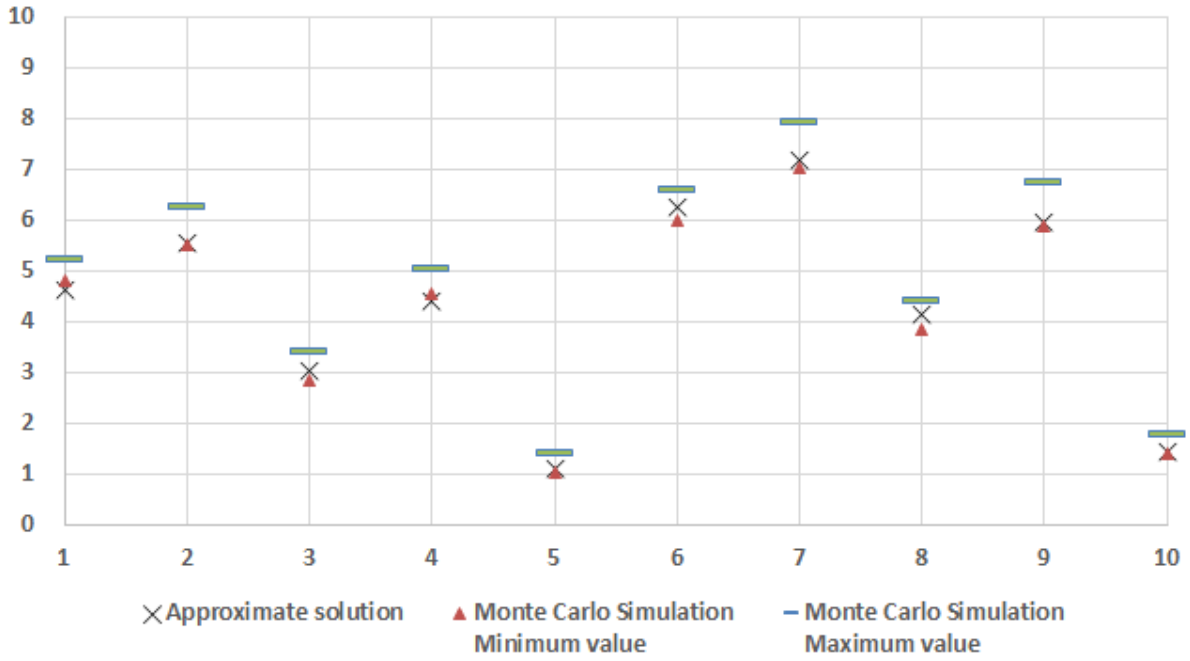

図 4.18　ノード 4 の平均稼働 Pod 数

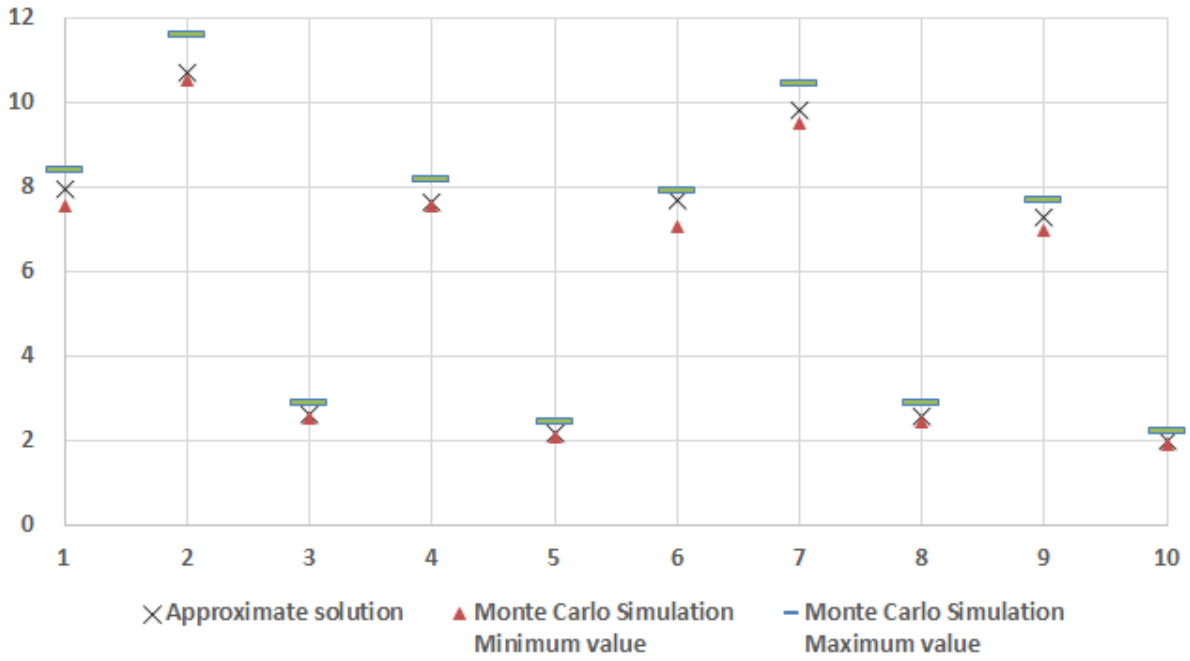

図 4.19　ノード 5 の平均稼働 Pod 数

表 4.5　平均計算時間 (sec.)

computation time for 10 data sets	Approximate solution	Monte Carlo Simulation
Mean	0.49	161.89
Min	0.23	12.88
Max	0.75	301.88

表 4.6　パラメータの範囲 (4.8.2)

Parameter	Variable Values
N	25
λ	100 ~ 1000
μ	1.0 ~ 4.0
ϕ_u	0.30
ϕ_d	0.28
T_o	60
T_f	3000 ~ 5000
T_d	150 ~ 200
α_n	0.04

4.8.2　KaaS への適用例

KaaS の変数推定

文献 [50] では，ヤフー（株）で運用する KaaS について，設計と運用実績から得た知見が述べられている．同社は，多数のコンテナ化されたアプリケーションを運用管理するため，Kubernetes[18] をプライベートクラウド向けにサービス化した Kubernetes as a Service（KaaS）として，2018 年 8 月から運用を開始している．

この KaaS では，各 Web サービスを開発・運用する部門ごとに独立した Kubernetes[18] が提供されている．同社の KaaS の運用実績は，2020 年 12 月時点で，コンテナ数約 204,980 個，ノード数約 27,900VM，約 860 クラスタの Kubernetes[18] がある．また，1 ノードあたりに稼働しているコンテナ数の平均値は，約 7.35 コンテナ／ノードとなっており，各ノード種別と etcd の VM 数の比率は，それぞれ Control Plane 7.2%，Worker 70.6%，Ingress 10.7%，etcd 11.5% であることが述べられている．

以上のことから，1 つの Pod に 1 つのコンテナがあると仮定すると，1 つのサービスあたり平均 25 ノード程度で運用されていると推測され，各ノードに平均が 7.35 程度のコンテナが存在するようなデータセットを作成した．表 4.6 は，本項で使用する

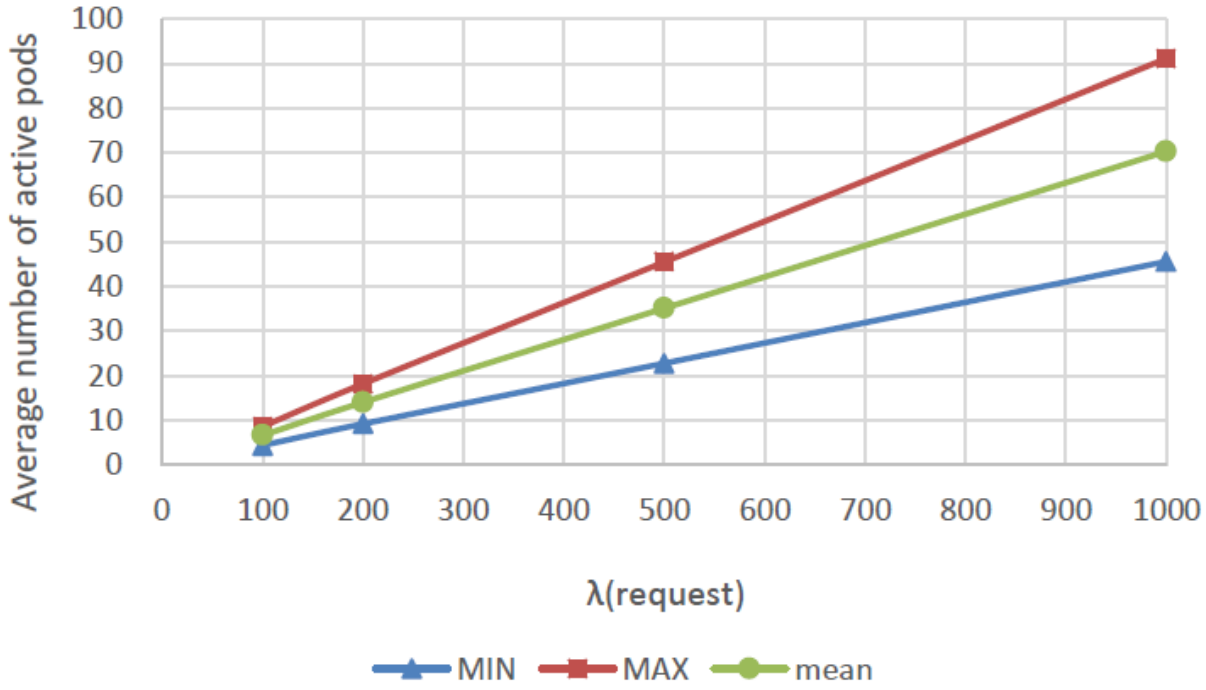

図 4.20　request の量の違いによる推移

データセットである．そして，サービスの利用量が当初の想定を超えた場合の検証を行った．具体的には request（λ）の量を 2 倍，5 倍，10 倍として数値計算を行った．

計算結果からの考察

図 4.20 に request の量を変化させた場合の各ノードの平均と最大，最小稼働 Pod 数の結果となる．計算結果から，request の量が当初の 10 倍を超えてくると，あるノードで平均稼働 Pod 数が 91.14 となった．

このノードの待機状態の Pod 数と平均稼働 Pod 数を組み合わせることで，このノードに必要な Pod 数を確率的に表したものが，図 4.21〜図 4.24 である．この結果から，このノードでは，想定の 10 倍を超える request が発生した場合，一時的に Pod 数が 110 を超えることがあることがわかる．Kubernetes [18] は，一つのノードにある Pod 数を 110 以下にするように推奨している [51]．すなわち，想定の 10 倍を超える request が発生した場合，稼働数が多いノードのスペック強化，配分確率αやβの変更，ノードの増設による処理の分散などの対策を実施する必要がある．

最後に，Pod のスケジューリング（β）に関するアルゴリズムについては，様々な研究があり，そのアルゴリズムを本章が提案する近似解法に組み込むことができれば，その効果を検証することが可能と考える．これについては，今後の課題とする．

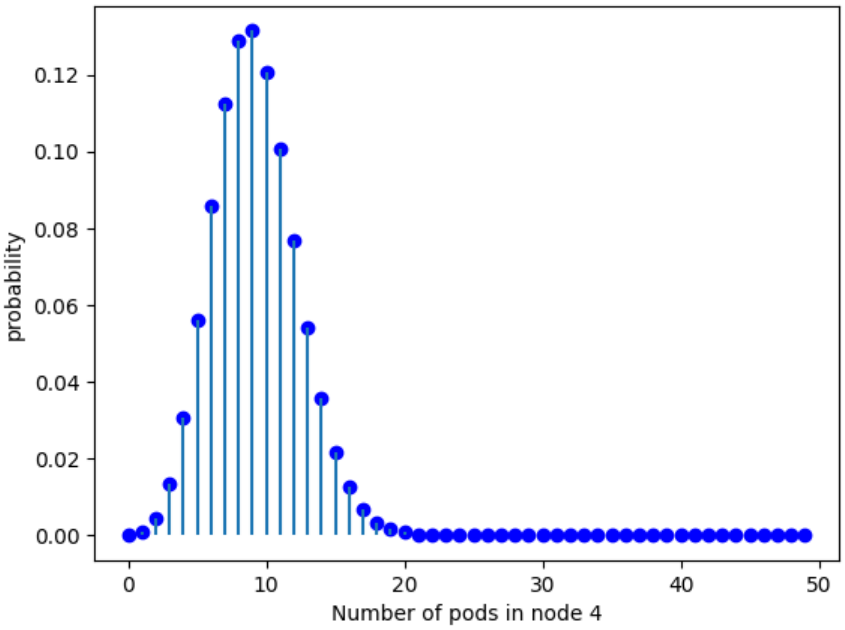

図 4.21　ノード内のポッド数の分布

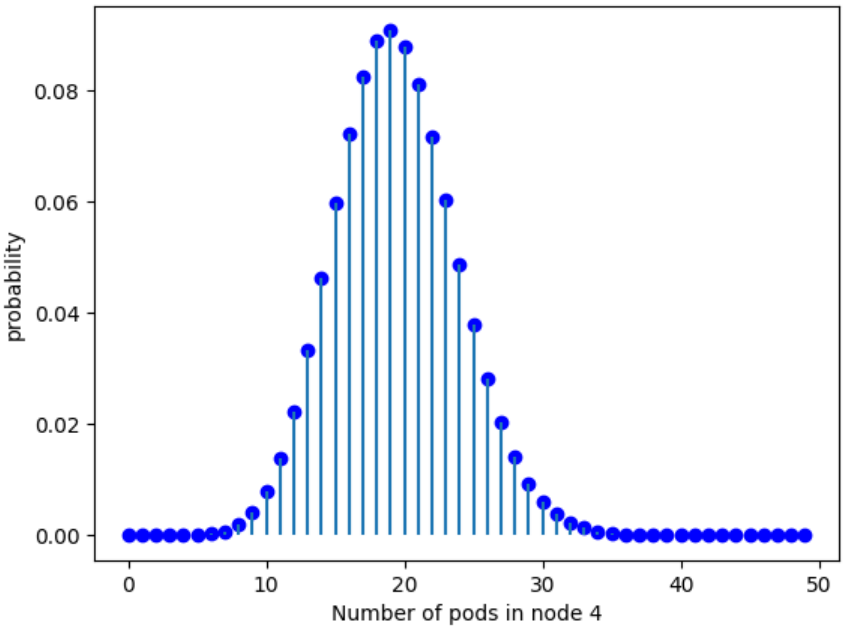

図 4.22　ノード内のポッド数の分布 (2 倍)

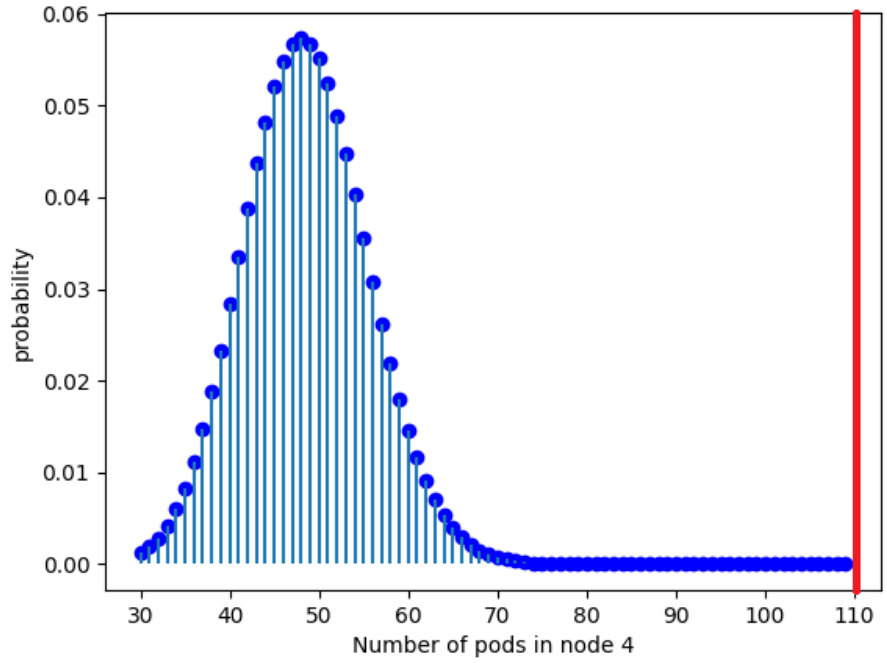

図 4.23　ノード内のポッド数の分布（5 倍）

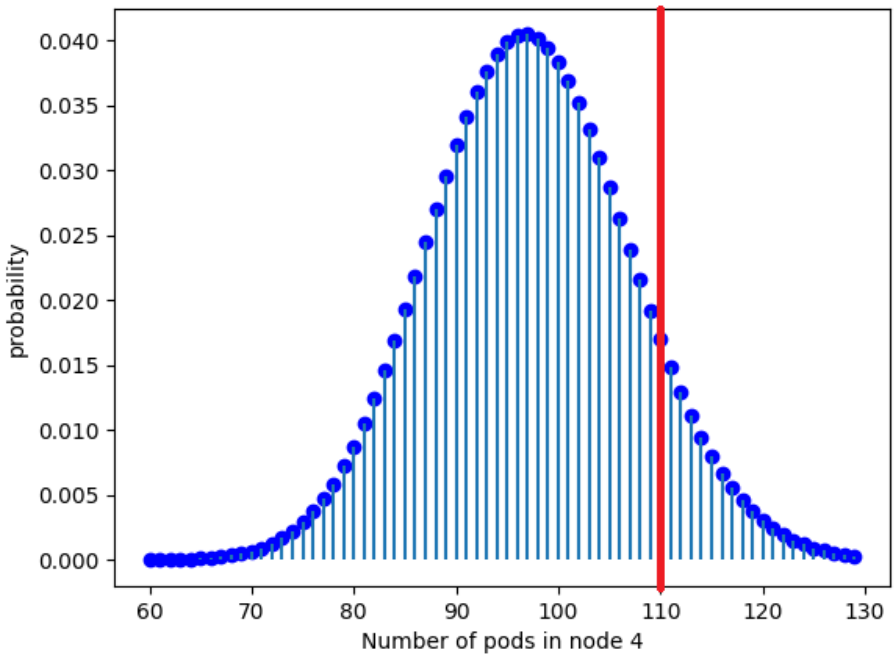

図 4.24　ノード内のポッド数の分布 (10 倍)

4.9　まとめ

本章の目的は，オーケストレーションツールで制御されたマルチクラウドを利用して，サービスを展開しているサービス提供者が必要とするネットワーク資源をオーケストレーションツールで運用されることを前提に，Pod の故障，再生及び，増減を加味した予測を行うことである．この前提は，先行研究が解析しているものと異なる．

本章で想定するマルチクラウドの数理モデルは，2.3 節で説明した補給整備システムの解析手法に基づく．具体的には，マルチベースとマルチクラウドの運用方式が類似しているところに着目し，マルチベースの数理モデルを用いてマルチクラウドの数理モデルを提案した．

そして，本章が提案する近似解法及び，モンテカルロシミュレーションでの数値計算を行った．計算結果を比較することにより，近似精度を検証した．本論文で提案した近似解法は，計算時間が短く，モンテカルロシミュレーションの結果ともよく一致しており，計算も容易であった．

加えて，実際に，サービスを提供する事業者の運用実績から推定したパラメータで，本章の数理モデルを使用した計算結果から，ネットワーク資源の予測に関しての有用性を検証した．

第 5 章

結論

安全かつ安定したクラウドサービスを提供するための技術開発が進んでおり，クラウドを支える仮想化技術においても，セキュリティ対策，ネットワーク資源配分，障害復旧，サービス品質など多様な観点で研究開発が必要である．本稿では，安定なクラウドサービスを提供するために，サービス妨害対策とサービスの信頼性に焦点をあて，数理モデルを提案し，ネットワーク資源配分の予測を行った．具体的には，以下のような結果となった．

第３章では，バッファリングノードで形成するオーバーレイネットワークのトポロジのタイプやバッファ容量等の偏りのあるトポロジやバッファ容量のばらつきがある場合にも，より安定した性能を確保することを目的として，ノード間のパケットの転送を制御している転送レート算出式を改良し，より長時間パケット損失を防ぐことを可能とした．また，シミュレーションにより既存方式と提案方式の緩和性能を評価した．実験の結果，本稿の提案方式が既存方式に対して，平均緩和時間がすべて上回る結果となった．加えて，バッファ容量等のパラメータを複雑に設定したものほど，本稿の提案方式が既存方式に対して，緩和時間を引き延ばすことも確認した．

これらのことから，本稿の提案方式がトポロジや複雑なパラメータの設定に対して，柔軟に拡散効果を発揮し，バッファあふれによる緩和時間が延長されたと考えられる．ただし，トポロジ及びバッファ容量等のパラメータの設定によっては，本稿の提案方式が既存方式に対して下回る結果も観測されており，これら要因について解明できなかった．今後，パラメータとトポロジとの関係を類型化することでその原因を究明し，多様なネットワーク状況に対して性能を維持できる方式を検討する予定である．

また，DDoS 攻撃の規模は年々増加傾向にあり，DDoS 攻撃の攻撃規模が増大した場合について，本システムへの影響を考察した．結果として，攻撃規模に応じた適切なネットワーク資源の配分を行うことで，攻撃規模が増大した場合でも，十分にミチゲーションの効果を発揮できる結果となった．

今後の展望としては，ミチゲーションの効果を最大限発揮するためのオーバーレイネットワークのトポロジ構築方法及びバッファ容量等の各種パラメータの設定方法の検討，DDoS 攻撃の検知，ミチゲーション及びフィルタリングルールの適用による攻撃の回避を包括したシステムの設計・評価がある．

第 4 章では，複数のクラウド，エッジ，オンプレミス環境からなり，Kubernetes などのオーケストレーションツールによって制御されているマルチクラウドを想定し，マルチクラウドでサービスを展開しているサービス提供者が必要とするネットワーク資源をオーケストレーションツールで運用されることを前提に，Pod の故障，再生及び，増減を加味した予測を行うための数理モデルを提案した．

具体的には，複数のベースから構成され，その各ベースで同一アイテムの運用，整備，保管が行われ，各ベース間でアイテムのやり取り (ベース間補給方式) を行うマルチベースを想定，反復法による 2 段階整備方式の稼働率解析法の概念を利用し，マルチベースに関する稼働率解析を可能とする近似解析法を提案した．また，マルチベースとマルチクラウドの運用方式が似ていることから，マルチベースの数理モデルを基にしてマルチクラウドの数理モデルを提案した．

そして，提案する近似解法及びモンテカルロシミュレーションでの数値計算を行った．本稿で提案した近似解法は，計算時間が短く，モンテカルロシミュレーションの結果ともよく一致しており，計算も容易であった．また，具体的なネットワークを想定し，本稿で示した近似解法の計算結果からそのネットワークに必要となる資源予測・管理における有用性を検証した．

今後の展望としては，モデルの仮定や近似解法の修正，Pod のスケジューリングのアルゴリズムの取り込みによる計算機資源の資源効率の検証などがある．

最後に，ネットワーク資源配分の予測では，様々な解析方法が考えられるが，どの方法を選択するかは，時間と精度のトレードオフとなる．今後，これらを予測するための高度な数理モデルの研究が進み，より複雑なモデルに対して，短い計算時間で精度の高い予測ができることを期待する．

謝辞

本稿を進めるにあたり終始あたたかいご指導と激励を賜りました広島市立大学情報科学研究科　前田 香織特任教授に心から感謝の意を表します．広島市立大学情報科学研究科　高野 知佐教授には研究あるいは授業科目に関して多大なるご指導をいただきました．深く感謝いたします．広島市立大学情報科学研究科　市原 英行教授には，研究に向かう姿勢や研究に関する困難克服のための具体的な方策までていねいに教えていただきました．心からお礼申し上げます．

防衛大学校理工学研究科 弓削 哲史教授には，博士論文の作成にあたり，親身になって相談にのっていただきました．深くお礼申し上げます．本稿の研究の礎を築いていただきました，防衛大学校理工学研究科でご指導いただいた柳先生，田村先生にも深く感謝申し上げます．

広島市立大学情報科学研究科ネットワーク研究室のみなさんにも，研究の過程で終始協力いただいたことに深く感謝いたします．勤務している広島県ＤＸ審議官の皆様には，社会人学生として学術研究の道に進むことを許してくださり，あたたかく応援していただいたいことに，深く感謝いたします．

最後に，これまで私をあたたかく応援してくれた家族に心から感謝します．

参考文献

[1] AWS Shieled, ”Treat Landscape Report-Q1 2020,” https://aws-shield-tlr.s3.amazonaws.com/2020-Q1_AWS_Shield_TLR.pdf

[2] Kaspersky DDoS REPORTS, ”DDoS attacks in Q1 2020,” https://securelist.com/ddos-attacks-in-q1-2020/96837/ , May.2020.

[3] CyberNewsFlash,”DDoS 攻撃を示唆して仮想通貨による送金を要求する脅迫行為 (DDoS 脅迫) について,” JPCERT/CC. https://www.jpcert.or.jp/newsflash/2020090701.html

[4] 山田洋之, 久保田光一, ”IDS を用いた DDoS 攻撃の検知,”情報処理学会第 78 回全国大会, pp.555-556, Mar.2016.

[5] 太田悟, 田島伸一, 佐藤信, 長野純一, 篠宮紀彦, 勅使河原可海, ”penFlow を用いた DDoS 攻撃検知システムの検討,” 情報処理学会第 75 回全国大会, pp.543-544, Mar.2013.

[6] Y. Chen, K. Hwang, and W. S. Ku, ”Collaborative Detection of DDoS Attacks over Multiple Network Domains,” IEEE Transactions on Parallel and Distributed Systems, vol.18, no.12, pp.1649–1662, Nov.2007.

[7] X. Wang, Ming Li and Muhai Li, ”A Scheme of Distributed Hop-count Filtering of Traffic,” IET International Communication Conference on Wireless Mobile and Computing (CCWMC), pp.516-521, Jul.2009.

[8] 増田絢斗, 今堀慎治, 小原泰弘, ”DDoS ミティゲーションを可能にするネットワークルーティングアルゴリズム,” 電子情報通信学会 信学技報, vol.118, no.268, pp.33-39, Oct.2018.

[9] R. Abubakar, A. Aldegheishem, M. F. Majeed, C. Maple, N. A. Alrajeh, H. Maryam, M. Jawad, ”An Effective Mechanism to Mitigate Real-Time DDoS At-

tack," IEEE Access, vol.8, pp.126215-126227, May.2020.

[10] M. P. Novaes, L. F. Carvalho, J. Lloret, M. L. Proença, "Long Short-Term Memory and Fuzzy Logic for Anomaly Detection and Mitigation in Software-Defined Network Environment," IEEE Access, vol.8, pp.83765-83781, May.2020.

[11] 平空也，高野知佐，前田香織, "拡散型フロー制御を用いる DDoS 攻撃緩和システムの提案," 電子情報通信学会, 信学技報, vol.117, no.173, pp.51-56, Aug.2017.

[12] 平空也，高野知佐，前田香織, "拡散型フロー制御を用いる DDoS 攻撃緩和システムの評価," 電子情報通信学会, 信学技報, vol.117, no.294, pp.1-6, Nov.2017.

[13] 平空也，高野知佐，前田香織, "拡散型フロー制御を用いる DDoS 攻撃緩和システム," 情報処理学会論文誌, vol.59, no.9, pp.1656-1665, Sep.2018.

[14] M. Lazuka, T. Parnell, A. Anghel, and H. Pozidis, "Search-Based Methods for Multi-Cloud Configuration," IEEE 15th International Conference on Cloud Computing, pp.438-448, Apr.2022.

[15] P. Farzin, S. Azizi,T. Parnell,M. Shojafar,O. Rana, and M. Singhal, "FLEX: A Platform for Scalable Service Placement in Multi-Fog and Multi-Cloud Environments," In Australasian Computer Science Week 2022, pp.106-114, Feb.2022.

[16] U. Jambulingam, and K. Balasubadra, "A Unique Multi-Agent-Based Approach for Enhanced QoS Resource Allocation in Multi Cloud Environment while Maintaining Minimized Energy and Maximize Revenue," International Journal of Computers Communications, vol.17, no.2, Mar.2022.

[17] C. Carrión, "Kubernetes Scheduling: Taxonomy, ongoing issues and challenges," ACM Computing Surveys, vol.55, no.138, pp.1–37, May.2022.

[18] Kubernetes, "Production-Grade Container Orchestration," `http://kubernetes.io/`, Aug.2022.

[19] Z. Rejiba, and J. Chamanara, "Custom scheduling in Kubernetes: A survey on common problems and solution approaches,"ACM Computing Surveys, vol.55, no.151, pp.1–37, Dec.2022.

[20] I. Sfiligoi, T. DeFanti, and F. Würthwein, "Auto-scaling HTCondor pools using Kubernetes compute resource," Practice and Experience in Advanced Research Computing vol.57, pp.1-4, May.2022.

[21] Z. Wei-guo, M. Xi-lin, Z. Jin-zhong, "Research on Kubernetes' Resource Scheduling Scheme," Proceedings of the 8th International Conference on Communication

and Network Security, pp.144–148, Nov.2018.

[22] 坪内佑樹, 青山真也, “マイクロサービスにおける異常検知・原因分析のためのデータセットの動的生成システム,” インターネットと運用技術シンポジウム, pp.63-70, Nov.2021.

[23] D. Chemodanov, P. Calyam, S. Valluripally, H. Trinh, J. Patman, and K. Palaniappan, ”On QoE-Oriented Cloud Service Orchestration for Application Providers,” IEEE TRANSACTIONS ON SERVICES COMPUTING, vol.14, no.4, Aug.2021.

[24] S. Ebneyousef, and S. Ghazanfari-Rad, ”Cloud Resource Demand Prediction to Achieve Efficient Resource Provisioning,” Iranian Conference on Signal Processing and Intelligent Systems, Dec.2022.

[25] N. Thanh Nguyen and Y. Kim, ”A Design of Resource Allocation Structure for Multi-Tenant Services in Kubernetes Cluster,” Asia Pacific Conference on Communications, pp.651-654, Oct.2022

[26] L. M. Al Qassem, T. Stouraitis, E. Damiani, and I. M. Elfadel, ”Optimal Resource Allocation for Containerized Cloud Microservices,” International Conference on Electrical and Computing Technologies and Applications, pp.271-274, Nov.2022.

[27] Y. Shi, I. Jyosthna Lingareddy, K. Suo, and T. N. Nguyen, ”Optimizing Resource Allocation in Cloud,” Electronics and Mobile Communication Conference, pp.1776-1785, Oct.2022.

[28] D. Saxena, J. Kumar, S. Member, A. Kumar Singh and S. Schmid, ”Performance Analysis of Machine Learning Centered Workload Prediction Models for Cloud,” IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, vol.34, no.4, Apr.2023.

[29] A. Rossi, A. Visentin, S. Prestwich and K. N. Brown, ”Bayesian Uncertainty Modelling for Cloud Workload Prediction,” International Conference on Cloud Computing, pp.19-29, Jul.2022.

[30] K. Kim, W. Wang, Yanjun Qi and M. Humphrey, ”Forecasting Cloud Application Workloads With CloudInsight for Predictive Resource Management,” IEEE Transactions on Cloud Computing, vol.10, pp.1848-1863, Jul.2022.

[31] 住達郎, 高野知佐, 会田雅樹, 石田賢治, ”拡散方程式に基づく自律分散的輻輳制御技術の実証実験,” 電子情報通信学会論文誌 D, vol.J95-D, no.12, pp.2048-2058, May.2012.

[32] ”Internet Exchange Point(IX; 相互接続点) とは,” JPNIC, https://www.nic.ad.jp/ja/basics/terms/ix.html Jan.2024.

[33] 青山真也, Kubernetes 完全ガイド第 2 版, 高橋隆志, 株式会社インプレス, 2020.

[34] C. C. Sherbrooke, “METRIC: A Multi-Echelon Technique for Recoverable Item Control,” Operations Research, vol.16, no.1, pp.122-141, Feb.1968.

[35] S. C. Albright, and A. Soni, “Markovian Multiechelon Repairable Inventory System,” Naval Research Logistics, vol.35, no.1, pp.49-61, Feb.1988.

[36] S. C. Albright, and A. Soni, “An Approximation to the Stationary Distribution of a Multiechelon Repairable Inventory System with Finite Source and Repair Channels,” Naval Research Logistics, vol.36, no.2, pp.179-195, Apr.1989.

[37] J. S. Kim, K. C. Shin, and S. K. Park, “An Optimal Algorithm for Repairable-item Inventory System with Depot Spares,” Operational Research, vol.51, no.3, pp.350-357, Mar.2000.

[38] 弓削哲史，川口礼人，柳繁,“反復法を用いた大規模 2 段階整備システムにおける稼働率解析,” 日本信頼性学会誌, vol.25 , no.2, pp.207-217, Mar.2003.

[39] N. Tamura, D. Muraoka, T. Yuge, and S. Yanagi, “Availability Analysis of a Two-Echelon Repair Model for Systems Comprising Multiple Items”, IEICE TRANS. FUNDAMENTALS, vol.E92-A, no.7, pp.1600-1607, Jul.2009.

[40] 奥田尚樹，田村信幸，弓削哲史，柳繁,“修理方式及び輸送を考慮した 2 段階整備方式の保全性解析,”電子情報通信学会技術研究報告, vol.109, no.67, pp.43-48, May.2009.

[41] S. Axsater, “Modelling Emergency Lateral Transshipment in Inventory Systems,” Management Science, vol.36, no.2, pp.1329-1338, Nov.1990.

[42] J. S. Kim, “Modeling lateral transshipments in multiechelon repairable-item inventory systems with finite repair channels,” Computers Operations Research, 30, pp.1401-1417, Aug.2003.

[43] H. Wong, D. Cattrysse, and D. Van Oudheusden, “Inventory Pooling of Repairable Spare Parts with Non-Zero Lateral Transshipment Time and Delayed Lateral Transshipments,” Eur. J. Oper. Res, vol.165, no.1, pp.207–218, Aug.2005.

[44] H. Wong, D. Cattrysse, and D. Van Oudheusden, “Stocking Decisions for Repairable Spare Parts Pooling in a Multi-hub System,” Int. J. Prod. Econ, vol.93-94, pp.309–317, Jan.2005.

[45] S. Axsater, “Evaluation of unidirectional lateral transshipments and substitutions in inventory systems,” Eur. J. Oper. Res, no.149, pp.438-447, Sep.2003.

[46] S. Yanagi, and M. Sasaki, “An Approximation Method for the Problem of a Repairable-Item Inventory System with Lateral Resupply,” IMA J. Management Math, vol.3, no.4, pp.305–314, Apr.1991.

[47] H. Wong, “Multi-item spare parts systems with lateral transshipments and waiting time constraints,” Eur. J. Oper. Res, no.171, pp.1071-1093, Jun.2006.

[48] Peering DB, https://www.peeringdb.com/

[49] Google Colaboratory, https://colab.research.google.com/

[50] 坂下幸徳, “マルチコンテナオーケストレーションを用いた大規模コンテナ環境の設計と運用会誌“, 情報処理, vol.62, no.8, Aug.2021.

[51] Kubernetes Community, ”Considerations for Large Clusters”, https://kubernetes.io/docs/setup/best-practices/cluster-large

研究業績一覧

[1] 奥田尚樹, 前田香織, 高野知佐, 市原英行, ”拡散型フロー制御を用いる DDoS 攻撃緩和方式の有効性評価,” 情報処理学会論文誌, vol.63, no.9, pp.1410–1418, Sep.2022.

[2] N. Okuda, N. Tamura, T. Yuge, and S. Yanagi, ”Availability Analysis of a Multibase System with Lateral Resupply between Bases,” IEICE Trans, vol.E98–A, no.10, pp.2084–2090, Oct.2015.

[3] N.Okuda, K.Maeda, C.Takano and H.Ichihara, ”A Resource Estimation Method in Multi-Cloud Envi ronment with a Model based on a Repairable-ItemInventory System,” COMPSAC, pp.1113-1120, Jun.2023.

[4] 奥田尚樹, 前田香織, 高野知佐, 市原英行, ”在庫管理システムの解析手法を用いたマルチクラウドの計算機資源の推定について,” 電子情報通信学会, Vol.J107-B, no.02, pp.63-71, Feb.2024.