

(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号
特許第7537528号
(P7537528)

(45)発行日 令和6年8月21日(2024.8.21)

(24)登録日 令和6年8月13日(2024.8.13)

(51)国際特許分類

G 0 6 F 9/50 (2006.01)

F I G 0 6 F 9/50 1 5 0 Z

請求項の数 8 (全21頁)

(21)出願番号	特願2022-581105(P2022-581105)	(73)特許権者	000004226 日本電信電話株式会社 東京都千代田区大手町一丁目5番1号
(86)(22)出願日	令和3年2月12日(2021.2.12)	(74)代理人	110001807 弁理士法人磯野国際特許商標事務所
(86)国際出願番号	PCT/JP2021/005175	(72)発明者	佐藤 諒平 東京都千代田区大手町一丁目5番1号 日本電信電話株式会社内
(87)国際公開番号	WO2022/172389	(72)発明者	中谷 裕一 東京都千代田区大手町一丁目5番1号 日本電信電話株式会社内
(87)国際公開日	令和4年8月18日(2022.8.18)	審査官	田中 幸雄
審査請求日	令和5年6月12日(2023.6.12)		

最終頁に続く

(54)【発明の名称】 リソース割当装置、リソース割当方法、および、リソース割当プログラム

(57)【特許請求の範囲】

【請求項1】

複数のユーザ端末から構成されるグループごとのステートをグループ内で共有させるためのステート管理部を、分散クラウド環境に配備された複数のデータセンタのうちのいずれかに割り当てる旨のリクエストを受信するリクエスト受信部と、
前記グループが要求する要件に応じた割当指標を用いて、前記ステート管理部を前記各データセンタに割り当てるときの割当コストを計算し、その割当コストに従って前記ステート管理部の割当先の前記データセンタを決定する割当計算部とを有することを特徴とするリソース割当装置。

【請求項2】

前記割当計算部は、グループを構成する前記各ユーザ端末と前記データセンタとの間の遅延時間を求め、それらの求めた遅延時間の平均遅延を前記ステート管理部の割当指標とすることを特徴とする
請求項1に記載のリソース割当装置。

【請求項3】

前記割当計算部は、グループを構成する前記各ユーザ端末と前記データセンタとの間の遅延時間を求め、それらの求めた遅延時間の最大遅延を前記ステート管理部の割当指標とすることを特徴とする
請求項1に記載のリソース割当装置。

【請求項4】

前記割当計算部は、グループを構成する前記各ユーザ端末と前記データセンタとの間の遅延時間を求め、それらの求めた遅延時間の分散または標準偏差を前記ステート管理部の割当指標とすることを特徴とする

請求項 1 に記載のリソース割当装置。

【請求項 5】

前記割当計算部は、前記リクエスト受信部が受信したリクエストの各グループの割当コストの総和を最小化する組み合わせ最適化問題に対して、貪欲法を用いた近似アルゴリズムにより、各グループの割当先を計算することを特徴とする

請求項 1 ないし請求項 4 のいずれか 1 項に記載のリソース割当装置。

【請求項 6】

前記割当計算部は、前記リクエスト受信部が受信したリクエストの各グループの割当コストの総和を最小化する組み合わせ最適化問題に対して、貪欲法と局所探索法とを併用する近似アルゴリズムにより、各グループの割当先を計算することを特徴とする

請求項 1 ないし請求項 4 のいずれか 1 項に記載のリソース割当装置。

【請求項 7】

リソース割当装置は、リクエスト受信部と、割当計算部とを有しており、

前記リクエスト受信部は、複数のユーザ端末から構成されるグループごとのステートをグループ内で共有させるためのステート管理部を、分散クラウド環境に配備された複数のデータセンタのうちのいずれかに割り当てる旨のリクエストを受信し、

前記割当計算部は、前記グループが要求する要件に応じた割当指標を用いて、前記ステート管理部を前記各データセンタに割り当てるときの割当コストを計算し、その割当コストに従って前記ステート管理部の割当先の前記データセンタを決定することを特徴とするリソース割当方法。

【請求項 8】

コンピュータを、請求項 1 ないし請求項 6 のいずれか 1 項に記載のリソース割当装置として機能させるためのリソース割当プログラム。

【発明の詳細な説明】

【技術分野】

【0001】

本発明は、リソース割当装置、リソース割当方法、および、リソース割当プログラムに関する。

【背景技術】

【0002】

分散クラウド環境は、NW (Network) 内に DC (Data Center) が分散して配置されたアーキテクチャである (非特許文献 1, 2)。DC は、サーバやユーザが従来担っていた処理をオフロード (代行) するための計算機資源 (リソース) を提供する。

以下、DC に割り当てる対象として、ステート管理部を例示する。なお、「ステート」とは、複数のユーザ間でリアルタイムにやりとりするサービスに用いられるデータである。「ステート管理部」とは、各ユーザから受信したアクセス内容をもとに、自身の管理するステートを更新し、その更新後のステートを各ユーザにデータ共有する処理部である。

【0003】

図 7 は、ステート管理部 5 z をオフロードする前のオンラインシステム 9 z 1 の構成図である。

オンラインシステム 9 z 1 では、サービス提供装置 1 z は、ステートを複数のユーザ端末 4 z 間でリアルタイムにデータ共有させるサービスを提供する。ステート管理部 5 z が管理するステートは、例えば以下が挙げられる。

・オンラインゲーム：アバタの位置・加速度・動作・装備、ダメージ判定や戦闘不能のステータスなど。

・XR(Extended Reality)：アバタ (ユーザ) の動作や五感情報、仮想世界において発生する事象など。

10

20

30

40

50

・オンライン会議：ユーザや画面共有の音声・映像データプレゼンタ権限やミュートの設定など。

このオンラインシステム 9 z 1 では、サービス提供装置 1 z と各ユーザ端末 4 z との間の通信距離が長いこと、大きな通信遅延がサービスの品質を低下させる要因となる。

【0004】

図 8 は、ステート管理部 5 z を DC 3 z にオフロードした後のオンラインシステム 9 z 2 の構成図である。

オンラインシステム 9 z 2 では、ステート管理部 5 z を動作させる装置を、サービス提供装置 1 z から、ユーザ端末 4 z に近い DC 3 z にオフロードさせている。このオフロードにより、下記のような効果が得られる。

- ・ユーザ：計算コスト・遅延の低減。
- ・通信事業者：トラフィック量・輻輳の軽減。
- ・サービス提供者：サーバ負荷・メンテナンスコストの削減。

【0005】

つまり、サービス品質を向上させるためには、複数の DC 3 z の候補から、どの DC 3 z にステート管理部 5 z を配置するかを選択することが重要となる。非特許文献 3 には、各ユーザ端末 4 z の間の遅延を求め、その遅延の最大値（最大の E2E 遅延）を小さくするようなリソース割当方法が記載されている。

【先行技術文献】

【非特許文献】

【0006】

【文献】Alicherry and T. V. Lakshman, "Network aware resource allocation in distributed clouds," in 2012 Proceedings IEEE INFOCOM, pp. 963-971, 2012.

【文献】M. Mukherjee, L. Shu, and D. Wang, "Survey of fog computing: Fundamental, network applications, and research challenges," in IEEE Communications Surveys & Tutorials, Vol. 20, No. 3, pp. 1826-1857, 2018.

【文献】A. Kawabata, B. C. Chatterjee, S. Ba and E. Oki, "A Real-Time Delay-Sensitive Communication Approach Based on Distributed Processing," in IEEE Access, vol. 5, pp. 20235-20248, 2017.

【発明の概要】

【発明が解決しようとする課題】

【0007】

同じオンラインシステムの環境を利用するユーザでも、ユーザのグループごとに様々なアプリケーションを実行させる。その際、例えば、以下の問題を定義する。

- ・複数の DC と多数のユーザとが、NW 内に散在している。
- ・ユーザ同士で比較的少人数（例：2 人～10 人）のグループを形成済みである。
- ・複数のグループに所属するユーザは存在しない。
- ・全グループのステート管理部を、それぞれ適切な DC に各々収容したい。

【0008】

また、上記の問題に対して、以下の前提条件がすでに与えられているものと仮定する。

- ・形成されたすべてのグループとそのメンバ。
- ・全ユーザと全 DC 間の遅延時間（例：ping 一回の測定値や複数回の平均値など）。
- ・各 DC に収容可能なユーザ数と、ユーザが満たすべき遅延要件（許容可能な遅延の最大値など）。

【0009】

図 9 は、ステート管理部を複数の DC にオフロードした場合の分散クラウド環境 8 z の構成図である。

分散クラウド環境 8 z では、波線で示す NW 内に 2 つの DC（DC1, DC2）が存在し、その DC には 5 人のユーザ（UA1, UA2, UA3, UB1, UB2）が収容されている。図 9 の例では、各ユーザは最寄りの DC に収容される。よって、DC1 には、ユーザ（UA1, UA2, UB1）

10

20

30

40

50

が収容される。DC2には、ユーザ（UA3,UB2）が収容される。

【0010】

ここで、前記の問題定義で説明したように、ユーザ間で第1グループ（UA1,UA2,UA3：2文字目が「A」のグループ）と、第2グループ（UB1,UB2：2文字目が「B」のグループ）とが形成されたとする。第1グループのメンバー同士で同じ対戦ゲームをするためには、UA1,UA2のステート（ST1）とUA3のステート（ST2）とをDC間で共有（同期）する必要がある。第2グループも同様である。

【0011】

また、第1グループのアプリケーションと、第2グループのアプリケーションとで、サービス要件が異なることもある。

例えば、第1グループで遊ばれるサバイバルゲームでは、100人など参加人数が多いことを重視し、対戦相手の情報は対戦格闘ゲームほど厳密には求められない。

一方、第2グループで遊ばれる対戦格闘ゲームでは、1対1などの少人数であるがメンバーが見る画面情報やメンバーの入力した操作情報がすばやく（フレーム単位で）相手側に反映される必要がある。

【0012】

このようなサービス要件を考慮しつつ、効率的にグループごとに適したリソース割当を行う手法が求められる。しかし、非特許文献3などの従来のリソース割当技術では、複数のグループが存在し、各グループが様々なサービス要件を求めるような複雑な状況には対応できなかった。

【0013】

そこで、本発明は、複数のユーザが形成したグループごとの要件に沿ったリソース割当を行うことを主な課題とする。

【課題を解決するための手段】

【0014】

前記課題を解決するために、本発明のリソース割当装置は、以下の特徴を有する。

本発明は、複数のユーザ端末から構成されるグループごとのステートをグループ内で共有させるためのステート管理部を、分散クラウド環境に配備された複数のデータセンタのうちのいずれかに割り当てる旨のリクエストを受信するリクエスト受信部と、

前記グループが要求する要件に応じた割当指標を用いて、前記ステート管理部を前記各データセンタに割り当てるときの割当コストを計算し、その割当コストに従って前記ステート管理部の割当先の前記データセンタを決定する割当計算部とを有することを特徴とする。

【発明の効果】

【0015】

本発明によれば、複数のユーザが形成したグループごとの要件に沿ったリソース割当を行うことができる。

【図面の簡単な説明】

【0016】

【図1】本実施形態に係わるオンラインシステムの構成図である。

【図2】本実施形態に係わるリソース割当装置のハードウェア構成図である。

【図3】本実施形態に係わる割当計算部がグループごとにステート管理部を割り当てる分散クラウド環境の構成図である。

【図4】本実施形態に係わる貪欲法のメイン処理を示すフローチャートである。

【図5】本実施形態に係わる図4の貪欲法のサブルーチン処理を示すフローチャートである。

【図6】本実施形態に係わる図4、図5の貪欲法の処理例を示す説明図である。

【図7】ステート管理部をオフロードする前のオンラインシステムの構成図である。

【図8】ステート管理部をDCにオフロードした後のオンラインシステムの構成図である。

【図9】ステート管理部を複数のDCにオフロードした場合の分散クラウド環境の構成図

10

20

30

40

50

である。

【発明を実施するための形態】

【0017】

以下、図面を参照して本実施形態を説明する。

【0018】

図1は、オンラインシステム9の構成図である。

オンラインシステム9は、サービス提供装置1と、リソース割当装置2と、分散クラウド環境8とがネットワークで接続されて構成される。分散クラウド環境8では、ステートを複数のユーザ端末4間でリアルタイムにデータ共有させるサービスが提供される。

そのため、ステート管理部5はグループごとのステートを管理する。各ステート管理部5はいずれかのDC3に配置される。

【0019】

まず、リソース割当装置2は、(手順1A)～(手順5A)のようなグループマッチング処理を行う。

(手順1A) ユーザ端末4はサービス提供装置1にアクセスする。

(手順2A) ユーザ端末4はサービス提供装置1上で、ステートを共有する相手である任意の(複数の)ユーザ端末4とマッチングし、グループを形成する。例えばオンラインゲームにおいては、「チームメイトや対戦相手」がステートを共有する相手になる。または、オンライン会議システムの場合は「同じ会議の参加者」がステートを共有する相手になる。

(手順3A) サービス提供装置1は、(手順2A)で形成されたグループのオフロード先として適切なDC3を選択し、そのDC3にグループごとのステート管理部5を配置することでサービスを開始する。

(手順4A) ステート管理部5には、サービスを提供するサーバ機能(オンラインゲームにおける対戦機能など)があらかじめVMやコンテナで実装されている。ステート管理部5は、グループ内で仮想空間などのステートを更新することで、サービスを提供する。

(手順5A) 対戦終了後にグループが解散された場合は、解散されたグループメンバーのユーザ端末4をサービス提供装置1に返却し、(手順2A)に戻る。

【0020】

そして、(手順4A)の詳細として、DC3は、各グループのステート管理部5が自身の管理するステートを更新する処理を行う。例えば、オンラインの対戦ゲームでは、以下の(手順1B)～(手順4B)に示すように、各ユーザ端末4はDC3にコマンドを送り、それと同時に、すでに同期処理され生成された仮想空間(ステート)をDC3から受信する。これにより、ユーザ端末4はDC3と1対1通信を行うだけで他のユーザ端末4のコマンドも取得できる。

【0021】

(手順1B) 各ユーザ端末4はDC3上のステート管理部5に自身のコマンドを書き込む。例えば、コントローラを用いて自身のアバタを前に進めるためのコマンドを送る。

(手順2B) DC3上のステート管理部5が各ユーザ端末4からの(手順1B)のコマンドを集約・同期する。例えば、各ユーザ端末4から届いたコマンドにしたがって、各ユーザのアバタに移動やアクションを実行させる。すなわち、先ほどよりも僅かに時間が経過した後の仮想世界をステートとして生成する。

(手順3B) DC3上のステート管理部5は(手順2B)の情報をを用いて仮想世界をレンダリング(画像化)するなどし、全ユーザ端末4に対してステートを送信する。例えば、ステート管理部5は全ユーザ端末4のコマンドを反映させたゲーム画面(の1コマ)を配信する。

(手順4B) ステートを受信した各ユーザ端末4は再びコマンドを送信する。例えば、敵が攻撃してくるのが見えたので回避コマンドを入力する。そして(手順1B)に戻る。

【0022】

以上は、オンラインの対戦ゲームの事例だが、IoT(Internet of Things)環境などに

10

20

30

40

50

において、複数の特定モジュールを組み合わせて最終的な処理結果を得る事例に分散クラウド環境 8 を適用してもよい。

例えば、気象予測モジュール用のユーザ端末 4 と、土壌分析モジュール用のユーザ端末 4 と、農作物分析モジュール用のユーザ端末 4 とを個別に用意する。そして、これら 3 つのモジュールをグループ化し、そのグループに属するモジュール群の出力結果を受け取り、それら进行处理したり田畑を管理したりする処理部（前記のステート管理部 5 に対応する処理部）を D C 3 に配置する。

【 0 0 2 3 】

（手順 3 A）において、リソース割当装置 2 は、サービス提供装置 1 からオフロードされるステート管理部 5 を、どの D C 3 に配置するか（以下「D C 配置」と呼ぶ）を、複数のユーザが形成したグループごとの要件（例えば、グループが利用しているアプリケーションごとの要件や、グループが利用しているサービスごとの要件）に沿って選択する。そのため、リソース割当装置 2 は、リクエスト受信部 2 1 と、NW データ収集部 2 2 と、割当計算部 2 3 と、制御部 2 4 とを有する。

【 0 0 2 4 】

リクエスト受信部 2 1 は、グループ構成と、ユーザの遅延要件とを含むグループごとのリクエストをサービス提供装置 1 から受信する。このリクエストは、複数のユーザ端末 4 から構成されるグループごとにデータ交換するためのステート管理部 5 を、分散クラウド環境 8 に配備された複数の D C 3 のうちのいずれかに割り当てる旨の内容である。

NW データ収集部 2 2 は、各ユーザ端末 4 と各 D C 3 との間の遅延情報と、各 D C 3 の空き容量（割当可能なユーザの数）情報とを含む NW データを分散クラウド環境 8 から測定および収集する。

【 0 0 2 5 】

割当計算部 2 3 は、リクエスト受信部 2 1 からのリクエストと、割当計算部 2 3 からの NW データとをもとに、リソース割当スキーム（以下「スキーム」）に従ってグループごとの D C 配置を計算する。スキームは、ステート管理部 5 をどの D C に、どの程度、どのようなポリシーで割り当てるかを決定するための情報（主にどの割当指標を重視するか）であり、分散クラウド環境 8 の性能はスキームに強く依存する。

【 0 0 2 6 】

なお、スキームは、リソース割当装置 2 がサービス提供装置 1 からリクエストとして受信してもよい。または、割当計算部 2 3 は、リソース割当装置 2 がサービス提供装置 1 からリクエストとして受信したアプリケーション種別をもとに、データベースを参照してスキームを求めてもよい。データベースには、アプリケーション種別ごとに対応するスキームが事前に登録されている。

または、割当計算部 2 3 は、グループ内のメンバからスキームの指定を受けてもよい。

制御部 2 4 は、割当計算部 2 3 からの D C 配置に従って、各ステート管理部 5 を各 D C 3 にリソース割り当てを行う。

【 0 0 2 7 】

図 2 は、リソース割当装置 2 のハードウェア構成図である。

リソース割当装置 2 は、C P U 9 0 1 と、R A M 9 0 2 と、R O M 9 0 3 と、H D D 9 0 4 と、通信 I / F 9 0 5 と、入出力 I / F 9 0 6 と、メディア I / F 9 0 7 とを有するコンピュータ 9 0 0 として構成される。

通信 I / F 9 0 5 は、外部の通信装置 9 1 5 と接続される。入出力 I / F 9 0 6 は、入出力装置 9 1 6 と接続される。メディア I / F 9 0 7 は、記録媒体 9 1 7 からデータを読み書きする。さらに、C P U 9 0 1 は、R A M 9 0 2 に読み込んだプログラム（リクエスト受信部 2 1、NW データ収集部 2 2、割当計算部 2 3、制御部 2 4 に備わるリソース割当プログラムなど）を実行することにより、各処理部を制御する。そして、このプログラムは、通信回線を介して配布したり、C D - R O M 等の記録媒体 9 1 7 に記録して配布したりすることも可能である。

【 0 0 2 8 】

図3は、割当計算部23がグループごとにステート管理部を割り当てる分散クラウド環境8の構成図である。

なお、図9の分散クラウド環境8zではグループの概念を考慮せず、ユーザ単位でDCの選択を行っていたために、DC間のステート共有が必要であった。そのため、ステート共有による余分なオーバーヘッドの影響で、ユーザ間の通信遅延も増大してしまっていた。

一方、図3の分散クラウド環境8では、割当計算部23はグループ単位でのDC配置を行う。これにより、1グループあたり1つのステート管理部5が1台のDC3に割り当てられる。例えば、第1グループ(UA1,UA2,UA3)用のステート管理部(STA)がDC1に割り当てられる一方、第2グループ(UB1,UB2)用のステート管理部(STB)がDC2に割り当てられる。よって、ステート共有時のオーバーヘッドを抑制し、厳しいリアルタイム要件に対応できる。

【0029】

また、割当計算部23は、グループごとのアプリケーション種別に適したスキームをリクエストから参照することで、グループごとの要件に適したDC配置を立案できる。

つまり、割当計算部23は、グループごとの要件に応じたスキームの割当指標を用いて、ステート管理部5を各データセンタに割り当てるときの割当コストを計算し、その割当コストに従ってステート管理部5の割当先のデータセンタを決定する。以下、アプリケーション種別とスキームとの組み合わせとして事例を例示する。

【0030】

「事例1：完全同期型」は、グループに属するすべてのユーザの通信を逐一待ってからステートを同期することで、グループに属するすべてのユーザが同じ画面を閲覧させるアプリケーションである。以下、完全同期型のアプリケーションを例示する。

- ・1対1の対戦格闘ゲーム。
- ・図1の説明で前記した田畑を管理するシステム。

完全同期型のスキーム(割当指標)は、「グループ内の最大遅延」を最小化するものである。グループ全体のQoS(Quality of Service)やQoE(Quality of Experience)が最も遅延の大きなユーザに依存するためである。

【0031】

「事例2：準同期型」は、ユーザ間で厳密な同期を取らない(取れない)アプリケーションであり、例えば、100人程度のFPS(First-Person Shooter)のサバイバルゲームである。グループに属する遅延の大きいユーザの通信を待たずにステートを同期することで、遅延の大きいユーザが見る画面がカクついたり、遅延の大きいユーザのキャラクターの最新位置が他ユーザに見えなくなったりする。

準同期型のスキーム(割当指標)は、「グループ内の平均遅延」を最小化するものである。例えば100人のグループのうちの3人程度のメンバだけ大きな遅延が発生したとしても、その不便を感じるのは3人だけで、残りの97人は小さい平均遅延で快適な環境が提供されれば充分である。

【0032】

「事例3：公平環境型」は、ユーザ間のプレイ環境において公平性が特に求められるリアルタイムアプリケーションであり、以下に例示する。

- ・10人程度が同時に同じサーキットを走行するレースゲーム。
- ・XR空間やバーチャル会議室では、ユーザの動作が視界に反映されるまでの時間(motion-to-photonlatency)が20[ms]以下になることが望ましい。
- ・現実の街を模した仮想空間内で会話をするアプリケーションでも、レスポンスを低下させないために一つの空間に滞在可能なユーザ数が制限されている。

公平環境型のスキーム(割当指標)は、「グループ内の遅延のばらつき(分散または標準偏差)」を最小化するものである。

一般に遅延が大きいほどQoS・QoEが低下し、仮想世界における優位性(例：ゲームのスコア)にも影響する。よって、公平環境型のアプリケーションに限らず、他の事例のアプリケーションでも、グループ内の遅延のばらつきを小さくすることが望ましい。

10

20

30

40

50

【0033】

以上、3種類のスキームを例示したが、例えば、準同期型のスキームとして「グループ内の平均遅延」および「グループ内の最大遅延」の最小化を同時に行うなど、複数の割当指標を組み合わせてもよい。同様に、公平環境型のスキームとして、「グループ内の平均遅延」と、「グループ内の最大遅延」と、「グループ内の遅延のばらつき」という3つの割当指標をバランスよく最小化してもよい。

【0034】

以下では、グループ内の遅延の最小化と平等化を両立するための定式化を行う。具体的には、3つの割当指標を総合的に評価可能なモデルを説明する。まず、モデルに用いられるパラメータを定義する。

記号「 $i \in I$ 」は、ユーザ i とその集合 I を示す。

記号「 $j \in J$ 」は、グループ j ($j=1,2,\dots,n$)とその集合 J を示す。

記号「 $k \in K$ 」は、 DCk (k 番目の DC 、 $k=1,2,\dots,m$)とその集合 K を示す。

記号「 $I_j \subset I$ 」は、グループ j に属するユーザの集合 I_j を示す。

記号「 w_{ij} 」は、二値変数であり、ユーザ i がグループ j に属しているなら「1」、そうでなければ「0」の値をとる。

記号「 d_{ik} 」は、ユーザ i - DCk 間の遅延時間を示す。

記号「 D_j 」は、ユーザ i の遅延要件を示す。

記号「 C_k 」は、 DCk が収容可能なユーザ数を示す。

これらの各パラメータは、リクエスト受信部21のリクエストや、NWデータ収集部22の収集データとして、割当計算部23に与えられる。

【0035】

【数1】

$$a_{jk} = \frac{\sum_{i \in I_j} d_{ik}}{\|I_j\|} = \frac{\sum_{i \in I} w_{ij} d_{ik}}{\sum_{i \in I} w_{ij}} \quad \dots \text{ (式1)}$$

$$v_{jk}^2 = \frac{\sum_{i \in I_j} (a_{jk} - d_{ik})^2}{\|I_j\|} = \frac{\sum_{i \in I} w_{ij} (a_{jk} - d_{ik})^2}{\sum_{i \in I} w_{ij}} \quad \dots \text{ (式2)}$$

【0036】

割当計算部23は、割当指標として(式1)および(式2)を計算する。最大遅延の計算式は(式4)で後記する。

(式1)の左辺「 a_{jk} 」は、グループ j を DCk に収容した場合のユーザ $i \in I_j$ の平均遅延を示す。

(式2)の左辺「 v_{jk}^2 」は、グループ j を DCk に収容した場合のユーザ $i \in I_j$ の遅延分散を示す。

【0037】

【数2】

$$\text{minimize} \sum_{j \in J} \sum_{k \in K} x_{jk} C_{jk} \quad \dots \text{ (式3)}$$

【0038】

(式3)は DC 割当の目的関数を示す。この目的関数は、グループ j の割当コストの総和を最小化(minimize)する問題に帰着する。

記号「 c_{jk} 」は、グループ j を DCk に収容した場合の割当コストである。

記号「 x_{jk} 」は、割当計算部 23 の計算結果である。「 x_{jk} 」は決定変数であり、グループ j を DCk に収容したら「1」の値、そうでなければ「0」の値をとる。

割当計算部 23 が割当コスト c_{jk} を求める具体的な計算式として（式 4）または（式 5）を説明する。なお、遅延のばらつきを示す数値として、（式 4）では分散を採用し、（式 5）では標準偏差を採用した。分散の代わりに標準偏差を用いることで、2 乗を含む項がなくなるので、各項のスケールをある程度合わせることができる。

【0039】

【数 3】

$$\left. \begin{aligned} c_{jk} &= (1 - \alpha - \beta) a_{jk} + \alpha v_{jk}^2 + \beta \max_{i \in I} w_{ij} d_{ik} \\ \alpha, \beta &\geq 0, \quad \alpha + \beta \leq 1 \end{aligned} \right\} \quad (\text{式4}) \quad 10$$

$$\left. \begin{aligned} c_{jk} &= (1 - \alpha - \beta) a_{jk} + \alpha s_{jk} + \beta \max_{i \in I} w_{ij} d_{ik} \\ \alpha, \beta &\geq 0, \quad \alpha + \beta \leq 1, \quad s_{jk} = \sqrt{v_{jk}^2} \end{aligned} \right\} \quad (\text{式5})$$

【0040】

α, β ($\alpha, \beta \geq 0, \alpha + \beta \leq 1$) は 3 つの割当指標間のトレードオフを調整するためのパラメータであり、サービス要件等に応じて、サービス提供者、ユーザ、または、NW 事業者などが任意に決定する。

（式 4）の右辺第 1 項の「 $1 - \alpha - \beta$ 」は、その数値が大きいほど「グループ内の平均遅延」を重視するハイパーパラメータである。

（式 4）の右辺第 2 項の「 α 」は、その数値が大きいほど「グループ内の遅延のばらつき」を重視するハイパーパラメータである。

（式 4）の右辺第 3 項の「 β 」は、その数値が大きいほど「グループ内の最大遅延」を重視するハイパーパラメータである。なお、右辺第 3 項の β よりも後の計算式はグループ内の最大遅延の計算式である。

【0041】

つまり、（式 4）または（式 5）では、以下のように 3 つのハイパーパラメータの配分に応じて、3 種類の指標の優先度合いを調整できる。

- ・ $\alpha = 0$ かつ $\beta = 0$ とした場合は、平均遅延のみを最適化する（事例 2：準同期型に適した設定）。

- ・ $\alpha \neq 0$ かつ $\beta = 0$ ($\alpha + \beta < 1$) とした場合は（例： $\alpha = 0.5, \beta = 0$ ）、平均遅延と分散（標準偏差）とを最適化する。

- ・ $\alpha = 1$ かつ $\beta = 0$ ($\alpha + \beta = 1$) とした場合は、分散（標準偏差）のみを最適化する（事例 3：公平環境型に適した設定）。

- ・ $\alpha = 0$ かつ $\beta \neq 0$ ($\alpha + \beta < 1$) とした場合は、平均遅延と最大遅延とを最適化する。

- ・ $\alpha = 0$ かつ $\beta = 1$ ($\alpha + \beta = 1$) とした場合は、最大遅延のみを最適化する（事例 1：完全同期型に適した設定）。

- ・ $\alpha \neq 0$ かつ $\beta \neq 0$ ($\alpha + \beta = 1$) とした場合は、分散（標準偏差）と最大遅延とを最適化する。

- ・ $\alpha \neq 0$ かつ $\beta \neq 0$ ($\alpha + \beta < 1$) とした場合は、平均遅延・分散（標準偏差）・最大遅延のすべてを最適化する。

【0042】

【数 4】

$$\left. \begin{aligned} c_{jk} &= \alpha a_{jk} + \beta v_{jk}^2 + \gamma \max_{i \in \mathcal{I}} w_{ij} d_{ik}. \\ \alpha, \beta, \gamma &\geq 0 \end{aligned} \right\} \quad (\text{式6})$$

$$\left. \begin{aligned} c_{jk} &= \alpha a_{jk} + \beta s_{jk} + \gamma \max_{i \in \mathcal{I}} w_{ij} d_{ik}. \\ \alpha, \beta, \gamma &\geq 0 \end{aligned} \right\} \quad (\text{式7})$$

10

【0043】

なお、割当計算部23は、割当コスト c_{jk} を求める計算式として、分散を用いる（式4）の代わりに（式6）を用いてもよいし、標準偏差を用いる（式5）の代わりに（式7）を用いてもよい。

（式6）および（式7）では、3種類のハイパーパラメータ（ α, β, γ ）を用いている。これにより、パラメータ設定の自由度が高いが、その分扱い（最適なパラメータ設定）が難しくなる。

【0044】

20

右辺第1項の「 α 」は、その数値が大きいほど「グループ内の平均遅延」を重視するハイパーパラメータである。

右辺第2項の「 β 」は、その数値が大きいほど「グループ内の遅延のばらつき」を重視するハイパーパラメータである。

右辺第3項の「 γ 」は、その数値が大きいほど「グループ内の最大遅延」を重視するハイパーパラメータである。

一方、（式4）や（式5）では、パラメータが一つ少なく設定範囲も狭い。しかし、割合で重みが設定できることもあり、3種類のハイパーパラメータ（ α, β, γ ）を用いるよりも、扱いが容易になる可能性が高い。

【0045】

30

【数5】

$$\left. \begin{aligned} \text{subject to } \sum_{k \in \mathcal{K}} x_{jk} &\leq 1, \quad \forall j \in \mathcal{J}, \\ \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} x_{jk} w_{ij} &\leq C_k, \quad \forall k \in \mathcal{K}, \\ x_{jk} w_{ij} d_{ik} &\leq D_i, \quad \forall i \in \mathcal{I}, \forall j \in \mathcal{J}, \forall k \in \mathcal{K}, \\ x_{jk} &\in \{0, 1\}, \quad \forall j \in \mathcal{J}, \forall k \in \mathcal{K}. \end{aligned} \right\} \quad (\text{式8})$$

40

【0046】

（式8）は、（式3）の目的関数に対応する制約条件（subject to）を示す。制約条件は以下の通りである。

- ・1グループに割当てられるDCは1個以下である。
- ・収容されるユーザ数が各DCの容量を超えない。
- ・各ユーザの実際の遅延が遅延要件を満たす。

50

【0047】

以上、(式3)～(式8)として定式化した問題は、組み合わせ最適化問題であり、大域的最適解を求めるためには、割当計算部23は、すべての組み合わせを調査(総当たり計算)する必要がある。このときの本問題の計算量は、グループ数を n 、DC数を m として計算量オーダー(m の n 乗)であり膨大である。

【0048】

そこで、割当計算部23が総当たり計算の代わりに用いる近似アルゴリズムとして、貪欲法(greedy algorithm)と、局所探索法とを順に説明する。割当計算部23は、貪欲法を単独で用いる、または、貪欲法と局所探索法とを併用する。これにより、総当たり計算で求まる最適解(または最適解に近いスコアの準最適解)を、総当たり計算よりも少ない計算量で求めることができる。

10

【0049】

図4は、貪欲法のメイン処理を示すフローチャートである。なお、貪欲法とは、1つの大きな問題を複数の小さな問題に分割し、小さな問題を個別に評価して評価値の高い候補を採用する手法である。

割当計算部23は、リクエスト受信部21からのリクエストと、NWデータ収集部22からのNWデータとをもとに、図6で後記するコストテーブルを作成する(S101)。割当計算部23は、S101のコストテーブルの各グループについて、コストの最小値を算出する(S102)。

割当計算部23は、コストテーブルの各グループについてコストの最小値について昇順ソートする(S103)。

20

【0050】

割当計算部23は、グループの変数 j に初期値1を代入する(S104)。

ここで、割当計算部23は、グループ j をコストテーブルのグループ数まで1つずつ順番に選択するループを実行し(S105～S107)、このループ内でグループ j にDC割当を行うサブルーチン(図5)を呼び出す(S106)。

なお、S106の処理において、あるユーザの遅延要件を満たせるような割当先DCが存在しないとき、もしくはDCのキャパシティが十分でない場合は、割当できないグループが発生する可能性がある。

【0051】

30

図5は、図4の貪欲法のサブルーチン処理(S106)を示すフローチャートである。

割当計算部23は、コストテーブルの第 j グループ行を読み込み(S111)、その中でコストが最小となるDCに割当を試みる(S112)。

割当計算部23は、ユーザの遅延要件を満たし(S113, Yes)、かつ、DCの容量が充分にある(S114, Yes)場合に、グループ j をそのDCに割り当てる(S115)。

一方、(S113, No)、または、(S114, No)の場合は、割当計算部23は、そのDCを第 j グループのコストテーブルから削除する(S116)。

【0052】

図6は、図4、図5の貪欲法の処理例を示す説明図である。

説明をわかりやすくするため、図6では以下の問題設定とする。

40

- ・6個のグループ(G1-G6)を3個のDC(DC1-DC3)に割り当てたい。

- ・各DCは2グループまで割当可能である。なお、本来の C_k の単位はユーザ数であるが、ここでは簡単のためグループ数で指定した。

- ・簡単のため、ユーザの遅延要件は考えない(つまり $D=\infty$)。

- ・平均遅延のみを考慮するため、 $\alpha=\beta=0$ とする。もちろん他のパラメータ設定を用いてもよい。

【0053】

コストテーブル201は、グループ j をDC k に收容した場合の割当コスト c_{jk} について、グループ j を行としDC k を列としたときの2次元のテーブルである。割当計算部23は図4のS101でコストテーブル201を作成する。そして、割当計算部23は、コストテ

50

ーブル 201 を最小値で昇順ソートした結果をコストテーブル 202 とする (S103)。

DC 割当テーブル 211 ~ 217 は、コストテーブル 202 をもとにした割当計算部 23 の計算結果「 x_{jk} 」をテーブル形式にしたものである。例えば、DC 割当テーブル 214 では、DC1 に「G4, G2」という 2 グループが割り当てられ、DC2 にはグループが未割当であり (記号「-」)、DC3 に「G5」という 1 グループが割り当てられている。

【0054】

以下の手順により、割当計算部 23 は、コストテーブル 202 の上位に位置するグループから順に、S106 のサブルーチン呼び出すことで各グループを各 DC に割り当てる。

(第 1 行の G4) 割当計算部 23 は、G4 をコスト最小 (= 6.0) となる DC1 に割り当てる (DC 割当テーブル 211 → DC 割当テーブル 212)。

(第 2 行の G2) 割当計算部 23 は、G2 をコスト最小となる DC1 に割り当てる (DC 割当テーブル 212 → DC 割当テーブル 213)。

(第 3 行の G5) 割当計算部 23 は、G5 をコスト最小となる DC3 に割り当てる (DC 割当テーブル 213 → DC 割当テーブル 214)。

(第 4 行の G6) 割当計算部 23 は、G6 をコスト最小となる DC1 に割り当てようとするも、DC1 は容量不足である。よって、割当計算部 23 は、G6 をコストが 2 番目に小さい DC3 に割り当てる (DC 割当テーブル 214 → DC 割当テーブル 215)。

(第 5 行の G3) 割当計算部 23 は、G3 をコスト最小となる DC2 に割り当てる (DC 割当テーブル 215 → DC 割当テーブル 216)。

(第 6 行の G1) 割当計算部 23 は、G1 をコスト最小となる DC2 に割り当てる (DC 割当テーブル 216 → DC 割当テーブル 217)。

【0055】

以上、図 4 ~ 図 6 を参照して貪欲法について説明した。

以下では、貪欲法のアルゴリズムを詳細に説明する疑似コードを例示する。この疑似コードは、代入「 $A \leftarrow 1$ 」(変数 A に値 1 を代入)、繰り返し制御「for ~ end for、while ~ end while」、分岐制御「if ~ end if」などを行う手続型言語である。また、疑似コードの行頭には、説明用の行番号 (L01, L02, ...) を付加した。

各関数 (function) は、「Input: ~」で与えられた入力変数をもとに、所定の計算を行い、その計算結果を「Output: ~」で示す出力変数として応答 (return) する。

【0056】

まず、Greedy_Allocation 関数を示す。

Input: $\alpha, \beta, w_{ij}, d_{ik}, D_i$, and C_k for $\forall i \in I, \forall j \in J$, and $\forall k \in K$

Output: x_{jk} for $\forall j \in J$ and $\forall k \in K$

L01: function Greedy_Allocation($\alpha, \beta, w_{ij}, d_{ik}, D_i, C_k$)

L02: for all $i \in I, j \in J, k \in K$ do

L03: $w[i][j] \leftarrow w_{ij}, d[i][k] \leftarrow d_{ik}$

L04: $D[i] \leftarrow D_i, C[k] \leftarrow C_k$

L05: Calculate c_{jk} using (式 4) ~ (式 7)

L06: $c[j][k] \leftarrow c_{jk}$

L07: end for

L08: for all $j \in J$ do

L09: $J[j] \leftarrow j, L[j] \leftarrow \min_k c[j][k]$

L10: end for

L11: Sort J in ascending order of L

L12: $X \leftarrow \text{GroupDC_Mapping}(J, c, w, d, D, C)$

L13: return X

L14: end function

【0057】

Greedy_Allocation 関数 ($\alpha, \beta, w_{ij}, d_{ik}, D_i, C_k$) では、 α および β を用いてコスト表を作成し (L05)、表の中で最小コストが小さいグループから優先的に (L11) DC 割当を行う

ための前処理を行う。なお、実際のDC割当については引数 w_{ij}, d_{ik}, D_i, C_k と、ここで作成されたコスト表 c およびコスト表を参照する順列 J を引数としたGroupDC_Mapping関数を呼び出して実行する（L12）。

図4のフローチャートでは、S100がL01に、S101がL05に、S102がL09に、S103がL11に、S106がL12に、それぞれ対応する。

【0058】

次に、GroupDC_Mapping関数を示す。

L16: function GroupDC_Mapping(J, c, w, d, D, C)

L17: for $j'=1$ to size(J) do

L18: $j \leftarrow J[j']$

L19: Discover I_j from I using w

L20: $X[j] \leftarrow DC_Selection(c[j], I_j, d, D, C)$

L21: if $X[j] = \infty$ then

L22: $D[i] \leftarrow \infty$ for $\forall i \in I_j$

L23: $X[j] \leftarrow DC_Selection(c[j], I_j, d, D, C)$

L24: end if

L25: end for

L26: return X

L27: end function

【0059】

GroupDC_Mapping関数(J, c, w, d, D, C)では、与えられたコスト表 c を順列 J にしたがって（L17）順に参照し（すなわちコスト表のある行 $c[j]$ を抽出し）、各グループにDCを割り当てるためのDC_Selection関数を順次呼び出して割当を得る（L20, L23）。仮にDC_Selection関数がユーザの遅延要件 D を満たすことのできるDCが存在しないことを示した場合には、遅延要件 D を無視して割当を行う。

なお、L20, L23の $X[j] \leftarrow k$ は、割り当て有なら値1（ $x_{jk} = 1$ if $k = k'$ ）、または、割り当て無なら値0（ $x_{jk} = 0$ otherwise）と等価である。

図4のフローチャートでは、S104、S105、および、S107がL17～L25のfor文に、S112がL20およびL23に、それぞれ対応する。

【0060】

さらに、DC_Selection関数を示す。

L29: function DC_Selection($c[j], I_j, d, D, C$)

L30: $tmp \leftarrow c[j]$

L31: while True do

L32: $k \leftarrow \arg \min_k tmp[k]$

L33: if $tmp[k] = \infty$ then

L34: return ∞

L35: end if

L36: if $d[i][k] \leq D[i]$ for $\forall i \in I$ and $|I_j| \leq C[k]$ then

L37: $C[k] \leftarrow C[k] - |I_j|$

L38: return k

L39: else

L40: $tmp[k] \leftarrow \infty$

L41: end if

L42: end while

L43: end function

【0061】

DC_Selection関数($c[j], I_j, d, D, C$)では、グループメンバ I_j について、各ユーザの遅延要件 D を満たし、残容量 C がグループサイズ $|I_j|$ より大きいDCの中から（L36）、遅延コスト $c[j][k]$ が最も小さい DC_k を割当先として選択する（L37, L38）。

図5のフローチャートでは、S113およびS114がL36に、S115がL37に、S116がL40に、それぞれ対応する。なお、L40の「 $\text{tmp}[k] \leftarrow \infty$ 」とは、グループコストを無限大にすることで、そのグループを割り当て対象から削除することと等価である。

【0062】

以上説明したように、割当計算部23は、貪欲法を単独で用いる場合は、Greedy_Allocation関数を呼び出せばよい。これにより、グループをソートし、コストが小さいものから優先的に割り当てていくことで、少ない計算時間で高精度な最適化が可能となる。一方、割当計算部23は、提案法と局所探索法とを組み合わせることで、さらに良い解を求めてもよい。

【0063】

以下は、提案法と局所探索法とを組み合わせたアルゴリズムの疑似コードである。

Input: $N, \alpha, \beta, w_{ij}, d_{ik}, D_i$, and C_k

Output: x_{jk}

M01: $X \leftarrow \text{Greedy_Allocation}(\alpha, \beta, w_{ij}, d_{ik}, D_i, C_k)$

M02: Calculate the total cost t_x of X like (式3)

M03: $n \leftarrow 0$

M04: while $n++ < N$ do

M05: Randomize the order of J

M06: $X' \leftarrow \text{GroupDC_Mapping}(J, c, w, d, D, C)$

M07: Calculate the total cost t'_x of X'

M08: if $t'_x < t_x$ then

M09: $X \leftarrow X', t_x \leftarrow t'_x$

M10: end if

M11: end while

【0064】

この疑似コード（M01～M11）では、まず、割当計算部23は、Greedy_Allocation関数を実行する（M01）。次に、割当計算部23は、GroupDC_Mapping関数を一定回数（M04のN回）実行する（M06）。このとき、「コストテーブルのソート」の部分で、割当計算部23は、グループの順列をランダムに変更する（M05）。そして、割当計算部23は、N回の繰り返しの中で最も全体のコストが低かった割当を選択する（M08～M10）。

この疑似コード（M01～M11）により、GroupDC_Mapping関数をN回繰り返す旨の探索回数の追加により、計算コストは増えるものの、貪欲法よりも割当コストがさらに小さくなる可能性がある。なお、探索回数Nはサービス提供者や分散環境運用者が任意に設定する。

【0065】

ここで、リクエスト受信部21に与えられるグループ数に着目して、以下のように分類する。

（分類1）オフライン割当は、割り当てるべきすべてのグループがあらかじめリクエスト受信部21に与えられる場合である。

（分類2）バッチ割当は、割り当てるべきグループの一部（複数のグループ）が逐次リクエスト受信部21に与えられ、その都度（比較的小規模な）オフライン割当を実行する場合である。例えば大規模システムでは、例えば10秒の間にいくつものグループが作成されるため、この場合は10秒ごとにバッチ割当をおこなうニーズが出てくる。

（分類3）オンライン割当は、割り当てるべきグループが一つずつリクエスト受信部21に与えられ、その都度割当を実行する場合である。例えば、バッチ処理を実行するほど頻繁にグループが形成されない場合や、グループ形成から割当完了までの時間を可能な限り小さくしたい場合などが挙げられる。

【0066】

これまで説明したGreedy_Allocation関数は、リクエスト受信部21にまとまった数の

10

20

30

40

50

グループが与えられた場合（オフライン割当やバッチ処理）を想定している。一方、リクエスト受信部 2 1 に割り当てべきグループが逐次やってくる場合には、以下に示すオンライン割当の疑似コードを実行すればよい。

Input: $\alpha, \beta, w_{ij}, d_{ik}, D_i$, and C_k for $\forall i \in I, \forall j \in J$, and $\forall k \in K$

Output: x_{jk} for $\forall j \in J$ and $\forall k \in K$

N01: while True do

N02: if receive an offload request for group j then

N03: Calculate c_{jk} for $\forall k \in K$

N04: $c[k] \leftarrow c_{jk}$

N05: $X[j] \leftarrow \text{DC_Selection}(c[j], l_j, d, D, C)$

N06: if $X[j] = \infty$ then

N07: $D[i] \leftarrow \infty$ for $\forall i \in I_j$

N08: $X[j] \leftarrow \text{DC_Selection}(c[j], l_j, d, D, C)$

N09: end if

N10: return $X[j]$

N11: end if

N12: end while

【0067】

この疑似コード（N01～N12）では、まず、割当計算部 2 3 は、グループ j の割当リクエストを受信したとき（N02）、そのグループ j と各 DC 間の割当コスト c_{jk} をすべて計算し（N03）、グループ j についてのみのコストテーブルを作成する（N04）。

そして、割当計算部 2 3 は、グループ j について DC_Selection 関数を実行する（N05, N08）。このとき、割当計算部 2 3 は、各 DC の容量 $C_k = C[k]$ を常にモニタしておき、グループ j が DC を離れたときはその分容量を増やす。

【0068】

以下、（分類 1）～（分類 3）それぞれの計算量を示すため、各分類の関数の呼び出す回数を例示する。

（分類 1）オフライン割当として、グループ数 = 10 がいっぺんに与えられる場合は、以下の通りである。

（分類 1）で貪欲法だけを実行する場合は、Greedy_Allocation 関数を 1 回、GroupDC_Mapping 関数を 1 回、DC_Selection 関数を 10 回以上実行する。なお、DC_Selection 関数を呼び出す回数が 10 回を超える場合は、ユーザの遅延要件を満たせる DC がなかった場合（L20 行目の返り値が ∞ となった場合）に、L23 行目に移って再び DC_Selection 関数が呼び出される場合である。

（分類 1）で貪欲法と局所探索とを併せて実行する場合は、Greedy_Allocation 関数を 1 回実行した後、GroupDC_Mapping 関数および DC_Selection 関数をワンセットとして N 回繰り返す（Greedy_Allocation 関数を 1 回、GroupDC_Mapping 関数を $1+N$ 回、DC_Selection 関数を $10 \times (1+N)$ 回以上）。 N はオペレータ等が任意に設定可能なパラメータである。

【0069】

（分類 2）バッチ割当として、グループ数 = 10 が、「グループ数 = 5」→「グループ数 = 3」→「グループ数 = 2」の順に 3 段階で与えられる場合は、以下の通りである。

（分類 2）で貪欲法だけを実行する場合は、以下のようになる。

- ・ Greedy_Allocation 関数を 1 回（グループ数 5 のとき） + 1 回（グループ数 3 のとき） + 1 回（グループ数 2 のとき）で計 3 回実行する。

- ・ GroupDC_Mapping 関数を 1 回（グループ数 5 のとき） + 1 回（グループ数 3 のとき） + 1 回（グループ数 2 のとき）で計 3 回実行する。

- ・ DC_Selection 関数を 5 回以上（グループ数 5 のとき） + 3 回以上（グループ数 3 のとき） + 2 回以上（グループ数 2 のとき）で計 10 回以上実行する。

【0070】

10

20

30

40

50

(分類2)で貪欲法と局所探索とを併せて実行する場合は、Greedy_Allocation関数を各1回ずつ行った後、GroupDC_Mapping関数およびDC_Selection関数をワンセットとして各バッチ処理時にN回ずつ繰り返すことになるため、以下のようになる。

- ・ Greedy_Allocation関数を各1回ずつ計3回実行する。
- ・ GroupDC_Mapping関数を各(1+N)ずつ×3回実行する。
- ・ DC_Selection関数を $5 \times (1+N)$ 回以上 + $3 \times (1+N)$ 回以上 + $2 \times (1+N)$ 回以上で計 $10 \times (1+N)$ 回以上実行する。

【0071】

(分類3)オンライン割当として、グループ数=10が1つずつ与えられる場合は、以下のようになる。

(分類3)で貪欲法だけを実行する場合は、DC_Selection関数を1回or2回ずつ10回呼び出されるため計10回以上実行する。

(分類3)で貪欲法と局所探索とを併せて実行することは不可である。1つのグループしかない場合は、割当順序が関係ないため、局所探索をおこなう余地がないためである。

【0072】

[効果]

本発明のリソース割当装置2は、複数のユーザ端末4から構成されるグループごとのステートをグループ内で共有させるためのステート管理部5を、分散クラウド環境8に配備された複数のDC3のうちのいずれかに割り当てる旨のリクエストを受信するリクエスト受信部21と、

グループが要求する要件に応じた割当指標を用いて、ステート管理部5を各DC3に割り当てるときの割当コストを計算し、その割当コストに従ってステート管理部5の割当先のDC3を決定する割当計算部23とを有することを特徴とする。

【0073】

これにより、グループのメンバであるユーザ端末4が、そのグループでステート管理部5を共有して行うアプリケーションに適した割当先にステート管理部5を割り当てることができる。よって、複数のユーザが形成したグループごとの要件に沿ったリソース割当を行うことができる。

【0074】

本発明は、割当計算部23が、グループを構成する各ユーザ端末4とDC3との間の遅延時間を求め、それらの求めた遅延時間の平均遅延をステート管理部5の割当指標とすることを特徴とする。

【0075】

これにより、サバイバルゲームなどの多人数が参加するアプリケーションにおいて、その多数が満足する割当先を発見できる。

【0076】

本発明は、割当計算部23が、グループを構成する各ユーザ端末4とDC3との間の遅延時間を求め、それらの求めた遅延時間の最大遅延をステート管理部5の割当指標とすることを特徴とする。

【0077】

これにより、対戦格闘ゲームなどのグループに属するすべてのユーザの通信で厳密な同期を要するアプリケーションに適した割当先を発見できる。

【0078】

本発明は、割当計算部23が、グループを構成する各ユーザ端末4とDC3との間の遅延時間を求め、それらの求めた遅延時間の分散または標準偏差をステート管理部5の割当指標とすることを特徴とする。

【0079】

これにより、レースゲームなどのユーザ間のプレイ環境の公平性が特に求められるリアルタイムアプリケーションに適した割当先を発見できる。

【0080】

10

20

30

40

50

本発明は、割当計算部 2 3 が、リクエスト受信部 2 1 が受信したリクエストの各グループの割当コストの総和を最小化する組み合わせ最適化問題に対して、貪欲法を用いた近似アルゴリズムにより、各グループの割当先を計算することを特徴とする。

【0081】

これにより、組み合わせ最適化問題に対して総当たり計算よりも少ない計算量で、適切な割当先を計算することができる。

【0082】

本発明は、割当計算部 2 3 が、リクエスト受信部 2 1 が受信したリクエストの各グループの割当コストの総和を最小化する組み合わせ最適化問題に対して、貪欲法と局所探索法とを併用する近似アルゴリズムにより、各グループの割当先を計算することを特徴とする。

10

【0083】

これにより、貪欲法を単独で用いる場合と比べて、さらに良い割当先を発見する可能性が高まる。

【符号の説明】

【0084】

- 1 サービス提供装置
- 2 リソース割当装置
- 3 DC（データセンタ）
- 4 ユーザ端末
- 5 ステート管理部
- 8 分散クラウド環境
- 9 オンラインシステム
- 2 1 リクエスト受信部
- 2 2 NWデータ収集部
- 2 3 割当計算部
- 2 4 制御部

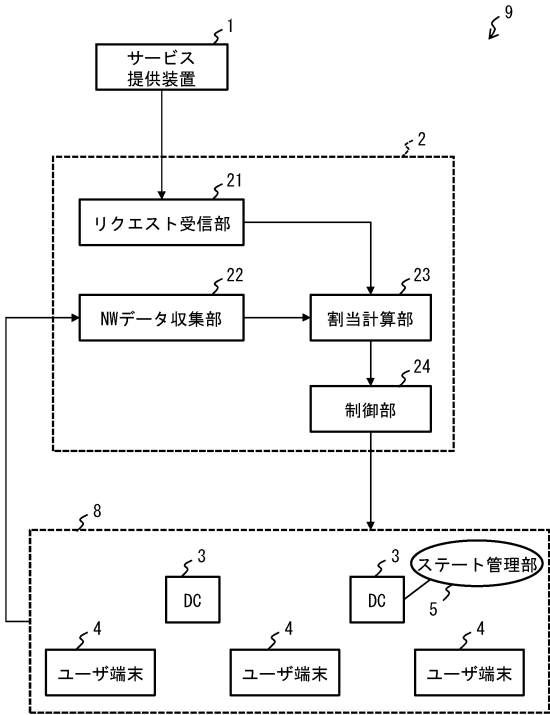
20

30

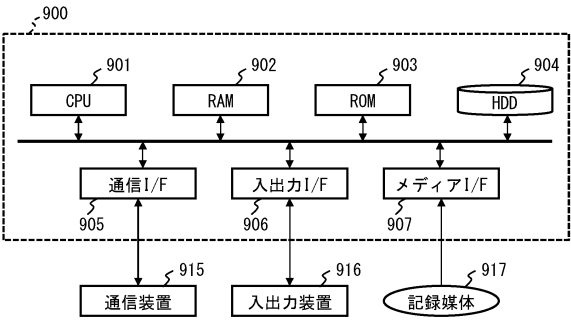
40

50

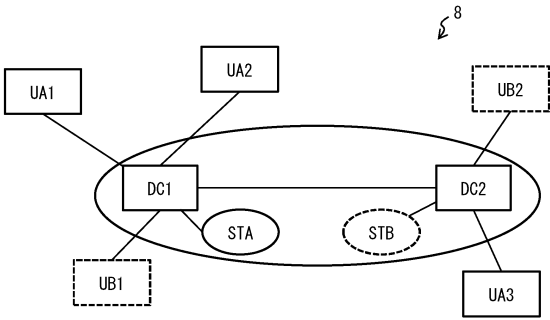
【図面】
【図 1】



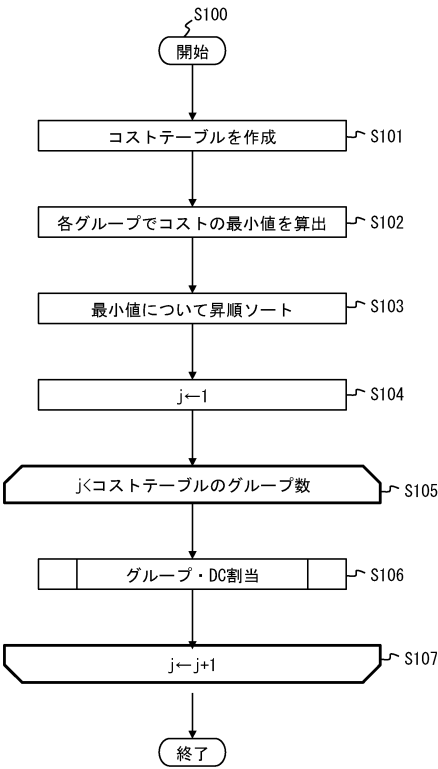
【図 2】



【図 3】



【図 4】



10

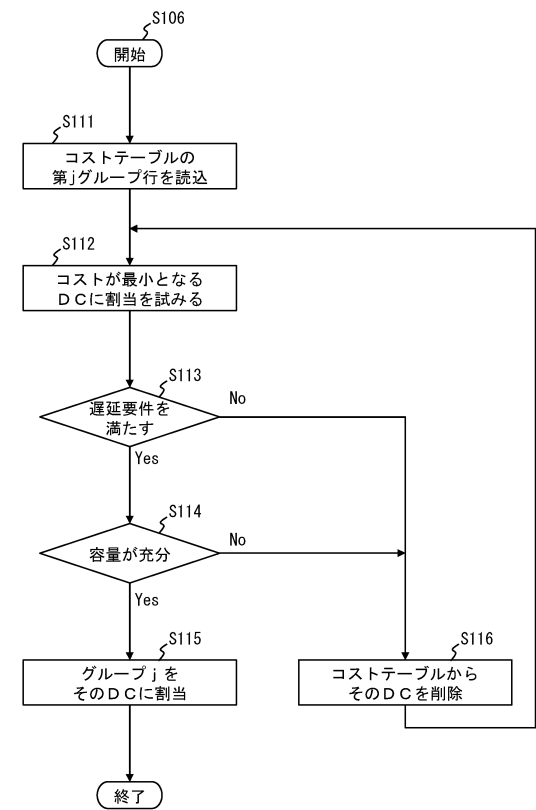
20

30

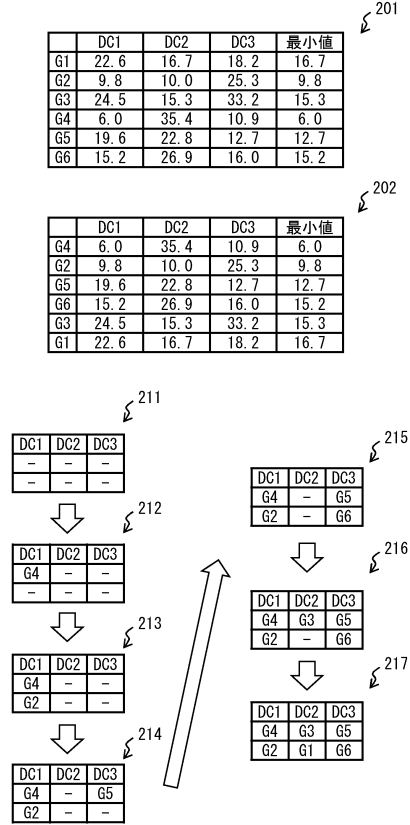
40

50

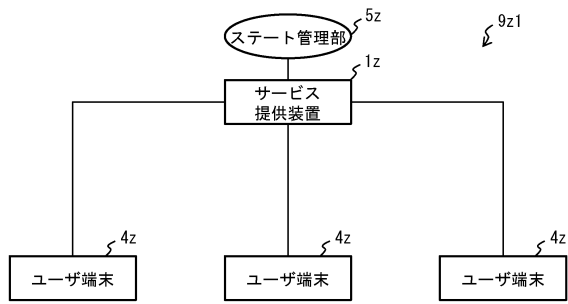
【図 5】



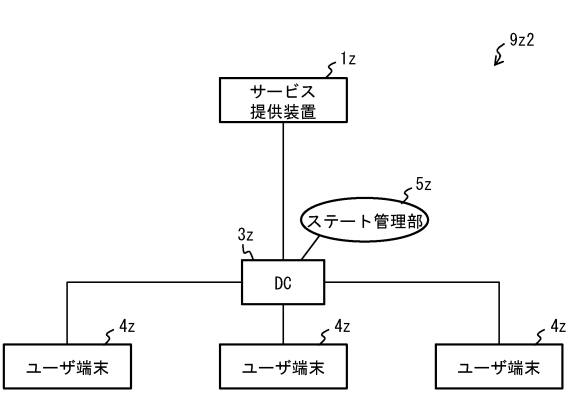
【図 6】



【図 7】



【図 8】



10

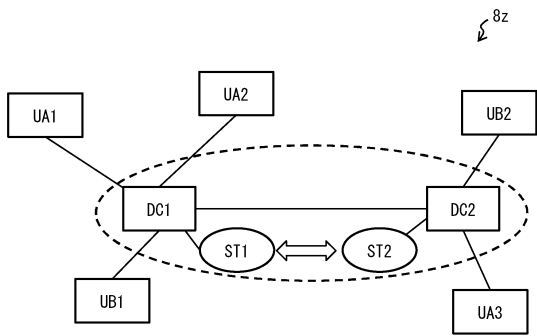
20

30

40

50

【図 9】



10

20

30

40

50

フロントページの続き

- (56)参考文献 特表平9-508255(JP,A)
佐藤諒平ほか, ICNにおける低コストなキャッシュ配置最適化手法の一検討, 電子情報通信学会技術研究報告, 日本, 一般社団法人電子情報通信学会, 2020年02月27日, Vol. 119 No. 460, 455~460ページ
- (58)調査した分野 (Int.Cl., DB名)
G06F 9/50