(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第6191695号
(P6191695)
(45)発行日　平成29年9月6日(2017.9.6)　(24)登録日　平成29年8月18日(2017.8.18)

(51)Int.Cl.　　F I
G06F 9/50 (2006.01)　　G06F 9/46 462Z
G06F 9/46 (2006.01)　　G06F 9/46 350

請求項の数 9　(全 37 頁)

(21)出願番号　特願2015-530673(P2015-530673)
(86)(22)出願日　平成26年6月24日(2014.6.24)
(86)国際出願番号　PCT/JP2014/003383
(87)国際公開番号　WO2015/019538
(87)国際公開日　平成27年2月12日(2015.2.12)
審査請求日　平成29年5月11日(2017.5.11)
(31)優先権主張番号　特願2013-162456(P2013-162456)
(32)優先日　平成25年8月5日(2013.8.5)
(33)優先権主張国　日本国(JP)

(73)特許権者　000004237
日本電気株式会社
東京都港区芝五丁目７番１号
(74)代理人　100103090
弁理士　岩壁　冬樹
(74)代理人　100124501
弁理士　塩川　誠人
(72)発明者　小泉　清一
東京都港区芝五丁目７番１号　日本電気株式会社内

審査官　大桃　由紀雄

最終頁に続く

(54)【発明の名称】仮想リソース制御システムおよび仮想リソース制御方法

(57)【特許請求の範囲】
【請求項１】
サービスを提供するサービスシステムと一対一に対応し、対応するサービスシステム内の個々のノードに対する仮想リソースの割当量を定めるサービス管理装置と、
サービス管理装置に対応する１つのサービスシステム全体に対する仮想リソースの割当量の不足量または余剰量を表すリソース過不足量を前記サービス管理装置から受け付け、前記サービス管理装置に対応する１つのサービスシステム全体に対する仮想リソースの割当量を前記サービス管理装置に通知するハブ装置と、
各サービスシステム全体に対する仮想リソースの割当量を算出するリソース管理装置とを備え、
各サービス管理装置は、
自装置に対応するサービスシステム内の各ノードの入力および出力を処理順に表現するサービスモデルを保持するモデル保持手段と、
前記サービスシステム内の各ノードに関して、仮想リソースの割当量と、１つのリクエストに対して消費する仮想リソースの量である単位リソース消費量と、平均処理時間とを測定し、前記サービスシステムで生じたリクエストを表すログを取得するモニタリング手段と、
各ノードの仮想リソースの割当量、単位リソース消費量、および平均処理時間と、前記サービスモデルとに基づいて、ノードに入力が生じたときにおける当該ノードの未使用リソース量および使用中のリソース量を表現するリソースモデルを生成し、前記サービスモ

デルと前記リソースモデルとを組み合わせたハイブリッドモデルを生成するモデル生成手段と、
前記ハイブリッドモデルと、前記ログとを用いて、前記サービスシステム内の各ノードのリソース消費状況のシミュレーションを実行して、ノード毎に、仮想リソースの残量の最小値である最小リソース残量と、仮想リソースの不足量の最大値である最大リソース不足量とを算出し、ノード毎の最小リソース残量および最大リソース不足量に基づいて前記リソース過不足量を算出し、当該リソース過不足量を前記ハブ装置に通知し、前記ハブ装置から、当該リソース過不足量が示す不足量または余剰量を解消した、当該サービス管理装置に対応する１つのサービスシステム全体に対する仮想リソースの割当量の通知を受けるリソース過不足量算出手段と、
ノード毎に、最小リソース残量または最大リソース不足量が解消されるように、仮想リソースの割当量を更新する仮想リソース割当量更新手段とを含み、
前記ハブ装置は、
各サービス管理装置から通知されたリソース過不足量を保持するリソース過不足量保持手段と、
一定期間毎に、各サービス管理装置から通知されたリソース過不足量に基づいて、各サービスシステム全体に対する仮想リソースの割当量の不足量または余剰量を表す全体リソース過不足量を算出し、当該全体リソース過不足量をリソース管理装置に通知し、前記リソース管理装置から、当該全体リソース過不足量を解消した、各サービスシステム全体に対する仮想リソースの割当量の通知を受ける全体リソース過不足量算出手段と、
リソース過不足量が示す不足量または余剰量を解消した、サービス管理装置に対応する１つのサービスシステム全体に対する仮想リソースの割当量を算出し、当該割当量を前記リソース過不足量の送信元のサービス管理装置に通知するシステム別仮想リソース割当量算出手段とを含み、
前記リソース管理装置は、
前記ハブ装置から通知された全体リソース過不足量を保持する全体リソース過不足量保持手段と、
一定期間毎に、前記全体リソース過不足量を確認し、前記全体リソース過不足量を解消した、各サービスシステム全体に対する仮想リソースの割当量を算出し、当該割当量を前記ハブ装置に通知する全体仮想リソース割当量算出手段とを含む
ことを特徴とする仮想リソース制御システム。
【請求項２】
仮想リソースの種別毎にリソース管理装置を備える
請求項１に記載の仮想リソース制御システム。
【請求項３】
リソース過不足量算出手段は、仮想リソースの種別毎に、最小リソース残量、最大リソース不足量、およびリソース過不足量を算出し、
仮想リソース割当量更新手段は、仮想リソースの種別毎に、仮想リソースの割当量を更新し、
リソース過不足量保持手段は、各サービス管理装置から通知されたリソース過不足量を仮想リソースの種別毎に保持し、
全体リソース過不足量算出手段は、仮想リソースの種別毎に、全体リソース過不足量を算出し、当該全体リソース過不足量を仮想リソースの種別に応じたリソース管理装置に通知し、
システム別仮想リソース割当量算出手段は、サービス管理装置に対応する１つのサービスシステム全体に対する仮想リソースの割当量を、仮想リソースの種別毎に算出する
請求項２に記載の仮想リソース制御システム。
【請求項４】
モデル保持手段は、サービスシステム内の各ノードの入力および出力を処理順に表現するとともに、ノードの出力の分岐および集約を表現するサービスモデルを保持する

請求項１から請求項３のうちのいずれか１項に記載の仮想リソース制御システム。
【請求項５】
サービスを提供するサービスシステムと一対一に対応し、対応するサービスシステム内の個々のノードに対する仮想リソースの割当量を定めるサービス管理装置と、サービス管理装置に対応する１つのサービスシステム全体に対する仮想リソースの割当量の不足量または余剰量を表すリソース過不足量を前記サービス管理装置から受け付け、前記サービス管理装置に対応する１つのサービスシステム全体に対する仮想リソースの割当量を前記サービス管理装置に通知するハブ装置と、各サービスシステム全体に対する仮想リソースの割当量を算出するリソース管理装置とを用いた仮想リソース制御方法であって、
各サービス管理装置が、
自装置に対応するサービスシステム内の各ノードの入力および出力を処理順に表現するサービスモデルを保持し、
前記サービスシステム内の各ノードに関して、仮想リソースの割当量と、１つのリクエストに対して消費する仮想リソースの量である単位リソース消費量と、平均処理時間とを測定し、前記サービスシステムで生じたリクエストを表すログを取得し、
各ノードの仮想リソースの割当量、単位リソース消費量、および平均処理時間と、前記サービスモデルとに基づいて、ノードに入力が生じたときにおける当該ノードの未使用リソース量および使用中のリソース量を表現するリソースモデルを生成し、前記サービスモデルと前記リソースモデルとを組み合わせたハイブリッドモデルを生成し、
前記ハイブリッドモデルと、前記ログとを用いて、前記サービスシステム内の各ノードのリソース消費状況のシミュレーションを実行して、ノード毎に、仮想リソースの残量の最小値である最小リソース残量と、仮想リソースの不足量の最大値である最大リソース不足量とを算出し、ノード毎の最小リソース残量および最大リソース不足量に基づいて前記リソース過不足量を算出し、当該リソース過不足量を前記ハブ装置に通知し、
前記ハブ装置が、
各サービス管理装置から通知されたリソース過不足量を保持し、
一定期間毎に、各サービス管理装置から通知されたリソース過不足量に基づいて、各サービスシステム全体に対する仮想リソースの割当量の不足量または余剰量を表す全体リソース過不足量を算出し、当該全体リソース過不足量をリソース管理装置に通知し、
前記リソース管理装置が、
前記ハブ装置から通知された全体リソース過不足量を保持し、
一定期間毎に、前記全体リソース過不足量を確認し、前記全体リソース過不足量を解消した、各サービスシステム全体に対する仮想リソースの割当量を算出し、当該割当量を前記ハブ装置に通知し、
前記ハブ装置が、
前記リソース管理装置から、各サービスシステム全体に対する仮想リソースの割当量の通知を受け、
リソース過不足量が示す不足量または余剰量を解消した、サービス管理装置に対応する１つのサービスシステム全体に対する仮想リソースの割当量を算出し、当該割当量を前記リソース過不足量の送信元のサービス管理装置に通知し、
各サービス管理装置が、
自装置に対応する１つのサービスシステム全体に対する仮想リソースの割当量の通知を受け、
ノード毎に、最小リソース残量または最大リソース不足量が解消されるように、仮想リソースの割当量を更新する
ことを特徴とする仮想リソース制御方法。
【請求項６】
サービスを提供するサービスシステムと一対一に対応し、対応するサービスシステム内の個々のノードに対する仮想リソースの割当量を定めるサービス管理装置であって、
当該サービス管理装置に対応するサービスシステム内の各ノードの入力および出力を処

理順に表現するサービスモデルを保持するモデル保持手段と、
前記サービスシステム内の各ノードに関して、仮想リソースの割当量と、１つのリクエストに対して消費する仮想リソースの量である単位リソース消費量と、平均処理時間とを測定し、前記サービスシステムで生じたリクエストを表すログを取得するモニタリング手段と、
各ノードの仮想リソースの割当量、単位リソース消費量、および平均処理時間と、前記サービスモデルとに基づいて、ノードに入力が生じたときにおける当該ノードの未使用リソース量および使用中のリソース量を表現するリソースモデルを生成し、前記サービスモデルと前記リソースモデルとを組み合わせたハイブリッドモデルを生成するモデル生成手段と、
前記ハイブリッドモデルと、前記ログとを用いて、前記サービスシステム内の各ノードのリソース消費状況のシミュレーションを実行して、ノード毎に、仮想リソースの残量の最小値である最小リソース残量と、仮想リソースの不足量の最大値である最大リソース不足量とを算出し、ノード毎の最小リソース残量および最大リソース不足量に基づいて、当該サービス管理装置に対応する１つのサービスシステム全体に対する仮想リソースの割当量の不足量または余剰量を表すリソース過不足量を算出し、当該サービス管理装置に対応する１つのサービスシステム全体に対する仮想リソースの割当量を定めるハブ装置に前記リソース過不足量を通知し、前記ハブ装置から、当該リソース過不足量が示す不足量または余剰量を解消した、当該サービス管理装置に対応する１つのサービスシステム全体に対する仮想リソースの割当量の通知を受けるリソース過不足量算出手段と、
ノード毎に、最小リソース残量または最大リソース不足量が解消されるように、仮想リソースの割当量を更新する仮想リソース割当量更新手段とを備える
ことを特徴とするサービス管理装置。
【請求項７】
サービスを提供するサービスシステムと一対一に対応し、対応するサービスシステム内の個々のノードに対する仮想リソースの割当量を定めるサービス管理装置、および、各サービスシステム全体に対する仮想リソースの割当量を算出するリソース管理装置とともに用いられ、サービス管理装置に対応する１つのサービスシステム全体に対する仮想リソースの割当量の不足量または余剰量を表すリソース過不足量を前記サービス管理装置から受け付け、前記サービス管理装置に対応する１つのサービスシステム全体に対する仮想リソースの割当量を前記サービス管理装置に通知するハブ装置であって、
各サービス管理装置から通知されたリソース過不足量を保持するリソース過不足量保持手段と、
一定期間毎に、各サービス管理装置から通知されたリソース過不足量に基づいて、各サービスシステム全体に対する仮想リソースの割当量の不足量または余剰量を表す全体リソース過不足量を算出し、当該全体リソース過不足量をリソース管理装置に通知し、前記リソース管理装置から、当該全体リソース過不足量を解消した、各サービスシステム全体に対する仮想リソースの割当量の通知を受ける全体リソース過不足量算出手段と、
リソース過不足量が示す不足量または余剰量を解消した、サービス管理装置に対応する１つのサービスシステム全体に対する仮想リソースの割当量を算出し、当該割当量を前記リソース過不足量の送信元のサービス管理装置に通知するシステム別仮想リソース割当量算出手段とを備える
ことを特徴とするハブ装置。
【請求項８】
サービスを提供するサービスシステムと一対一に対応し、対応するサービスシステム内の個々のノードに対する仮想リソースの割当量を定めるコンピュータに搭載されるサービス管理装置用プログラムであって、
前記コンピュータに、
前記コンピュータに対応するサービスシステム内の各ノードの入力および出力を処理順に表現するサービスモデルを保持するモデル保持処理、

前記サービスシステム内の各ノードに関して、仮想リソースの割当量と、１つのリクエストに対して消費する仮想リソースの量である単位リソース消費量と、平均処理時間とを測定し、前記サービスシステムで生じたリクエストを表すログを取得するモニタリング処理、

各ノードの仮想リソースの割当量、単位リソース消費量、および平均処理時間と、前記サービスモデルとに基づいて、ノードに入力が生じたときにおける当該ノードの未使用リソース量および使用中のリソース量を表現するリソースモデルを生成し、前記サービスモデルと前記リソースモデルとを組み合わせたハイブリッドモデルを生成するモデル生成処理、

前記ハイブリッドモデルと、前記ログとを用いて、前記サービスシステム内の各ノードのリソース消費状況のシミュレーションを実行して、ノード毎に、仮想リソースの残量の最小値である最小リソース残量と、仮想リソースの不足量の最大値である最大リソース不足量とを算出し、ノード毎の最小リソース残量および最大リソース不足量に基づいて、前記コンピュータに対応する１つのサービスシステム全体に対する仮想リソースの割当量の不足量または余剰量を表すリソース過不足量を算出し、前記コンピュータに対応する１つのサービスシステム全体に対する仮想リソースの割当量を定めるハブ装置に前記リソース過不足量を通知し、前記ハブ装置から、当該リソース過不足量が示す不足量または余剰量を解消した、前記コンピュータに対応する１つのサービスシステム全体に対する仮想リソースの割当量の通知を受けるリソース過不足量算出処理、および、

ノード毎に、最小リソース残量または最大リソース不足量が解消されるように、仮想リソースの割当量を更新する仮想リソース割当量更新処理

を実行させるためのサービス管理装置用プログラム。

【請求項９】

サービスを提供するサービスシステムと一対一に対応し、対応するサービスシステム内の個々のノードに対する仮想リソースの割当量を定めるサービス管理装置、および、各サービスシステム全体に対する仮想リソースの割当量を算出するリソース管理装置とともに用いられ、サービス管理装置に対応する１つのサービスシステム全体に対する仮想リソースの割当量の不足量または余剰量を表すリソース過不足量を前記サービス管理装置から受け付け、前記サービス管理装置に対応する１つのサービスシステム全体に対する仮想リソースの割当量を前記サービス管理装置に通知するコンピュータに搭載されるハブ装置用プログラムであって、

前記コンピュータに、

各サービス管理装置から通知されたリソース過不足量を保持するリソース過不足量保持処理、

一定期間毎に、各サービス管理装置から通知されたリソース過不足量に基づいて、各サービスシステム全体に対する仮想リソースの割当量の不足量または余剰量を表す全体リソース過不足量を算出し、当該全体リソース過不足量をリソース管理装置に通知し、前記リソース管理装置から、当該全体リソース過不足量を解消した、各サービスシステム全体に対する仮想リソースの割当量の通知を受ける全体リソース過不足量算出処理、および、

リソース過不足量が示す不足量または余剰量を解消した、サービス管理装置に対応する１つのサービスシステム全体に対する仮想リソースの割当量を算出し、当該割当量を前記リソース過不足量の送信元のサービス管理装置に通知するシステム別仮想リソース割当量算出処理

を実行させるためのハブ装置用プログラム。

【発明の詳細な説明】

【技術分野】

【０００１】

本発明は、複数のサービスシステムに対して仮想リソースの割当量の制御を行う仮想リソース制御システム、仮想リソース制御方法、および、その仮想リソース制御システム、仮想リソース制御方法に適用されるサービス管理装置、ハブ装置、サービス管理装置用プ

ログラム、ハブ装置用プログラムに関する。
【背景技術】
【０００２】
ＩａａＳ（Infrastructure as a Service）等のクラウドプラットフォーム上では、利用者にサービスを提供する複数のサービスシステムが存在する。そのようなサービスシステムの種類は、例えば、ＥＣ（Electronic Commerce）システムや動画配信システム等、多様である。個々のサービスシステムは、Ｗｅｂサーバ、アプリケーションサーバ、データベースサーバ等の複数のノードを含む。以下、アプリケーションサーバを、Ａｐｐサーバと記す場合がある。また、データベースサーバを、ＤＢサーバと記す場合がある。
【０００３】
特許文献１には、仮想リソースの運用管理システムが記載されている。特許文献１に記載の運用管理システムは、テンプレートによって構成が定められた仮想システムを制御対象にしている。
【０００４】
また、特許文献２に記載された仮想サーバリソース調整システムは、それぞれの仮想サーバについて、現在割り当てられているリソース量から削減可能なリソース量である空きリソース量を算出し、各空きリソース量を合計して総空きリソース量を算出する。そして、この仮想サーバリソース調整システムは、高負荷が発生した仮想サーバの負荷から、追加が必要なリソース量を算出し、総空きリソース量と比較する。この仮想サーバリソース調整システムは、総空きリソース量の方が多ければ、リソース追加が可能であると判断する。また、特許文献２では、リソースの例として、ＣＰＵ（Central Processing Unit ）、メモリ、ディスクＩ／Ｏ（Input/Output）、ネットワークＩ／Ｏが挙げられている。
【０００５】
また、特許文献３には、クライアントとサーバとを備え、データリソースが、いくつかのサーバに渡って区分された構成が開示されている。また、特許文献３には、サーバの負荷バランスをとることが記載されている。
【先行技術文献】
【特許文献】
【０００６】
【特許文献１】特開２０１１－８１５７９号公報（段落００７２、図１８）
【特許文献２】特開２０１０－３３２９２号公報（段落００１６、００２８～００２９、図５、図６）
【特許文献３】特表２００６－５２２９６１号公報（段落００３３、００４４、００６３～００７３、００８２、００８３、図２、図４）
【発明の概要】
【発明が解決しようとする課題】
【０００７】
クラウドプラットフォーム上に存在する各サービスシステムは、それぞれ構成や振る舞いが異なる。そのため、サービスシステムの稼働状況に合ったリソースを動的に割り当て、高いサービス品質を維持するためには、サービスシステムにおけるリソース消費状態を把握して適切なリソース割り当てを行う必要がある。このようなきめ細かな仮想リソースの割り当て制御は、管理対象の構成が静的な場合には適用しやすい。しかし、クラウド環境のようにサービスシステムの追加や削除が頻繁に発生する環境では、このようなきめ細かな仮想リソースの割り当て制御は困難であった。
【０００８】
また、特許文献１に記載の技術は、テンプレートによって構成が定められた仮想システムを制御対象にしているので、サービスシステムの追加や削除が頻繁に発生する環境には適さない。
【０００９】
サービスシステムの追加や削除が生じる環境において、個々のサービスシステムに対し

て仮想リソースを適切に割り当てられることが好ましい。

【００１０】

そこで、本発明は、サービスシステムの追加や削除が生じる環境において、個々のサービスシステムに対する仮想リソースの割当量を適切に定めることができる仮想リソース制御システム、仮想リソース制御方法、および、その仮想リソース制御システム、仮想リソース制御方法に適用されるサービス管理装置、ハブ装置、サービス管理装置用プログラム、ハブ装置用プログラムを提供することを目的とする。

【課題を解決するための手段】

【００１１】

本発明による仮想リソース制御システムは、サービスを提供するサービスシステムと一対一に対応し、対応するサービスシステム内の個々のノードに対する仮想リソースの割当量を定めるサービス管理装置と、サービス管理装置に対応する１つのサービスシステム全体に対する仮想リソースの割当量の不足量または余剰量を表すリソース過不足量をサービス管理装置から受け付け、サービス管理装置に対応する１つのサービスシステム全体に対する仮想リソースの割当量をサービス管理装置に通知するハブ装置と、各サービスシステム全体に対する仮想リソースの割当量を算出するリソース管理装置とを備え、各サービス管理装置は、自装置に対応するサービスシステム内の各ノードの入力および出力を処理順に表現するサービスモデルを保持するモデル保持手段と、サービスシステム内の各ノードに関して、仮想リソースの割当量と、１つのリクエストに対して消費する仮想リソースの量である単位リソース消費量と、平均処理時間とを測定し、サービスシステムで生じたリクエストを表すログを取得するモニタリング手段と、各ノードの仮想リソースの割当量、単位リソース消費量、および平均処理時間と、サービスモデルとに基づいて、ノードに入力が生じたときにおける当該ノードの未使用リソース量および使用中のリソース量を表現するリソースモデルを生成し、サービスモデルとリソースモデルとを組み合わせたハイブリッドモデルを生成するモデル生成手段と、ハイブリッドモデルと、ログとを用いて、サービスシステム内の各ノードのリソース消費状況のシミュレーションを実行して、ノード毎に、仮想リソースの残量の最小値である最小リソース残量と、仮想リソースの不足量の最大値である最大リソース不足量とを算出し、ノード毎の最小リソース残量および最大リソース不足量に基づいてリソース過不足量を算出し、当該リソース過不足量をハブ装置に通知し、ハブ装置から、当該リソース過不足量が示す不足量または余剰量を解消した、当該サービス管理装置に対応する１つのサービスシステム全体に対する仮想リソースの割当量の通知を受けるリソース過不足量算出手段と、ノード毎に、最小リソース残量または最大リソース不足量が解消されるように、仮想リソースの割当量を更新する仮想リソース割当量更新手段とを含み、ハブ装置は、各サービス管理装置から通知されたリソース過不足量を保持するリソース過不足量保持手段と、一定期間毎に、各サービス管理装置から通知されたリソース過不足量に基づいて、各サービスシステム全体に対する仮想リソースの割当量の不足量または余剰量を表す全体リソース過不足量を算出し、当該全体リソース過不足量をリソース管理装置に通知し、リソース管理装置から、当該全体リソース過不足量を解消した、各サービスシステム全体に対する仮想リソースの割当量の通知を受ける全体リソース過不足量算出手段と、リソース過不足量が示す不足量または余剰量を解消した、サービス管理装置に対応する１つのサービスシステム全体に対する仮想リソースの割当量を算出し、当該割当量をリソース過不足量の送信元のサービス管理装置に通知するシステム別仮想リソース割当量算出手段とを含み、リソース管理装置は、ハブ装置から通知された全体リソース過不足量を保持する全体リソース過不足量保持手段と、一定期間毎に、全体リソース過不足量を確認し、全体リソース過不足量を解消した、各サービスシステム全体に対する仮想リソースの割当量を算出し、当該割当量をハブ装置に通知する全体仮想リソース割当量算出手段とを含むことを特徴とする。

【００１２】

また、本発明による仮想リソース制御方法は、サービスを提供するサービスシステムと一対一に対応し、対応するサービスシステム内の個々のノードに対する仮想リソースの割

当量を定めるサービス管理装置と、サービス管理装置に対応する１つのサービスシステム全体に対する仮想リソースの割当量の不足量または余剰量を表すリソース過不足量をサービス管理装置から受け付け、サービス管理装置に対応する１つのサービスシステム全体に対する仮想リソースの割当量をサービス管理装置に通知するハブ装置と、各サービスシステム全体に対する仮想リソースの割当量を算出するリソース管理装置とを用いた仮想リソース制御方法であって、各サービス管理装置が、自装置に対応するサービスシステム内の各ノードの入力および出力を処理順に表現するサービスモデルを保持し、サービスシステム内の各ノードに関して、仮想リソースの割当量と、１つのリクエストに対して消費する仮想リソースの量である単位リソース消費量と、平均処理時間とを測定し、サービスシステムで生じたリクエストを表すログを取得し、各ノードの仮想リソースの割当量、単位リソース消費量、および平均処理時間と、サービスモデルとに基づいて、ノードに入力が生じたときにおける当該ノードの未使用リソース量および使用中のリソース量を表現するリソースモデルを生成し、サービスモデルとリソースモデルとを組み合わせたハイブリッドモデルを生成し、ハイブリッドモデルと、ログとを用いて、サービスシステム内の各ノードのリソース消費状況のシミュレーションを実行して、ノード毎に、仮想リソースの残量の最小値である最小リソース残量と、仮想リソースの不足量の最大値である最大リソース不足量とを算出し、ノード毎の最小リソース残量および最大リソース不足量に基づいてリソース過不足量を算出し、当該リソース過不足量をハブ装置に通知し、ハブ装置が、各サービス管理装置から通知されたリソース過不足量を保持し、一定期間毎に、各サービス管理装置から通知されたリソース過不足量に基づいて、各サービスシステム全体に対する仮想リソースの割当量の不足量または余剰量を表す全体リソース過不足量を算出し、当該全体リソース過不足量をリソース管理装置に通知し、リソース管理装置が、ハブ装置から通知された全体リソース過不足量を保持し、一定期間毎に、全体リソース過不足量を確認し、全体リソース過不足量を解消した、各サービスシステム全体に対する仮想リソースの割当量を算出し、当該割当量をハブ装置に通知し、ハブ装置が、リソース管理装置から、各サービスシステム全体に対する仮想リソースの割当量の通知を受け、リソース過不足量が示す不足量または余剰量を解消した、サービス管理装置に対応する１つのサービスシステム全体に対する仮想リソースの割当量を算出し、当該割当量をリソース過不足量の送信元のサービス管理装置に通知し、各サービス管理装置が、自装置に対応する１つのサービスシステム全体に対する仮想リソースの割当量の通知を受け、ノード毎に、最小リソース残量または最大リソース不足量が解消されるように、仮想リソースの割当量を更新することを特徴とする。

【００１３】

また、本発明によるサービス管理装置は、サービスを提供するサービスシステムと一対一に対応し、対応するサービスシステム内の個々のノードに対する仮想リソースの割当量を定めるサービス管理装置であって、当該サービス管理装置に対応するサービスシステム内の各ノードの入力および出力を処理順に表現するサービスモデルを保持するモデル保持手段と、サービスシステム内の各ノードに関して、仮想リソースの割当量と、１つのリクエストに対して消費する仮想リソースの量である単位リソース消費量と、平均処理時間とを測定し、サービスシステムで生じたリクエストを表すログを取得するモニタリング手段と、各ノードの仮想リソースの割当量、単位リソース消費量、および平均処理時間と、サービスモデルとに基づいて、ノードに入力が生じたときにおける当該ノードの未使用リソース量および使用中のリソース量を表現するリソースモデルを生成し、サービスモデルとリソースモデルとを組み合わせたハイブリッドモデルを生成するモデル生成手段と、ハイブリッドモデルと、ログとを用いて、サービスシステム内の各ノードのリソース消費状況のシミュレーションを実行して、ノード毎に、仮想リソースの残量の最小値である最小リソース残量と、仮想リソースの不足量の最大値である最大リソース不足量とを算出し、ノード毎の最小リソース残量および最大リソース不足量に基づいて、当該サービス管理装置に対応する１つのサービスシステム全体に対する仮想リソースの割当量の不足量または余剰量を表すリソース過不足量を算出し、当該サービス管理装置に対応する１つのサービス

システム全体に対する仮想リソースの割当量を定めるハブ装置にリソース過不足量を通知し、ハブ装置から、当該リソース過不足量が示す不足量または余剰量を解消した、当該サービス管理装置に対応する１つのサービスシステム全体に対する仮想リソースの割当量の通知を受けるリソース過不足量算出手段と、ノード毎に、最小リソース残量または最大リソース不足量が解消されるように、仮想リソースの割当量を更新する仮想リソース割当量更新手段とを備えることを特徴とする。

【００１４】

また、本発明によるハブ装置は、サービスを提供するサービスシステムと一対一に対応し、対応するサービスシステム内の個々のノードに対する仮想リソースの割当量を定めるサービス管理装置、および、各サービスシステム全体に対する仮想リソースの割当量を算出するリソース管理装置とともに用いられ、サービス管理装置に対応する１つのサービスシステム全体に対する仮想リソースの割当量の不足量または余剰量を表すリソース過不足量をサービス管理装置から受け付け、サービス管理装置に対応する１つのサービスシステム全体に対する仮想リソースの割当量をサービス管理装置に通知するハブ装置であって、各サービス管理装置から通知されたリソース過不足量を保持するリソース過不足量保持手段と、一定期間毎に、各サービス管理装置から通知されたリソース過不足量に基づいて、各サービスシステム全体に対する仮想リソースの割当量の不足量または余剰量を表す全体リソース過不足量を算出し、当該全体リソース過不足量をリソース管理装置に通知し、リソース管理装置から、当該全体リソース過不足量を解消した、各サービスシステム全体に対する仮想リソースの割当量の通知を受ける全体リソース過不足量算出手段と、リソース過不足量が示す不足量または余剰量を解消した、サービス管理装置に対応する１つのサービスシステム全体に対する仮想リソースの割当量を算出し、当該割当量をリソース過不足量の送信元のサービス管理装置に通知するシステム別仮想リソース割当量算出手段とを備えることを特徴とする。

【００１５】

また、本発明によるサービス管理装置用プログラムは、サービスを提供するサービスシステムと一対一に対応し、対応するサービスシステム内の個々のノードに対する仮想リソースの割当量を定めるコンピュータに搭載されるサービス管理装置用プログラムであって、コンピュータに、コンピュータに対応するサービスシステム内の各ノードの入力および出力を処理順に表現するサービスモデルを保持するモデル保持処理、サービスシステム内の各ノードに関して、仮想リソースの割当量と、１つのリクエストに対して消費する仮想リソースの量である単位リソース消費量と、平均処理時間とを測定し、サービスシステムで生じたリクエストを表すログを取得するモニタリング処理、各ノードの仮想リソースの割当量、単位リソース消費量、および平均処理時間と、サービスモデルとに基づいて、ノードに入力が生じたときにおける当該ノードの未使用リソース量および使用中のリソース量を表現するリソースモデルを生成し、サービスモデルとリソースモデルとを組み合わせたハイブリッドモデルを生成するモデル生成処理、ハイブリッドモデルと、ログとを用いて、サービスシステム内の各ノードのリソース消費状況のシミュレーションを実行して、ノード毎に、仮想リソースの残量の最小値である最小リソース残量と、仮想リソースの不足量の最大値である最大リソース不足量とを算出し、ノード毎の最小リソース残量および最大リソース不足量に基づいて、コンピュータに対応する１つのサービスシステム全体に対する仮想リソースの割当量の不足量または余剰量を表すリソース過不足量を算出し、コンピュータに対応する１つのサービスシステム全体に対する仮想リソースの割当量を定めるハブ装置にリソース過不足量を通知し、ハブ装置から、当該リソース過不足量が示す不足量または余剰量を解消した、コンピュータに対応する１つのサービスシステム全体に対する仮想リソースの割当量の通知を受けるリソース過不足量算出処理、および、ノード毎に、最小リソース残量または最大リソース不足量が解消されるように、仮想リソースの割当量を更新する仮想リソース割当量更新処理を実行させることを特徴とする。

【００１６】

また、本発明によるハブ装置用プログラムは、サービスを提供するサービスシステムと

一対一に対応し、対応するサービスシステム内の個々のノードに対する仮想リソースの割当量を定めるサービス管理装置、および、各サービスシステム全体に対する仮想リソースの割当量を算出するリソース管理装置とともに用いられ、サービス管理装置に対応する１つのサービスシステム全体に対する仮想リソースの割当量の不足量または余剰量を表すリソース過不足量をサービス管理装置から受け付け、サービス管理装置に対応する１つのサービスシステム全体に対する仮想リソースの割当量をサービス管理装置に通知するコンピュータに搭載されるハブ装置用プログラムであって、コンピュータに、各サービス管理装置から通知されたリソース過不足量を保持するリソース過不足量保持処理、一定期間毎に、各サービス管理装置から通知されたリソース過不足量に基づいて、各サービスシステム全体に対する仮想リソースの割当量の不足量または余剰量を表す全体リソース過不足量を算出し、当該全体リソース過不足量をリソース管理装置に通知し、リソース管理装置から、当該全体リソース過不足量を解消した、各サービスシステム全体に対する仮想リソースの割当量の通知を受ける全体リソース過不足量算出処理、および、リソース過不足量が示す不足量または余剰量を解消した、サービス管理装置に対応する１つのサービスシステム全体に対する仮想リソースの割当量を算出し、当該割当量をリソース過不足量の送信元のサービス管理装置に通知するシステム別仮想リソース割当量算出処理を実行させることを特徴とする。

【発明の効果】

【００１７】

本発明によれば、サービスシステムの追加や削除が生じる環境において、個々のサービスシステムに対する仮想リソースの割当量を適切に定めることができる。

【図面の簡単な説明】

【００１８】

【図１】本発明の仮想リソース制御システムの構成例を示すブロック図である。

【図２】サービスシステムの例を示す模式図である。

【図３】ノード＿リソーステーブルの例を示す説明図である。

【図４】サービス＿ログテーブルの例を示す説明図である。

【図５】サービス＿リソース配分テーブルの例を示す説明図である。

【図６】サービス＿モデルテーブルの例を示す説明図である。

【図７】サービスモデルの例を模式的に示す説明図である。

【図８】リソースモデルの例を模式的に示す説明図である。

【図９】ハイブリッドモデルの例を示す説明図である。

【図１０】ハブ＿サービス＿リソース配分テーブルの例を示す説明図である。

【図１１】ハブ＿リソース配分テーブルの例を示す説明図である。

【図１２】プール＿ハブ＿リソース配分テーブルの例を示す説明図である。

【図１３】プール＿リソース容量テーブルの例を示す説明図である。

【図１４】プール＿ハブ＿リソース配分テーブルの例を示す説明図である。

【図１５】プール＿リソース容量テーブルの例を示す説明図である。

【図１６】サービス管理装置の処理経過の例を示すフローチャートである。

【図１７】ハブ装置の処理経過の例を示すフローチャートである。

【図１８】リソースプール管理装置の処理経過の例を示すフローチャートである。

【図１９】図３に示すノード＿リソーステーブルに各ノードの最大リソース不足量および最小リソース残量を追加した状態の例を示す説明図である。

【図２０】図１９に例示するノード＿リソーステーブルを更新した状態を示す説明図である。

【図２１】図１１に示すハブ＿リソース配分テーブルを更新した状態を示す説明図である。

【図２２】図１０に示すハブ＿サービス＿リソース配分テーブルを更新した状態を示す説明図である。

【図２３】図１２に例示するプール＿ハブ＿リソース配分テーブルを更新した状態を示す

説明図である。
【図２４】図１３に例示するプール＿リソース容量テーブルを更新した状態を示す説明図である。
【図２５】図１４に例示するプール＿ハブ＿リソース配分テーブルを更新した状態を示す説明図である。
【図２６】図１５に例示するプール＿リソース容量テーブルを更新した状態を示す説明図である。
【図２７】本発明の主要部を示すブロック図である。
【発明を実施するための形態】
【００１９】
以下、本発明の実施形態を図面を参照して説明する。
【００２０】
図１は、本発明の仮想リソース制御システムの構成例を示すブロック図である。本発明の仮想リソース制御システム１は、サービス管理装置２００，２１０と、ハブ装置３００と、リソースプール管理装置４００，４１０とを備える。仮想リソース制御システム１は、それぞれのサービスシステム１００，１１０の稼働状況に応じて、サービスシステム１００，１１０に対する仮想リソース（例えば、仮想ＣＰＵや仮想ＲＡＭ）の割当量を定める。
【００２１】
サービスシステム１００，１１０は、仮想リソースの割り当て対象である。個々のサービスシステム１００，１１０は、それぞれ、利用者に対してサービス提供処理を実行する。サービスの種類は、サービスシステム毎に異なる。また、サービスシステム１００は、例えば、Ｗｅｂサーバ１０１と、Ａｐｐサーバ１０２と、ＤＢサーバ１０３とを備える。サービスシステム１１０も、同様のサーバ群を備えるが、具体的な構成は、サービスシステム毎に異なっていてよい。また、Ｗｅｂサーバ１０１、Ａｐｐサーバ１０２およびＤＢサーバ１０３以外の装置がサービスシステムに設けられていてもよい。例えば、サービスシステム１００は、図２に例示するように、ファイアウォール１０４とロードバランサ１０５とを備えていてもよい。
【００２２】
以下では、サービスシステムが２つである場合を例にして説明するが、サービスシステムの数は、特に限定されない。
【００２３】
サービス管理装置２００，２１０は、サービスシステム１００，１１０と一対一に対応するように設けられる。従って、サービス管理装置の数は、サービスシステムの数と同数である。各サービス管理装置２００，２１０は、自装置に対応するサービスシステム内の各ノードの単位リソース消費量（１つのリクエストに対して消費するリソース量）や、その各ノードの平均処理時間を測定し、各ノードにおける仮想リソースの余剰分や、不足分を算出する。そして、各サービス管理装置２００，２１０は、自装置に対応するサービスシステム内の各ノードの仮想リソースの割当量を変更する。また、各サービス管理装置２００，２１０は、自装置に対応するサービスシステム内の各ノード全体で、仮想リソースの余剰や不足が生じている場合には、その余剰量や不足量に応じて、自装置に対応するサービスシステムのノード全体に割り当てる仮想リソースの割当量の変更をハブ装置３００に要求する。
【００２４】
サービス管理装置とサービスシステムは一対一に対応しているので、１つのサービス管理装置は１つのサービスに対応していると言える。
【００２５】
以下、サービス管理装置２００を例にして説明するが、各サービス管理装置２００，２１０の構成は同様である。
【００２６】

サービス管理装置２００は、サービス情報記憶装置２０１と、モニタリング部２０２と、ハイブリッドモデル生成部２０３と、ノード＿リソース量推定部２０４と、サービス＿リソース配分検証部２０５と、リソース再配分部２０６と、ハブ連携部２０７とを備える。

【００２７】

サービス情報記憶装置２０１は、ノード＿リソーステーブル、サービス＿ログテーブル、サービス＿リソース配分テーブル、サービス＿モデルテーブルを記憶する。

【００２８】

図３は、ノード＿リソーステーブルの例を示す説明図である。ノード＿リソーステーブルには、サービス管理装置２００に対応するサービスシステム１００内のノードとそのノードに割り当てられる仮想リソースの種別との組み合わせ毎に、ノードＩＤ、リソース型、リソース割当量、リソース消費量（単位リソース消費量）、最小リソース残量、最大リソース不足量、サービスＩＤが対応付けて格納される。リソース型は、仮想リソースの種別である。リソース割当量は、現在、ノードに割り当てられているリソース量である。最小リソース残量は、サービスシステム１００で生成されたログに基づいて、ノードのリソース消費をシミュレートしたときにおける、仮想リソースの残量の最小値である。最大リソース不足量は、そのシミュレーション時における、仮想リソースの不足量の最大値である。最大リソース不足量は、シミュレーションにおいて生じたリクエストの待ち数の最大値と単位リソース消費量との積として求められる。最小リソース残量、最大リソース不足量は、一時的に格納される。サービスＩＤは、サービス管理装置２００に対応するサービスシステム１００が提供するサービスのＩＤである。従って、サービス管理装置２００のノード＿リソーステーブル内において、サービスＩＤは共通である。

【００２９】

なお、本実施形態において、余剰が生じているときの余剰量は正の値で表され、不足が生じているときの不足量は負の値で表される。

【００３０】

また、ノード＿リソーステーブルには、サービスシステム１００内のノード毎に、ノードＩＤ、ノード名、および、そのノードの平均処理時間が格納される（図３の下段参照）。

【００３１】

図４は、サービス＿ログテーブルの例を示す説明図である。サービスシステム１００は、１つのリクエストが生じる毎に１つのログを生成する。このログの情報が、サービス＿ログテーブルに格納される。具体的には、ログＩＤ、サービスＩＤ、タイムスタンプが対応付けて格納される。図４に示す１行分のデータが、サービスシステム１００に生じた１つのリクエストを表す。

【００３２】

図５は、サービス＿リソース配分テーブルの例を示す説明図である。サービス＿リソース配分テーブルには、仮想リソースの種別（リソース型）毎に、サービス管理装置２００に対応するサービスシステム１００内のノード全体に現在割り当てられているリソース量、および、そのノード全体でのリソース過不足量が格納される。また、仮想リソースの種別毎にサービスＩＤも格納される。ただし、ノード＿リソーステーブルと同様に、サービス＿リソース配分テーブルでも、各行のサービスＩＤは共通である。仮想リソースの過不足がなくなるように、ノード＿リソーステーブル（図３参照）内のリソース割当量を更新したときに、サービス＿リソース配分検証部２０５は、そのリソース割当量に対応するリソース過不足量を消去する。

【００３３】

図６は、サービス＿モデルテーブルの例を示す説明図である。サービス＿モデルテーブルには、サービスＩＤと、サービスモデルと、ハイブリッドモデルが対応付けて格納される。図６では、サービスモデルおよびハイブリッドモデルの図示を省略している。サービスモデルは、サービス管理装置２００に対応するサービスシステム１００に応じて予め定

められ、サービス＿モデルテーブルに格納されている。一方、ハイブリッドモデルは、サービス＿モデル、各ノードに割り当てられているリソース量、測定された単位リソース消費量や平均処理時間に基づいて生成され、サービス＿モデルテーブルに格納される。

【００３４】

サービスモデルは、サービスシステム内の各ノードの入力および出力を処理順に表現したデータ構造である。図７は、サービスモデルの例を模式的に示す説明図である。図７において、“Ｗｅｂ”，“Ａｐｐ”，“ＤＢ”という文字とともに示した矩形は、ノード（サーバ）を表している。そして、矩形（ノード）の左側に表した円は、そのノードへの入力を表し、矩形（ノード）の右側に表した円は、そのノードからの出力を表す。また、円内に示した黒い点は、リクエストを表している。図７に示すように、サービスモデルがノードの出力の分岐や集約を表現してもよい。ノードの出力の分岐や集約を表現することで、同時並列に行われる処理を表現することができる。

【００３５】

ハイブリッドモデルは、サービスモデルにリソースモデルを組み合わせたデータ構造である。まず、リソースモデルについて説明する。リソースモデルは、ノードに入力が生じたときにおける、そのノードの仮想リソースの未使用リソース量および使用中のリソース量を表現するデータ構造である。図８は、リソースモデルの例を模式的に示す説明図である。リソースモデルは、サービスモデル内のノードの入力および出力の組み合わせに対応している。サービスモデル内のノードへの入力を図８では、矩形２１で表し、そのノードからの出力を矩形２２で表している。矩形２１の左側の円２３は、そのノードに割り当てられているリソース量の内、未使用のリソース量を表している、矩形２１の右側の円２４は、そのノードに割り当てられているリソース量の内、使用中のリソース量を表している。また、リソースモデルは、そのノードにおける単位リソース消費量２０も含む。リソースモデルの初期状態では、円２３は、ノードに割り当てられているリソース量全体を表す。ノードのリソース消費のシミュレーション時に、リクエストの発生にあわせて、円２３が示す未使用のリソース量、および円２４が表す使用中のリソース量が更新される。また、シミュレーションにおいて、リクエスト発生後、そのノードの平均処理時間が経過するタイミングで、そのリクエストによって使用されていたリソース量が未使用状態に戻るように、円２３が示す未使用のリソース量、および円２４が表す使用中のリソース量が更新される。

【００３６】

図９は、ハイブリッドモデルの例を示す説明図である。本例では、図７に示すサービスモデル内のノードの入力および出力の組み合わせに対応する各リソースモデルが生成され、サービスモデルと組み合わされている。図９に示すリソースモデルの未使用のリソース量、使用中のリソース量は、シミュレーションの一時点における値を表している。ハイブリッドモデルを用いてノードのリソース消費のシミュレーションを実行することで、Ｗｅｂサーバ、Ａｐｐサーバ、ＤＢサーバ等のノード毎に、未使用リソース量や使用中のリソース量を再現することができ、仮想リソースの余剰量や不足量を求めることができる。

【００３７】

上記のようなサービスモデルおよびハイブリッドモデルが、サービス＿モデルテーブル（図６参照）に格納される。

【００３８】

例えば、サービスモデルは、時間ペトリネット（TimePetriNet）で記述され、リソースモデルは、連続値ペトリネット（Continuous PetriNet）で記述される。ハイブリッドモデルは、それらを統合したハイブリッドペトリネット（HybridPetriNet）で記述される。

【００３９】

モニタリング部２０２は、サービス管理装置２００に対応するサービスシステム１００内の各ノードに割り当てられているリソース量、各ノードの単位リソース消費量を測定し、ノード＿リソーステーブル（図３参照）に格納する。また、モニタリング部２０２は、各ノードの平均処理時間を測定し、ノード＿リソーステーブルに格納する。さらに、モニ

タリング部２０２は、この処理を仮想リソースの種別毎に行う。また、モニタリング部２０２は、サービスシステム１００が個々のリクエスト毎に生成したログを取得し、サービス＿ログテーブルに格納する。

【００４０】

ハイブリッドモデル生成部２０３は、ノード＿リソーステーブル（図３参照）のリソース割当量、リソース消費量、平均処理時間、および予めサービス＿モデルテーブルに格納されているサービスモデル（図７参照）に基づいて、ノードの入力および出力の組み合わせに対応するリソースモデルをそれぞれ生成する。そして、ハイブリッドモデル生成部２０３は、各リソースモデルをサービスモデルと組み合わせてハイブリッドモデルを生成し、サービス＿モデルテーブルに格納する。

【００４１】

ノード＿リソース量推定部２０４は、ハイブリッドモデルシミュレータを備え、モニタリング部２０２がサービスシステム１００から取得したログと、ハイブリッドモデルとを用いて、ノードのリソース消費のシミュレーションを実行する。そして、ノード＿リソース量推定部２０４は、各ノードでのリソース消費状況を推測する。具体的には、ノード＿リソース量推定部２０４は、各ノードにおける各仮想リソースの最小リソース残量および最大リソース不足量を求める。

【００４２】

サービス＿リソース配分検証部２０５は、各ノードの最小リソース残量の和の絶対値と、各ノードの最大リソース不足量の和の絶対値との差を計算する。この差は、サービス管理装置２００に対応するサービス（サービスシステム１００が提供するサービス）に関する仮想リソースの余剰量または不足量に該当する。最小リソース残量の和の絶対値から最大リソース不足量の和の絶対値を減算した値が正であれば、余剰量に該当し、負であれば、不足量に該当する。サービス＿リソース配分検証部２０５は、上記の計算を、仮想リソースの種別毎に行う。この差分は、図５に示すリソース過不足量として、サービス＿リソース配分テーブルに格納される。

【００４３】

サービス＿リソース配分検証部２０５は、計算した差分が仮想リソースの余剰量を表していれば、その余剰量分の仮想リソースをリソースプールに返却すると判断し、余剰量分の仮想リソースを返却することを、ハブ連携部２０７を介して、ハブ装置３００に要求する。また、サービス＿リソース配分検証部２０５は、計算した差分が仮想リソースの不足量を表していれば、その不足分の仮想リソースをリソースプールから追加すると判断し、その不足分の仮想リソースの追加の許可を、ハブ連携部２０７を介して、ハブ装置３００に要求する。ハブ装置３００から要求に対する応答があったときに、サービス＿リソース配分検証部２０５は、仮想リソースの返却や追加を確定する。

【００４４】

リソース再配分部２０６は、最小リソース残量が正であるノードから、その最小リソース残量分の仮想リソースを回収すると判定する。また、リソース再配分部２０６は、仮想リソースの不足が生じているノードに対して、最大リソース不足量分の仮想リソースを追加すると判定する。仮想リソースの不足が生じているノードに対する仮想リソースの追加分は、新たに追加が許可された仮想リソースの割当量や、仮想リソースの余剰が生じているノードから回収予定のリソース量から配分されることになる。

【００４５】

ハブ連携部２０７は、余剰量分の仮想リソースを返却すること、および、不足分の仮想リソースの追加の許可を、サービス＿リソース配分検証部２０５に従ってハブ装置３００に要求する。

【００４６】

モニタリング部２０２、ハイブリッドモデル生成部２０３、ノード＿リソース量推定部２０４、サービス＿リソース配分検証部２０５、リソース再配分部２０６およびハブ連携部２０７は、例えば、サービス管理装置用プログラムに従って動作するコンピュータのＣ

ＰＵによって実現される。例えば、ＣＰＵが、サービス管理装置用プログラムを読み込み、そのプログラムに従って、モニタリング部２０２、ハイブリッドモデル生成部２０３、ノード＿リソース量推定部２０４、サービス＿リソース配分検証部２０５、リソース再配分部２０６およびハブ連携部２０７として動作してもよい。サービス管理装置用プログラムは、コンピュータが読み取り可能な記録媒体に記憶されていてもよい。また、これらの各要素がそれぞれ別々のハードウェアで実現されていてもよい。

【００４７】

ハブ装置３００は、各サービス管理装置２００，２１０からの要求を受け、一定時間毎にその各要求を参照して、サービス管理装置毎に、そのサービス管理装置に対応するサービスシステムに割り当てを許可する仮想リソースの割当量をサービス管理装置に通知する。従って、ハブ装置３００は、個々のサービス管理装置から仮想リソースの割当量の変更要求を受けた時に、すぐに要求元のサービス管理装置に応答するわけではない。すなわち、各サービス管理装置２００，２１０による要求と、ハブ装置３００から各サービス管理装置２００，２１０への割当量の通知は、非同期で行われる。

【００４８】

また、ハブ装置３００は、各サービス管理装置２００，２１０に対応する各サービスシステム１００，１１０全体で仮想リソースの余剰や不足が生じている場合には、その余剰量や不足量に応じて、各サービスシステム１００，１１０全体に対する仮想リソースの割当量の変更をリソースプール管理装置に要求する。

【００４９】

ハブ装置３００として、正常系のハブ装置と、待機系のハブ装置が設けられている。この場合、正常系のハブ装置と、待機系のハブ装置にそれぞれ、ハブ装置のＩＤ（以下、ハブＩＤと記す。）を定めておく。また、通常は、正常系のハブ装置が動作し、正常系のハブ装置に異常が生じたときに、正常系のハブ装置に代わって、待機系のハブ装置が動作する。正常系のハブ装置および待機系のハブ装置の構成は共通である。図１では、正常系のハブ装置３００のみを図示している。

【００５０】

なお、ハブ装置３００を冗長化せずに、１台のハブ装置３００のみを備える構成であってもよい。その場合、ハブ装置３００にハブＩＤを定めておかなくてもよい。

【００５１】

ハブ装置３００は、ハブ情報記憶装置３０１と、サービス連携部３０２と、ハブ＿リソース配分検証部３０３と、ハブ＿リソース再配分部３０４と、プール連携部３０５とを備える。

【００５２】

ハブ情報記憶装置３０１は、ハブ＿サービス＿リソース配分テーブルおよびハブ＿リソース配分テーブルを記憶する。

【００５３】

図１０は、ハブ＿サービス＿リソース配分テーブルの例を示す説明図である。ハブ＿サービス＿リソース配分テーブルには、サービスＩＤとリソース型の組み合わせ毎に、サービスＩＤ、ハブＩＤ、リソース量、リソース過不足量およびリソース型が対応付けて格納される。前述のように、サービス管理装置とサービスシステムとは一対一に対応し、サービスシステムはそれぞれ異なるサービス提供処理を実行する。従って、サービスＩＤは、サービス管理装置を識別するための情報として用いることができる。ハブＩＤは、ハブ装置３００自身のＩＤである。ハブ装置３００を冗長化せずに、１台のハブ装置３００のみを備える構成とする場合には、ハブＩＤの格納は省略してよい。リソース量は、サービスＩＤに対応するサービスシステム全体に割り当てられている仮想リソースの割当量である。例えば、図１０に示す１行目は、サービスＩＤ“ｓｖ１”に対応するサービス管理装置２００には、サービスシステム１００全体に割り当てるための割当量として仮想ＣＰＵのリソース量“１５”が定められていて、サービス管理装置２００からリソース過不足量が“－２”（すなわち、２仮想ＣＰＵｃｏｒｅ分不足している）という通知があったことを

表している。ハブ＿サービス＿リソース配分テーブルにおいて、“リソース量”が更新された場合、その“リソース量”に対応する“リソース過不足量”は消去される。

【００５４】

図１１は、ハブ＿リソース配分テーブルの例を示す説明図である。ハブ＿リソース配分テーブルには、仮想リソースの種別（リソース型）毎に、ハブＩＤと、全てのサービス管理装置２００，２１０に対応する全てのサービスシステム１００，１１０全体に対して現在割り当てられている仮想リソースの割当量（図１１に示すハブ＿リソース量）と、その全てのサービスシステム１００，１１０全体でのリソース過不足量（図１１に示すハブ＿リソース過不足量）と、リソース型とが格納される。“ハブ＿リソース量”が更新された場合、その“ハブ＿リソース量”に対応する“ハブ＿リソース過不足量”は消去される。

【００５５】

サービス連携部３０２は、ハブ連携部２０７から、余剰量分の仮想リソースの返却の要求や、不足分の仮想リソースの追加の要求を受け付け、その余剰量や不足量を、“リソース過不足量”として、ハブ＿サービス＿リソース配分テーブル（図１０参照）に格納する。また、サービス管理装置に対応するサービスシステムへの割当を許可する仮想リソースの割当量が変更された場合、サービス連携部３０２は、変更後の割当量をそのサービス管理装置に通知する。

【００５６】

ハブ＿リソース配分検証部３０３は、一定時間毎に、サービスシステム毎のリソース過不足量をハブ＿サービス＿リソース配分テーブル（図１０参照）から読み出す。そして、ハブ＿リソース配分検証部３０３は、リソース種別毎に、リソース過不足量の和を算出すし、その算出結果を、ハブ＿リソース過不足量として、ハブ＿リソース配分テーブル（図１１参照）に格納する。ハブ＿リソース過不足量（リソース種別毎のリソース過不足量の和）が正であれば、仮想リソースの余剰量を表している。ハブ＿リソース過不足量が負であれば、仮想リソースの不足量を表している。

【００５７】

ハブ＿リソース配分検証部３０３は、ハブ＿リソース過不足量が仮想リソースの余剰量を表していれば、その余剰量分の仮想リソースを、サービスシステム全体からリソースプールに返却すると判断し、その余剰量分の仮想リソースを返却することを、プール連携部３０５を介して、リソースプール管理装置４００，４１０に要求する。また、ハブ＿リソース配分検証部３０３は、ハブ＿リソース過不足量が仮想リソースの不足量を表していれば、サービスシステム全体に割り当てる仮想リソースの割当量にその不足分を追加すると判断し、その不足分の仮想リソースの追加の許可を、プール連携部３０５を介して、リソースプール管理装置４００，４１０に要求する。リソースプール管理装置４００，４１０から要求に対する応答があったときに、ハブ＿リソース配分検証部３０３は、仮想リソースの返却や追加を確定する。

【００５８】

ハブ＿リソース再配分部３０４は、サービスシステム毎に、仮想リソースの余剰量分の仮想リソースを回収すると判定したり、不足分の仮想リソースを追加すると判定したりする。仮想リソースの不足が生じているサービスシステムに対する仮想リソースの追加分は、新たに追加が許可された仮想リソースの割当量や、仮想リソースの余剰が生じているサービスシステムから回収予定のリソース量から配分される。

【００５９】

プール連携部３０５は、余剰量分の仮想リソースを返却すること、および、不足分の仮想リソースの追加の許可を、ハブ＿リソース配分検証部３０３に従ってリソースプール管理装置４００，４１０に要求する。

【００６０】

サービス連携部３０２、ハブ＿リソース配分検証部３０３、ハブ＿リソース再配分部３０４およびプール連携部３０５は、例えば、ハブ装置用プログラムに従って動作するコンピュータのＣＰＵによって実現される。例えば、ＣＰＵが、ハブ装置用プログラムを読み

込み、そのプログラムに従って、サービス連携部３０２、ハブ＿リソース配分検証部３０３、ハブ＿リソース再配分部３０４およびプール連携部３０５として動作してもよい。ハブ装置用プログラムは、コンピュータが読み取り可能な記録媒体に記憶されていてもよい。また、これらの各要素がそれぞれ別々のハードウェアで実現されていてもよい。

【００６１】

リソースプール管理装置４００，４１０は、ハブ装置３００からの要求に応じて、各サービスシステム全体に対して割当を許可する仮想リソースの割当量を、ハブ装置３００に通知する。

【００６２】

リソースプール管理装置は、仮想リソースの種別毎に設けられる。例えば、割当量を制御する仮想リソースが仮想ＣＰＵと仮想ＲＡＭであるとする。この場合、仮想ＣＰＵに対応するリソースプール管理装置と、仮想ＲＡＭに対応するリソースプール管理装置がそれぞれ設けられる。本実施形態では、リソースプール管理装置４００が、仮想ＣＰＵに対応していて、リソースプール管理装置４１０が、仮想ＲＡＭに対応している場合を例にして説明する。以下では、リソースプール管理装置が２台である場合を例にして説明するが、リソースプール管理装置は２台とは限らず、仮想リソースの種別の数によって定まる。

【００６３】

以下、リソースプール管理装置４００を例にして説明するが、各リソースプール管理装置４００，４１０の構成は同様である。

【００６４】

リソースプール管理装置４００は、プール情報記憶装置４０１と、ハブ連携部４０２と、割当変更部４０３とを備える。

【００６５】

プール情報記憶装置４０１は、プール＿ハブ＿リソース配分テーブルおよびプール＿リソース容量テーブルを記憶する。

【００６６】

図１２は、プール＿ハブ＿リソース配分テーブルの例を示す説明図である。プール＿ハブ＿リソース配分テーブルには、通信相手となるハブ装置３００のハブＩＤ、リソースプール管理装置４００自身のＩＤ（プールＩＤ）、ハブ＿リソース量、ハブ＿リソース過不足量、リソース型が対応付けて格納される。リソースプール管理装置４００は、仮想ＣＰＵに対応するので、リソース型として“ＣＰＵ”が格納されている（図１２参照）。前述のように、ハブ＿リソース量は、全てのサービスシステム１００，１１０全体に対して現在割り当てられている仮想リソースの割当量である。本例では、この仮想リソースは、仮想ＣＰＵである。また、ハブ＿リソース過不足量は、全てのサービスシステム１００，１１０全体でのリソース過不足量である。

【００６７】

図１３は、プール＿リソース容量テーブルの例を示す説明図である。プール＿リソース容量テーブルには、プールＩＤ、プール＿割当容量、プール＿リソース容量、リソース型、リソース単位が対応付けて格納される。リソースプール管理装置４００は、仮想ＣＰＵに対応するので、リソース型として“ＣＰＵ”が格納されている（図１３参照）。プール＿割当容量は、全てのサービスシステム１００，１１０全体に対して現在割り当てられている仮想リソースの割当量である。プール＿リソース容量は、全てのサービスシステム１００，１１０全体に対して割当可能な仮想リソースの割当量の最大値である。リソース単位は、仮想リソースの返却や追加をする際の単位を表す。図１３に示す例では、仮想ＣＰＵｃｏｒｅ単位で仮想ＣＰＵの返却や追加を行う場合を示している。

【００６８】

図１２および図１３では、仮想ＣＰＵに対応するリソースプール管理装置４００に記憶されるプール＿ハブ＿リソース配分テーブルおよびプール＿リソース容量テーブルの例を示した。図１４は、仮想ＲＡＭに対応するリソースプール管理装置４１０のプール情報記憶装置４０１に記憶されるプール＿ハブ＿リソース配分テーブルの例を示す。また、図１

５は、仮想ＲＡＭに対応するリソースプール管理装置４１０のプール情報記憶装置４０１に記憶されるプール＿リソース容量テーブルの例を示す。図１５に示す例では、１ＧＢ単位で、仮想ＲＡＭの返却や追加を行う場合を示している。

【００６９】

ハブ連携部４０２は、プール連携部３０５から、余剰量分の仮想リソースの返却の要求や、不足分の仮想リソースの追加の要求を受け付け、その余剰量や不足量を、“ハブ＿リソース過不足量”として、プール＿ハブ＿リソース配分テーブル（図１２参照）に格納する。また、各サービスシステム全体に対して割当を許可する仮想リソースの割当量が変更された場合、ハブ連携部４０２は、変更後の割当量をサービス管理装置に通知する。

【００７０】

割当変更部４０３は、一定時間毎に、ハブ＿リソース過不足量（図１２参照）を確認し、そのハブ＿リソース過不足量に応じて、各サービスシステム全体に対して割当を許可する仮想リソースの割当量を変更する。従って、ハブ装置３００による要求と、リソースプール管理装置からハブ装置３００への割当量の通知は、非同期で行われる。

【００７１】

ハブ連携部４０２および割当変更部４０３は、例えば、リソースプール管理装置用プログラムに従って動作するコンピュータのＣＰＵによって実現される。例えば、ＣＰＵが、リソースプール管理装置用プログラムを読み込み、そのプログラムに従って、ハブ連携部４０２および割当変更部４０３として動作してもよい。リソースプール管理装置用プログラムは、コンピュータが読み取り可能な記録媒体に記憶されていてもよい。

【００７２】

次に、本発明の処理経過について説明する。

図１６は、サービス管理装置の処理経過の例を示すフローチャートである。ここでは、サービス管理装置２００を例にして説明するが、他のサービス管理装置２１０に関しても同様である。

【００７３】

まず、モニタリング部２０２は、サービス管理装置２００に対応するサービスシステム１００内の各ノードに割り当てられているリソース量、各ノードの単位リソース消費量を測定する。モニタリング部２０２は、その測定結果をそれぞれ、ノード＿リソーステーブル（図３参照）に、リソース割当量、リソース消費量として格納する。モニタリング部２０２は、この処理を、仮想ＣＰＵ、仮想ＲＡＭ等の仮想リソースの種別毎に行う。また、モニタリング部２０２は、サービスシステム１００内の各ノードの平均処理時間も測定して、ノード＿リソーステーブルに格納する。さらに、モニタリング部２０２は、サービスシステム１００からログを取得し、サービスログテーブル（図４参照）に格納する（ステップＡ１）。

【００７４】

次に、ハイブリッドモデル生成部２０３は、ノード＿リソーステーブルのリソース割当量、リソース消費量、平均処理時間（すなわち、ステップＡ１で測定された各ノードのリソース割当量、単位リソース消費量、平均処理時間）と、予めサービス＿モデルテーブルに格納されているサービスモデル（図７参照）とに基づいて、ハイブリッドモデルを生成し、サービス＿モデルテーブルに格納する（ステップＡ２）。

【００７５】

ハイブリッドモデル生成部２０３は、ノード毎に、着目しているノードの入力および出力の組み合わせに関して、リソースモデル（図８参照）を生成する。ハイブリッドモデル生成部２０３は、着目しているノードにおける各仮想リソースの単位リソース消費量を、リソースモデル内において単位リソース消費量２０（図８参照）として定める。また、着目しているノードにおける各仮想リソースの割当量を、未使用リソース量の初期値として定め、各仮想リソースの使用中のリソース量の初期値を０と定める。また、着目しているノードの平均処理時間を、リクエスト発生後の処理時間として定める。

【００７６】

このようにハイブリッドモデル生成部２０３は、ノード毎に、リソースモデルを生成し、各リソースモデルと、サービスモデルにおけるノードの入力および出力とを対応付けることによって、ハイブリッドモデル（図９参照）を生成する。

【００７７】

ハイブリッドモデルを用いることで、サービスシステムの振る舞いと仮想リソースの消費状況を一元的に再現することができる。その結果、いつ、どのノードで、仮想リソースがどれだけ足りなかったか、あるいは、どれだけ余っていたかを再現することができる。この仮想リソースの不足量や余剰量が、仮想リソースの再配分の手掛かりとして利用される。

【００７８】

ノード＿リソース量推定部２０４は、サービス＿ログテーブル（図４参照）が示す各リクエストが、タイムスタンプによって示される時刻に生じるものとして、各ノードでの未使用のリソース量および使用中のリソース量の変化をシミュレートすることによって、リソース消費状況を推定する（ステップＡ３）。そして、ステップＡ３で、ノード＿リソース量推定部２０４は、最小リソース残量および最大リソース不足量を、それぞれのノードで、それぞれの仮想リソースの種別毎に求める。

【００７９】

ステップＡ３において、ノード＿リソース量推定部２０４は、１つのリクエストが生じると、最初のノードにおける各仮想リソースの未使用リソース量を、単位リソース消費量分だけ減少させる。このとき、ノード＿リソース量推定部２０４は、そのノードにおける各仮想リソースの使用中のリソース量を、単位リソース消費量分だけ増加させる。ノード＿リソース量推定部２０４は、リクエストが生じる毎に、同様の処理を行う。また、ノード＿リソース量推定部２０４は、１つのリクエストが生じた時刻から、そのノードの平均処理時間が経過した時刻において、各仮想リソースの使用中のリソース量を、単位リソース消費量分だけ減少させ、各仮想リソースの未使用リソース量を、単位リソース消費量分だけ増加させる。すなわち、使用中と判定していた仮想リソースを未使用の状態に戻す。

【００８０】

このように、ノード＿リソース量推定部２０４は、ハイブリッドモデル内のリソースモデルにおける未使用リソース量および使用中のリソース量を変化させる。これらのリソース量の値は、図８に示す円２３，２４において適宜更新される（図９参照）。

【００８１】

リクエストが生じた時刻から平均処理時間が経過する時刻までの間に、さらにリクエストが発生している状態では、各仮想リソースの未使用リソース量は、さらに減少していく。このとき、新たに１つのリクエストが発生した時刻において、未使用リソース量が、単位リソース消費量よりも少ない場合、ノード＿リソース量推定部２０４は、そのシミュレーションにおいて、そのリクエストの処理を待機させる。すなわち、ノード＿リソース量推定部２０４は、未使用リソース量および使用中のリソース量を更新せず、以前に生じたリクエストに対する処理時間が経過して、未使用リソース量が増加する時刻まで、新たに発生したリクエストの処理を待機させる。リクエストの待ち数（待機中のリクエストの数）は、１つとは限らず、２つ以上になる場合もある。未使用リソース量が増加し、未使用リソース量が、単位リソース消費量よりも多くなった時刻に、待機中のリクエストに応じた処理を開始するものとして、ノード＿リソース量推定部２０４は、再度、未使用リソース量および使用中のリソース量を更新する。

【００８２】

なお、仮想リソースの種別が複数存在する場合、少なくとも１つの種別の仮想リソースの未使用リソース量が単位リソース消費量よりも少ない場合、新たに発生したリクエストは待ち状態とする。そして、全ての種別の仮想リソースの未使用リソース量がそれぞれ単位リソース消費量以上であることを条件に、リクエストに応じた処理が開始されるものとして、ノード＿リソース量推定部２０４は、シミュレーションを実行する。

【００８３】

１つのノードでの処理が終了したならば、次のノードが処理を開始するものとして、ノード＿リソース量推定部２０４は、他のノードにおいても、同様に、未使用リソース量および使用中のリソース量を更新する。

【００８４】

ノード＿リソース量推定部２０４は、上記のシミュレーションにおいて、各ノードの各仮想リソースの変化を監視し、各ノードの各仮想リソースについて、それぞれ、最小リソース残量および最大リソース不足量を特定し、ノード＿リソーステーブル（図３参照）に格納する。ノード＿リソース量推定部２０４は、着目しているノードにおけるリクエストの待ち数の最大値と単位リソース消費量との積を、そのノードでの最大リソース不足量として算出すればよい。

【００８５】

ただし、ノード＿リソーステーブルに格納する際、ノード＿リソース量推定部２０４は、最小リソース残量の小数点以下を切り捨て、最大リソース不足量の小数点以下を切り上げる。また、本実施形態では、仮想リソースの返却や追加をする際の単位量（以下、単位量と記す。）が“１”で表される場合を例にする。例えば、仮想ＣＰＵの単位量は“１仮想ＣＰＵｃｏｒｅ”であり、仮想ＲＡＭの単位量が“１ＧＢ”である場合を例にする。よって、上記のように切り捨てや切り上げによって、最小リソース残量および最大リソース不足量を整数に揃えることで、最小リソース残量および最大リソース不足量は単位量の整数倍になる。

【００８６】

サービス＿リソース配分検証部２０５は、仮想リソースの種別毎に、各ノードの最小リソース残量の和の絶対値から、各ノードの最大リソース不足量の和の絶対値を減算した結果を、リソース過不足量として計算し、サービス＿リソース配分テーブル（図５参照）に格納する（ステップＡ４）。

【００８７】

なお、ステップＡ４以降の処理は、リソース型毎（仮想リソースの種別毎）に行われる。例えば、サービス管理装置２００は、仮想ＣＰＵに関してステップＡ４以降の処理を行うとともに、仮想ＲＡＭに関してもステップＡ４以降の処理を行う。仮想リソースの種別が他にもあれば、その種別に関しても、ステップＡ４以降の処理を行う。

【００８８】

ステップＡ４の後、サービス＿リソース配分検証部２０５は、着目しているリソース型について算出されたリソース過不足量と単位量とを比較する（ステップＡ５）。以下、単位量分の余剰リソース量を“１”と表す。また、単位量分の不足リソース量を“－１”と表す。

【００８９】

リソース過不足量が１以上である場合、サービス＿リソース配分検証部２０５は、そのリソース過不足量（この場合、余剰リソース量）をリソースプールに返却すると判断し、余剰量分の仮想リソースを返却することを、ハブ連携部２０７を介して、ハブ装置３００に要求する。このとき、サービス＿リソース配分検証部２０５は、ハブ連携部２０７を介して、リソース過不足量をハブ装置３００に送信する。そして、サービス＿リソース配分検証部２０５は、余剰量分のリソース量を削減した新たなリソース割当量（サービスシステム１００のノード全体に対するリソース割当量）をハブ装置３００から通知されたときに、仮想リソースの返却を確定する（ステップＡ６）。具体的には、サービス＿リソース配分検証部２０５は、ハブ装置３００から通知された新たなリソース割当量を、サービス＿リソース配分テーブルに格納し、サービス＿リソース配分テーブルからリソース過不足量を消去する。ステップＡ６の後、ステップＡ８に移行する。

【００９０】

リソース過不足量が－１以下である場合、サービス＿リソース配分検証部２０５は、そのリソース過不足量（この場合、リソースの不足量）分の仮想リソースを追加すると判断し、その不足量の追加の許可をハブ連携部２０７を介して、ハブ装置３００に要求する。

このとき、サービス＿リソース配分検証部２０５は、ハブ連携部２０７を介して、リソース過不足量をハブ装置３００に送信する。そして、サービス＿リソース配分検証部２０５は、不足量分のリソース量を増加させた新たなリソース割当量（サービスシステム１００のノード全体に対するリソース割当量）をハブ装置３００から通知されたときに、仮想リソースの追加を確定する（ステップＡ７）。具体的には、サービス＿リソース配分検証部２０５は、ハブ装置３００から通知された新たなリソース割当量を、サービス＿リソース配分テーブルに格納し、サービス＿リソース配分テーブルからリソース過不足量を消去する。ステップＡ７の後、ステップＡ８に移行する。

【００９１】

ただし、各サービス管理装置２００，２１０からハブ装置３００への要求と、ハブ装置３００からの応答は、非同期に行われる。従って、ステップＡ６，Ａ７において、サービス管理装置がハブ装置３００に要求を行ってから、ハブ装置３００から応答があるまでの期間は、一律ではない。

【００９２】

また、リソース過不足量が－１より大きく、１未満である場合、ステップＡ５の後、ステップＡ８に移行する。

【００９３】

ステップＡ８において、リソース再配分部２０６は、最小リソース残量が正となっているノードから、その最小リソース残量分の仮想リソースを回収することを決定する。具体的には、リソース再配分部２０６は、ノード＿リソーステーブル（図３参照）において、最小リソース残量が正となっているノードのリソース割当量から、その最小リソース残量を減算することによって、リソース割当量を更新する。また、リソース再配分部２０６は、最大リソース不足量が負となっているノードに対して、その最大リソース不足量分の仮想リソースを追加することを決定する。具体的には、リソース再配分部２０６は、ノード＿リソーステーブル（図３参照）において、最大リソース不足量が負となっているノードのリソース割当量に、その最大リソース不足量に相当するリソース量を追加することによって、リソース割当量を更新する。また、リソース再配分部２０６は、リソース割当量を更新したノードの最小リソース残量および最大リソース不足量を消去する。

【００９４】

ステップＡ８の後、サービスシステム１００のオペレータが、ノード＿リソーステーブルにおける更新後のリソース割当量を参照し、そのリソース割当量に合わせて、サービスシステム１００内の各ノードの仮想リソースの配分を物理的に更新する。また、ノード＿リソーステーブルにおける更新後のリソース割当量に合わせて、自動的に、サービスシステム１００内の各ノードの仮想リソースの割当を物理的に更新する手段が、各サービス管理装置２００，２１０に設けられていてもよい。

【００９５】

図１７は、ハブ装置３００の処理経過の例を示すフローチャートである。ハブ装置３００のサービス連携部３０２は、各サービス管理装置２００，２１０から余剰量分の仮想リソースの返却の要求（図１６に示すステップＡ６）や、不足量分の仮想リソースの追加許可の要求（図１６に示すステップＡ７）の要求を受けるときに、リソース過不足量の情報も受信する。このとき、サービス連携部３０２は、要求元のサービス管理装置に対応するサービスＩＤおよび、返却や追加の要求のあったリソース型に対応するリソース過不足量として、受信したリソース過不足量をハブ＿サービス＿リソース配分テーブル（図１０参照）に格納する。

【００９６】

ハブ装置３００は、以下に示すステップＢ１～Ｂ５の処理を、リソース型毎に行う。

【００９７】

ハブ＿リソース配分検証部３０３は、一定時間毎にハブ＿サービス＿リソース配分テーブルを読み出し、読み出したリソース過不足量の和を計算する。そして、ハブ＿リソース配分検証部３０３は、その計算結果を、着目しているリソース型におけるハブ＿リソース

過不足量として、ハブ＿リソース配分テーブル（図１１参照）に格納する（ステップＢ１）。

【００９８】

ステップＢ１の後、ハブ＿リソース配分検証部３０３は、ステップＢ１で計算したハブ＿リソース過不足量と、着目しているリソース型の単位量とを比較する（ステップＢ２）。前述のように、単位量分の余剰リソース量を“１”と表す。また、単位量分の不足リソース量を“－１”と表す。

【００９９】

ハブ＿リソース過不足量が１以上である場合、ハブ＿リソース配分検証部３０３は、そのハブ＿リソース過不足量（この場合、余剰リソース量）をリソースプールに返却すると判断し、余剰量分の仮想リソースを返却することを、プール連携部３０５を介して、着目しているリソース型に対応するリソースプール管理装置に要求する。ここでは、着目しているリソース型が仮想ＣＰＵであり、リソースプール管理装置４００に対して要求を行う場合を例にして説明する。また、ハブ＿リソース配分検証部３０３は、この要求を行うときに、ハブ＿リソース過不足量もリソースプール管理装置４００に送信する。そして、ハブ＿リソース配分検証部３０３は、余剰量分を削減した新たなリソース割当量（各サービスシステム１００，１１０全体に対するリソース割当量）をリソースプール管理装置４００から通知されたときに、仮想リソースの返却を確定する（ステップＢ３）。具体的には、ハブ＿リソース配分検証部３０３は、リソースプール管理装置４００から通知された新たなリソース割当量をハブ＿リソース量として、ハブ＿リソース配分テーブル（図１１参照）に格納し、ハブ＿リソース配分テーブルからハブ＿リソース過不足量を消去する。ステップＢ３の後、ステップＢ５に移行する。

【０１００】

ハブ＿リソース過不足量が－１以下である場合、ハブ＿リソース配分検証部３０３は、そのハブ＿リソース過不足量（この場合、リソースの不足量）分の仮想リソースを追加すると判断し、その不足量の追加の許可を、プール連携部３０５を介して、着目しているリソース型に対応するリソースプール管理装置４００に要求する。また、ハブ＿リソース配分検証部３０３は、この要求を行うときに、ハブ＿リソース過不足量もリソースプール管理装置４００に送信する。そして、ハブ＿リソース配分検証部３０３は、不足量分のリソース量を増加させた新たなリソース割当量（各サービスシステム１００，１１０全体に対するリソース割当量）をリソースプール管理装置４００から通知されたときに、仮想リソースの追加を確定する（ステップＢ４）。具体的には、ハブ＿リソース配分検証部３０３は、リソースプール管理装置４００から通知された新たなリソース割当量をハブ＿リソース量として、ハブ＿リソース配分テーブルに格納し、ハブ＿リソース配分テーブルからハブ＿リソース過不足量を消去する。ステップＢ４の後、ステップＢ５に移行する。

【０１０１】

ただし、ハブ装置３００からリソースプール管理装置４００への要求と、リソースプール管理装置４００からの応答は、非同期に行われる。従って、ステップＢ３，Ｂ４において、ハブ装置３００がリソースプール管理装置４００に要求を行ってから、リソースプール管理装置４００から応答があるまでの期間は、一律ではない。このことは、ハブ装置３００が他のリソースプール管理装置４１０に要求を行う場合においても同様である。

【０１０２】

また、ハブ＿リソース過不足量が－１より大きく、１未満である場合、ステップＢ２の後、ステップＢ５に移行する。

【０１０３】

ステップＢ５において、ハブ＿リソース再配分部３０４は、ハブ＿サービス＿リソース配分テーブルにおいて、着目している仮想リソースのリソース過不足量が正となっているサービスシステムから、そのリソース過不足量（この場合、余剰リソース量）分の仮想リソースを回収することを決定する。具体的には、ハブ＿リソース再配分部３０４は、ハブ＿サービス＿リソース配分テーブル（図１０参照）において、着目している仮想リソース

の“リソース量”から、正の値のリソース過不足量（余剰リソース量）を減算することによって、“リソース量”を更新する。また、ハブ＿リソース再配分部３０４は、ハブ＿サービス＿リソース配分テーブルにおいて、着目している仮想リソースのリソース過不足量が負となっているサービスシステムに対して、そのリソース過不足量（この場合、不足量）分の仮想リソースを追加することを決定する。具体的には、ハブ＿リソース再配分部３０４は、ハブ＿サービス＿リソース配分テーブルにおいて、着目している仮想リソースの“リソース量”に、不足量に相当するリソース量を追加することによって、“リソース量”を更新する。また、ハブ＿リソース再配分部３０４は、ハブ＿サービス＿リソース配分テーブルにおいて、“リソース量”を更新した行におけるリソース過不足量を消去する。

【０１０４】

そして、ステップＢ５において、ハブ＿リソース再配分部３０４は、サービス連携部３０２を介して、更新後の“リソース量”を、その“リソース量”に対応するサービスＩＤによって特定されるサービス管理装置２００に送信する。

【０１０５】

図１８は、リソースプール管理装置の処理経過の例を示すフローチャートである。ここでは、リソースプール管理装置４００を例にして説明するが、他のリソースプール管理装置４１０の処理経過も同様である。

【０１０６】

リソースプール管理装置４００のハブ連携部４０２は、ハブ装置３００から余剰量分の仮想リソースの返却の要求（図１７に示すステップＢ３）や、不足量分の仮想リソースの追加許可の要求（図１７に示すステップＢ４）の要求を受けるときに、ハブ＿リソース過不足量の情報も受信する。このとき、ハブ連携部４０２は、受信したハブ＿リソース過不足量をプール＿ハブ＿リソース配分テーブル（図１２参照）に格納する（ステップＣ１）。

【０１０７】

割当変更部４０３は、一定時間毎に、プール＿ハブ＿リソース配分テーブル内のハブ＿リソース過不足量を確認し、そのハブ＿リソース過不足量が０以上であるか否かを判定する（ステップＣ２）。

【０１０８】

ハブ＿リソース過不足量が０以上であるということは、各サービスシステム１００，１１０全体において、リソースプール管理装置４００に対応する仮想リソース（本例では、仮想ＣＰＵ）に余剰が生じていることを意味する。この場合、割当変更部４０３は、そのハブ＿リソース過不足量（ここでは、余剰リソース量）分のリソース量を、リソースプールに返却することを認めることを決定する（ステップＣ３）。

【０１０９】

ハブ＿リソース過不足量が０未満であるということは、各サービスシステム１００，１１０全体において、リソースプール管理装置４００に対応する仮想リソースの不足が生じていることを意味する。この場合、割当変更部４０３は、そのハブ＿リソース過不足量（ここでは、不足量）分のリソース量を、各サービスシステム１００，１１０全体に対して新たに追加することを決定する（ステップＣ４）。

【０１１０】

ステップＣ３またはステップＣ４の後、割当変更部４０３は、ステップＣ３またはステップＣ４での決定に応じて、プール＿ハブ＿リソース配分テーブルおよびプール＿リソース容量テーブルを更新する（ステップＣ５）。

【０１１１】

具体的には、ステップＣ３からステップＣ５に移行した場合、割当変更部４０３は、プール＿ハブ＿リソース配分テーブル内のハブ＿リソース量（図１２参照）から、ハブ＿リソース過不足量（ここでは、余剰リソース量）分のリソース量を減算することによって、ハブ＿リソース量を更新する。このとき、割当変更部４０３は、プール＿ハブ＿リソース配分テーブル内のハブ＿リソース過不足量を消去する。さらに、割当変更部４０３は、更

新後のハブ＿リソース量と同じ値で、プール＿リソース容量テーブルのプール＿割当容量を更新する。

【０１１２】

また、ステップＣ４からステップＣ５に移行した場合、割当変更部４０３は、プール＿ハブ＿リソース配分テーブル内のハブ＿リソース量（図１２参照）に、ハブ＿リソース過不足量（ここでは、不足量）分のリソース量を加算することによって、ハブ＿リソース量を更新する。このとき、割当変更部４０３は、プール＿ハブ＿リソース配分テーブル内のハブ＿リソース過不足量を消去する。さらに、割当変更部４０３は、更新後のハブ＿リソース量と同じ値で、プール＿リソース容量テーブルのプール＿割当容量を更新する。

【０１１３】

プール＿ハブ＿リソース配分テーブル内のハブ＿リソース量（図１２参照）、および、プール＿リソース容量テーブルのプール＿割当容量（図１３参照）は、いずれも、各サービスシステム１００，１１０全体に対して割当を許可する仮想リソースの割当量である。

【０１１４】

そして、ステップＣ５において、割当変更部４０３は、ハブ連携部４０２を介して、プール＿ハブ＿リソース配分テーブル内の更新後のハブ＿リソース量を、ハブ装置３００に送信する。

【０１１５】

本発明によれば、ノード＿リソース量推定部２０４が、ハイブリッドモデルを用いて、サービスシステム内のノード毎に、最小リソース残量や最大リソース不足量を特定する。そして、サービス＿リソース配分検証部２０５がリソース過不足量を計算する。従って、１つのサービスシステム全体での仮想リソースの余剰量や不足量を把握することができる。

【０１１６】

そして、ハブ装置３００は、１つのサービスシステム全体での仮想リソースの余剰量や不足量を受信するとともに、余剰量分のリソースの返却要求や、不足量のリソースの追加要求をサービス管理装置２００，２１０から受け付ける。このとき、ハブ装置３００は、１つのサービス管理装置からの要求に対して、同期して応答を行わない。ハブ装置３００は、個々のサービス管理装置２００，２１０から要求を受けつつも、個々のサービス管理装置２００，２１０とは同期せずに、一定時間毎に、各サービスシステム１００，１１０全体での仮想リソースの余剰量や不足量を把握し、個々のサービス管理装置に対して、新たな仮想リソースの割当量を通知する。従って、個々のサービスシステムに対する仮想リソースの割当量を適切に定めることができる。

【０１１７】

また、各サービスシステム１００，１１０全体に対する仮想リソースの割当量を最小化しつつ、個々のサービスシステムに対する仮想リソースの割当量を決定することができる。

【０１１８】

また、本発明では、ハブ装置３００を設け、上記のように、サービス管理装置２００，２１０からハブ装置３００への要求と、その要求に対するハブ装置３００の応答は非同期で行われる。従って、新たにサービスシステムを追加する場合には、新たにサービス管理装置を追加し、ハブ装置３００は、そのサービス管理装置に対しても、他のサービス管理装置２００，２１０に対する動作と同様の動作を行えばよい。また、既存のサービスシステムが廃止された場合には、ハブ装置３００は、そのサービスシステムに対応するサービス管理装置に対する動作を停止すればよい。従って、サービスシステムの追加や削除が生じる環境において、個々のサービスシステムに対する仮想リソースの割当量を適切に定めることができる。

【０１１９】

また、サービスシステムの追加や削除が生じても、仮想リソース制御システムとしての改修量を少なく抑えることができる。

【０１２０】
また、ハブ装置３００を設けていることによって、割当量の決定対象となる仮想リソースの種別が増加したり減少したりした場合であっても、仮想リソース制御システムとしての改修量を少なく抑えることができる。例えば、新たにストレージの割当量も決定する場合には、ストレージに対応するリソースプール管理装置を新たに設け、ハブ装置３００は、そのリソースプール管理装置に対しても、他のリソースプール管理装置４００，４１０に対する動作と同様の動作を行えばよい。また、割当量の決定対象となる仮想リソースの種別を削減する場合には、ハブ装置３００は、その仮想リソースに対応するリソースプール管理装置に対する動作を停止すればよい。
【０１２１】
仮に、ハブ装置３００を設けずに、ｍ台のサービス管理装置とｎ台のリソースプール管理装置が直接連携を行って、各リソースプール管理装置が個々のサービス管理装置に対して仮想リソースの割当量を通知する場合、リソース割当量の制御環境をｍ＊ｎ種類用意しなければならない。また、この場合、サービス管理装置を増減させたり、リソースプール管理装置を増減させたりする場合のシステム全体の改修量は多くなる。
【０１２２】
これに対して、本発明では、上記のように、サービスシステムの追加や削除、あるいは、割当量の決定対象となる仮想リソースの種別の追加や削除が生じたとしても、仮想リソース制御システムとしての改修量を抑えることができる。
【０１２３】
以下、具体例を用いて、本発明の動作について説明する。図２に例示するように、ロードバランサ、ファイアウォール、Ｗｅｂサーバ、Ａｐｐサーバ、ＤＢサーバ等のノードをコンポーネントとするＷｅｂサーバシステムが、サービスシステムとして稼働しているとする。このようなサービスシステムのサービスモデルとして、例えば、図７に示すサービスモデルがサービス管理装置２００に予め保持されているとする。また、以下に示す具体例では、仮想ＣＰＵおよび仮想ＲＡＭの２種類の仮想リソースを、割当量の決定対象とするものとする。
【０１２４】
以下に示す例では、図１に示すサービスシステム１００，１１０以外のサービスシステムも存在し、そのサービスシステムに対応するサービス管理装置が設けられているものとする。
【０１２５】
まず、サービス管理装置２００を例にして、サービス管理装置の動作の具体例を示す。サービス管理装置２００のモニタリング部２０２は、サービスシステム１００内のノードを対象にして、各ノードのリソース割当量、各ノードの単位リソース消費量、および各ノードの平均処理時間を測定し、図３に示すように、ノード＿リソーステーブルに格納する。仮想ＣＰＵのリソース消費量の単位は、仮想ＣＰＵｃｏｒｅ数であり、仮想ＲＡＭのリソース消費量の単位は、ギガバイトであり、平均処理時間の単位は秒（ｓｅｃ）であるとする。図３に示す例では、ノード“sv1_n1”の仮想ＣＰＵのリソース割当量、および仮想ＲＡＭのリソース割当量は、それぞれ、“２仮想ＣＰＵｃｏｒｅ”、“１ＧＢ”である。また、そのノード“sv1_n1”の仮想ＣＰＵの単位リソース消費量、および仮想ＲＡＭの単位リソース消費量は、ぞれぞれ、“０．５仮想ＣＰＵｃｏｒｅ”、“０．２ＧＢ”である。また、そのノード“sv1_n1”の平均処理時間は２ｓｅｃである。ハイブリッドモデル生成部２０３は、図３に例示するノード＿リソーステーブルに格納されたリソース割当量、単位リソース消費量、平均処理時間と、サービスモデル（図７参照）とを用いて、図９に例示するハイブリッドモデルを生成する。
【０１２６】
ノード＿リソース量推定部２０４は、サービス＿ログテーブル（図４参照）の各リクエストをハイブリッドモデルへの入力とすることで、サービスシステムの振る舞いおよびリソース消費状況をシミュレートする。図９に例示したハイブリッドモデルにおける未使用

のリソース量、使用中のリソース量は、シミュレーションの一時点における値を表している。図９では、例えば、Ｗｅｂサーバにおいて、２つのリクエストを処理中であり、仮想ＣＰＵおよび仮想ＲＡＭをそれぞれ、“１仮想ＣＰＵｃｏｒｅ”、“０．４ＧＢ”だけ使用中である状態を表している。

【０１２７】

ノード＿リソース量推定部２０４は、シミュレーションにおいて、新たなリクエストの発生時刻に、未使用のリソース量が単位リソース消費量未満であれば、未使用のリソース量が単位リソース消費量となるまで、そのリクエストを待ち状態として、リクエストの待ち数を１増加させる。なお、シミュレーション開始時において、各ノードのリクエストの待ち数の初期値は０である。ノード＿リソース量推定部２０４は、ノード毎に、シミュレーション中のリクエストの待ち数の最大値と単位リソース消費量との積を計算し、その値を最大リソース不足量とする。

【０１２８】

また、ノード＿リソース量推定部２０４は、シミュレーション中における各ノードの未使用のリソース量の最小値も計測し、その各計測値を、各ノードの最小リソース残量とする。

【０１２９】

ノード＿リソース量推定部２０４は、シミュレーションによって得た各ノードの最大リソース不足量および最小リソース残量をノード＿リソーステーブルに格納する。図３に示すノード＿リソーステーブルに各ノードの最大リソース不足量および最小リソース残量を追加した状態の例を図１９に示す。

【０１３０】

図１９に示す例において、ノード“sv1_n1”に着目すると、最大リソース不足量が－１であり、単位リソース消費量が０．５であるので、リクエストの待ち数の最大値は、１／０．５＝２である。また、リソース割当量２を使い切るときのリクエスト数は２／０．５＝４である。従って、４つのリクエストが発生してリソース割当量２を使い切った状態で、さらにリクエストが２つ発生していたことになる。このとき、仮想ＲＡＭの不足量は、０．４ＧＢとなるが、小数点以下は切り上げられるため、仮想ＲＡＭの最大リソース不足量は、－１として格納されている。

【０１３１】

また、図１９に示す例において、ノード“sv1_n2”に着目すると、最小リソース残量が１となっている。このことは、同時に処理されるリクエストの最大数が４であり、リソース割当量４のうち、最大で０．７５＊４＝３が使用された結果、最小リソース残量が１になったことを意味する。

【０１３２】

サービス＿リソース配分検証部２０５は、図１９に例示する各ノードの最小リソース残量の和の絶対値から、各ノードの最大リソース不足量の和の絶対値を減算した値（リソース過不足量）を、リソース型毎に算出する。サービス＿リソース配分検証部２０５は、リソース型毎のリソース過不足量を、図５に例示するようにサービス＿リソース配分テーブルに格納する。

【０１３３】

図５に示す例では、サービスシステム１００の各ノード全体として、仮想ＣＰＵが２仮想ＣＰＵｃｏｒｅ分不足し、仮想ＲＡＭが４ＧＢ余っていることを表している。サービス＿リソース配分検証部２０５は、仮想ＣＰＵに関し、ハブ連携部２０７を介して、リソース過不足量“－２”をハブ装置３００に送信し、仮想ＣＰＵのリソースを２仮想ＣＰＵｃｏｒｅ追加することを要求する。また、サービス＿リソース配分検証部２０５は、仮想ＲＡＭに関し、ハブ連携部２０７を介して、リソース過不足量“４”をハブ装置３００に送信し、４ＧＢ分の仮想ＲＡＭをサービスシステム１００から返却することを要求する。

【０１３４】

ハブ装置３００から仮想ＲＡＭのリソース量を現状から４ＧＢ減少したリソース割当量

が通知されると、サービス＿リソース配分検証部２０５は、そのリソース割当量をサービス＿リソース配分テーブル（図５参照）に格納し、その行のリソース過不足量を消去する。同様に、ハブ装置３００から仮想ＣＰＵのリソース量を現状から２仮想ＣＰＵｃｏｒｅ追加したリソース割当量が通知されると、サービス＿リソース配分検証部２０５は、そのリソース割当量をサービス＿リソース配分テーブル（図５参照）に格納し、その行のリソース過不足量を消去する。

【０１３５】

本例では、図５に示す仮想ＣＰＵのリソース量は、“１５”から“１７”に更新され、図５に示す仮想ＲＡＭのリソース量は、“２１”から“１７”に更新される。そして、図５に示す各リソース過不足量は消去される。

【０１３６】

そして、リソース再配分部２０６は、図１９に例示するノード＿リソーステーブルにおいて、最大リソース不足量が負となっている行のリソース割当量に、最大リソース不足量分のリソースを追加することによって、リソース割当量を更新する。そして、リソース再配分部２０６は、その最大リソース不足量を削除する。また、リソース再配分部２０６は、最小リソース残量が正となっている行のリソース割当量から、最小リソース残量分のリソース量を減算することによって、リソース割当量を更新する。そして、リソース再配分部２０６は、その最小リソース残量を削除する。

【０１３７】

図２０は、図１９に例示するノード＿リソーステーブルを更新した状態を示す。例えば、ノード“sv1_n1”の仮想ＣＰＵの割当量、仮想ＲＡＭの割当量は、それぞれ、３仮想ＣＰＵｃｏｒｅ、２ＧＢに更新される（図２０参照）。

【０１３８】

他のサービス管理装置も同様の動作を行う。

【０１３９】

次に、ハブ装置３００の動作の具体例を示す。ハブ装置３００のサービス連携部３０２は、仮想リソースの返却や追加の要求とともにリソース過不足量を受信すると、そのリソース過不足量を、図１０に例示するようにハブ＿サービス＿リソース配分テーブルに格納する。図１０に示す例では、サービスＩＤ“sv1”に対応するサービス管理装置２００と、サービスＩＤ“sv2”に対応するサービス管理装置２１０とから、各リソース型のリソース過不足量を受信した場合を例示している。

【０１４０】

ハブ＿リソース配分検証部３０３は、一定時間毎（例えば、５分毎）にハブ＿サービス＿リソース配分テーブルを読み出し、リソース型毎にリソース過不足量の和を計算し、ハブ＿リソース過不足量としてハブ＿リソース配分テーブル（図１１参照）に格納する。図１１に示す例では、各サービスシステム１００，１１０全体として、仮想ＣＰＵが５仮想ＣＰＵｃｏｒｅ分不足し、仮想ＲＡＭが８ＧＢ分余っていることが推定される。

【０１４１】

ハブ＿リソース配分検証部３０３は、仮想ＣＰＵに対応するリソースプール管理装置４００にハブ＿リソース過不足量“－５”を送信する。そして、ハブ＿リソース配分検証部３０３は、サービスシステム１００，１１０全体に仮想ＣＰＵの割当量として５仮想ＣＰＵｃｏｒｅ追加することを要求する。また、ハブ＿リソース配分検証部３０３は、仮想ＲＡＭに対応するリソースプール管理装置４１０に対して、ハブ＿リソース過不足量“８”を送信し、８ＧＢ分の仮想ＲＡＭをサービスシステム１００，１１０全体から返却することを要求する。

【０１４２】

リソースプール管理装置４００から仮想ＣＰＵのリソース量を現状から５仮想ＣＰＵｃｏｒｅ追加したリソース割当量が通知されると、ハブ＿リソース配分検証部３０３は、そのリソース割当量を、ハブ＿リソース量としてハブ＿リソース配分テーブルに格納する。ハブ＿リソース配分検証部３０３は、その行のハブ＿リソース過不足量を消去する。

【０１４３】
同様に、リソースプール管理装置４１０から仮想ＲＡＭのリソース量を現状から８ＧＢ減少したリソース割当量が通知されると、ハブ＿リソース配分検証部３０３は、そのリソース割当量を、ハブ＿リソース量としてハブ＿リソース配分テーブルに格納する。ハブ＿リソース配分検証部３０３は、その行のハブ＿リソース過不足量を消去する。
【０１４４】
図１１に示すハブ＿リソース配分テーブルに対して上記の更新を行った後の状態を、図２１に示す。
【０１４５】
また、ハブ＿リソース再配分部３０４は、図１０に例示するハブ＿サービス＿リソース配分テーブルにおいて、リソース過不足量が負となっている行のリソース量に、リソース過不足量分のリソースを追加することによって、リソース量を更新し、そのリソース過不足量を消去する。また、ハブ＿リソース再配分部３０４は、リソース過不足量が正となっている行のリソース量から、リソース過不足量分のリソース量を減算することによって、リソース量を更新し、そのリソース過不足量を消去する。図１０に示すハブ＿サービス＿リソース配分テーブルに対して上記の更新を行った後の状態を、図２２に示す。
【０１４６】
そして、ハブ＿リソース再配分部３０４は、図２２に示すリソース量を、サービスＩＤに対応する各サービス管理装置に送信する。
【０１４７】
次に、仮想ＣＰＵに対応するリソースプール管理装置４００を例にして、リソースプール管理装置の動作の具体例を示す。リソースプール管理装置４００のハブ連携部４０２は、ハブ装置３００から、仮想リソースの返却や追加の要求とともにハブ＿リソース過不足量を受信すると、そのハブ＿リソース過不足量を、図１２に例示するようにプール＿ハブ＿リソース配分テーブルに格納する。
【０１４８】
割当変更部４０３は、一定時間毎（例えば、１０分毎）にプール＿ハブ＿リソース配分テーブルのハブ＿リソース過不足量を確認する。割当変更部４０３は、ハブ＿リソース過不足量が負であれば、ハブ＿リソース過不足量分のリソース量をハブ＿リソース量に追加することによって、ハブ＿リソース量を更新する。また、割当変更部４０３は、ハブ＿リソース過不足量が正であれば、ハブ＿リソース過不足量分のリソース量をハブ＿リソース量から減算することによって、ハブ＿リソース量を更新する。ハブ＿リソース量を更新するとき、割当変更部４０３は、ハブ＿リソース過不足量を消去する。図１２に例示するプール＿ハブ＿リソース配分テーブルに対して上記の更新を行った状態を、図２３に示す。
【０１４９】
さらに、割当変更部４０３は、図１３に例示するプール＿リソース容量テーブルのプール＿割当容量も、更新後のハブ＿リソース量と同じ値に更新する。図１３に例示するプール＿リソース容量テーブルに対して上記の更新を行った状態を、図２４に示す。
【０１５０】
そして、割当変更部４０３は、ハブ連携部４０２を介して、プール＿ハブ＿リソース配分テーブル内の更新後のハブ＿リソース量を、ハブ装置３００に送信する。
【０１５１】
ここでは、仮想ＣＰＵに対応するリソースプール管理装置４００を例にして説明した。仮想ＲＡＭに対応するリソースプール管理装置４１０の動作も同様である。リソースプール管理装置４１０がハブ装置３００から仮想リソースの返却や追加の要求とともにハブ＿リソース過不足量を受信した結果、リソースプール管理装置４１０のプール＿ハブ＿リソース配分テーブルが図１４に示す状態であるとする。また、リソースプール管理装置４１０のプール＿リソース容量テーブルが図１５に示す状態であるとする。この場合、図１４に示すプール＿ハブ＿リソース配分テーブルは、図２５に示す状態に更新される。また、図１５に示すプール＿リソース容量テーブルは、図２６に示す状態に更新される。

【０１５２】
次に、本発明の主要部について説明する。図２７は、本発明の主要部を示すブロック図である。本発明の仮想リソース制御システムは、サービス管理装置５０と、ハブ装置６０と、リソース管理装置７０とを備える。
【０１５３】
サービス管理装置５０（例えば、サービス管理装置２００，２１０）は、サービスを提供するサービスシステムと一対一に対応し、対応するサービスシステム内の個々のノードに対する仮想リソースの割当量を定める。
【０１５４】
ハブ装置６０（例えば、ハブ装置３００）は、サービス管理装置５０に対応する１つのサービスシステム全体に対する仮想リソースの割当量の不足量または余剰量を表すリソース過不足量をサービス管理装置５０から受け付け、サービス管理装置５０に対応する１つのサービスシステム全体に対する仮想リソースの割当量をサービス管理装置５０に通知する。
【０１５５】
リソース管理装置７０（例えば、リソースプール管理装置４００，４１０）は、各サービスシステム全体に対する仮想リソースの割当量を算出する。
【０１５６】
サービス管理装置５０は、モデル保持手段５１と、モニタリング手段５２と、モデル生成手段５３と、リソース過不足量算出手段５４と、仮想リソース割当量更新手段５５とを備える。
【０１５７】
モデル保持手段５１（例えば、サービスモデルテーブルを記憶するサービス情報記憶装置２０１）は、自装置に対応するサービスシステム内の各ノードの入力および出力を処理順に表現するサービスモデルを保持する。
【０１５８】
モニタリング手段５２（例えば、モニタリング部２０２）は、サービスシステム内の各ノードに関して、仮想リソースの割当量と、１つのリクエストに対して消費する仮想リソースの量である単位リソース消費量と、平均処理時間とを測定し、サービスシステムで生じたリクエストを表すログを取得する。
【０１５９】
モデル生成手段５３（例えば、ハイブリッドモデル生成部２０３）は、各ノードの仮想リソースの割当量、単位リソース消費量、および平均処理時間と、サービスモデルとに基づいて、ノードに入力が生じたときにおける当該ノードの未使用リソース量および使用中のリソース量を表現するリソースモデルを生成し、サービスモデルとリソースモデルとを組み合わせたハイブリッドモデルを生成する。
【０１６０】
リソース過不足量算出手段５４（例えば、サービス＿リソース配分検証部２０５）は、ハイブリッドモデルと、ログとを用いて、サービスシステム内の各ノードのリソース消費状況のシミュレーションを実行して、ノード毎に、仮想リソースの残量の最小値である最小リソース残量と、仮想リソースの不足量の最大値である最大リソース不足量とを算出し、ノード毎の最小リソース残量および最大リソース不足量に基づいてリソース過不足量を算出し、当該リソース過不足量をハブ装置６０に通知し、ハブ装置６０から、当該リソース過不足量が示す不足量または余剰量を解消した、当該サービス管理装置に対応する１つのサービスシステム全体に対する仮想リソースの割当量の通知を受ける。
【０１６１】
仮想リソース割当量更新手段５５（例えば、リソース再配分部２０６）は、ノード毎に、最小リソース残量または最大リソース不足量が解消されるように、仮想リソースの割当量を更新する。
【０１６２】

ハブ装置６０は、リソース過不足量保持手段６１と、全体リソース過不足量算出手段６２と、システム別仮想リソース割当量算出手段６３とを備える。

【０１６３】

リソース過不足量保持手段６１（例えば、ハブ＿サービス＿リソース配分を記憶するハブ情報記憶装置３０１）は、各サービス管理装置５０から通知されたリソース過不足量を保持する。

【０１６４】

全体リソース過不足量算出手段６２（例えば、ハブ＿リソース配分検証部３０３）は、一定期間毎に、各サービス管理装置５０から通知されたリソース過不足量に基づいて、各サービスシステム全体に対する仮想リソースの割当量の不足量または余剰量を表す全体リソース過不足量（例えば、ハブ＿リソース過不足量）を算出し、当該全体リソース過不足量をリソース管理装置７０に通知し、リソース管理装置７０から、当該全体リソース過不足量を解消した、各サービスシステム全体に対する仮想リソースの割当量の通知を受ける。

【０１６５】

システム別仮想リソース割当量算出手段６３（例えば、ハブ＿リソース再配分部３０４）は、リソース過不足量が示す不足量または余剰量を解消した、サービス管理装置に対応する１つのサービスシステム全体に対する仮想リソースの割当量を算出し、当該割当量をリソース過不足量の送信元のサービス管理装置に通知する。

【０１６６】

リソース管理装置７０は、全体リソース過不足量保持手段７１と、全体仮想リソース割当量算出手段７２とを備える。

【０１６７】

全体リソース過不足量保持手段７１（例えば、プール＿ハブ＿リソース配分テーブルを記憶するプール情報記憶装置４０１）は、ハブ装置６０から通知された全体リソース過不足量を保持する。

【０１６８】

全体仮想リソース割当量算出手段７２（例えば、割当変更部４０３）は、一定期間毎に、全体リソース過不足量を確認し、全体リソース過不足量を解消した、各サービスシステム全体に対する仮想リソースの割当量を算出し、当該割当量をハブ装置６０に通知する。

【０１６９】

そのような構成により、サービスシステムの追加や削除が生じる環境において、個々のサービスシステムに対する仮想リソースの割当量を適切に定めることができる。

【０１７０】

仮想リソースの種別毎にリソース管理装置７０を備える構成であってもよい。

【０１７１】

リソース過不足量算出手段５４が、仮想リソースの種別毎に、最小リソース残量、最大リソース不足量、およびリソース過不足量を算出し、仮想リソース割当量更新手段５５が、仮想リソースの種別毎に、仮想リソースの割当量を更新し、リソース過不足量保持手段６１が、各サービス管理装置から通知されたリソース過不足量を仮想リソースの種別毎に保持し、全体リソース過不足量算出手段６２が、仮想リソースの種別毎に、全体リソース過不足量を算出し、当該全体リソース過不足量を仮想リソースの種別に応じたリソース管理装置に通知し、システム別仮想リソース割当量算出手段６３が、サービス管理装置に対応する１つのサービスシステム全体に対する仮想リソースの割当量を、仮想リソースの種別毎に算出する構成であってもよい。

【０１７２】

モデル保持手段５１は、サービスシステム内の各ノードの入力および出力を処理順に表現するとともに、ノードの出力の分岐および集約を表現するサービスモデルを保持してもよい。

【０１７３】

以上、実施形態を参照して本願発明を説明したが、本願発明は上記の実施形態に限定されるものではない。本願発明の構成や詳細には、本願発明のスコープ内で当業者が理解し得る様々な変更をすることができる。

【０１７４】

この出願は、２０１３年８月５日に出願された日本特許出願２０１３－１６２４５６を基礎とする優先権を主張し、その開示の全てをここに取り込む。

【産業上の利用の可能性】

【０１７５】

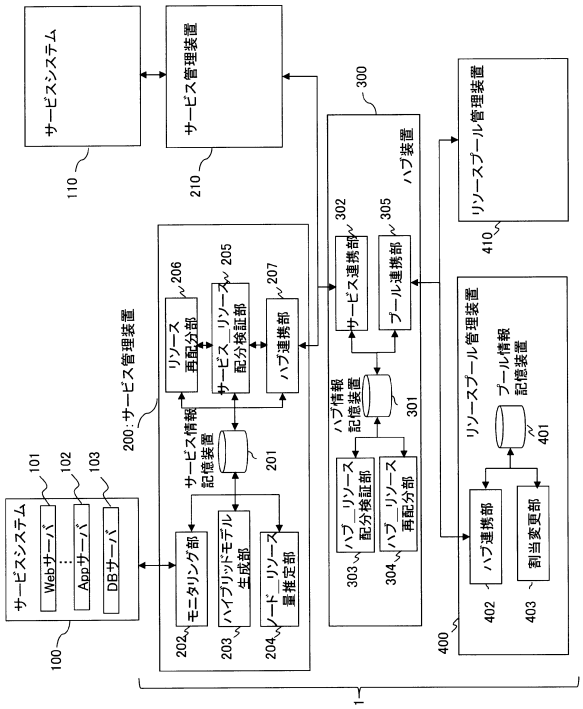
本発明は、複数のサービスシステムに対して仮想リソースの割当量の制御を行う仮想リソース制御システムに好適に適用される。

【符号の説明】

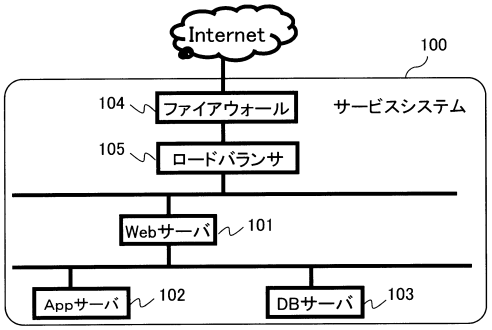
【０１７６】

２００，２１０　サービス管理装置
２０１　サービス情報記憶装置
２０２　モニタリング部
２０３　ハイブリッドモデル生成部
２０４　ノード＿リソース量推定部
２０５　サービス＿リソース配分検証部
２０６　リソース再配分部
２０７　ハブ連携部
３００　ハブ装置
３０１　ハブ情報記憶装置
３０２　サービス連携部
３０３　ハブ＿リソース配分検証部
３０４　ハブ＿リソース再配分部
３０５　プール連携部
４００　リソースプール管理装置
４０１　プール情報記憶装置
４０２　ハブ連携部
４０３　割当変更部

【図１】



【図２】



【図３】

ノード_リソーステーブル

ノードID	リソース型	リソース割当量	リソース消費量	最小リソース残量	最大リソース不足量	サービスID
sv1_n1	CPU	2	0.5			sv1
sv1_n2	CPU	4	0.75			sv1
sv1_n1	RAM	1	0.2			sv1
sv1_n2	RAM	6	0.5			sv1
…	…	…	…	…	…	…

ノードID	ノード名	平均処理時間
sv1_n1	Web1	2sec
sv1_n2	App1	3sec
…	…	…

【図４】

サービス_ログテーブル

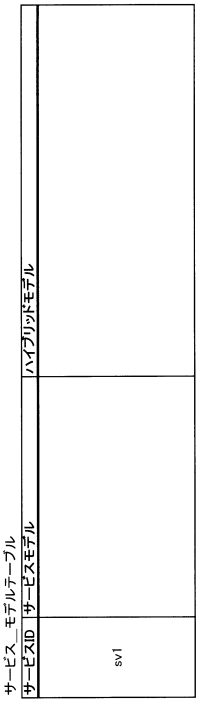
ログID	サービスID	タイムスタンプ
log1	sv1	2013-02-12_15:19:21.100
log2	sv1	2013-02-12_15:19:23.610
…	…	…

【図５】

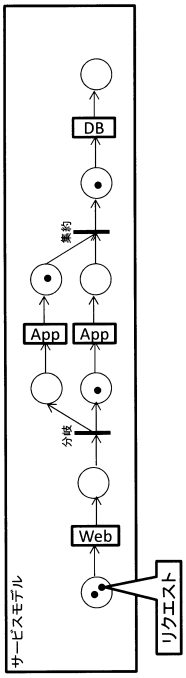
サービス_リソース配分テーブル

サービスID	リソース量	リソース過不足量	リソース型
sv1	15	-2	CPU
sv1	21	4	RAM

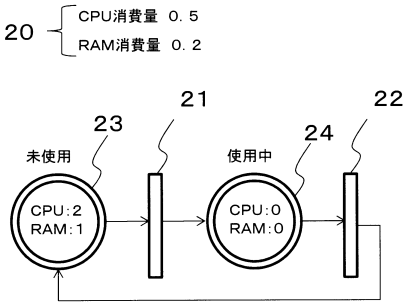
【図６】


サービス_モデルテーブル
サービスID
サービスモデル
ハイブリッドモデル
sv1

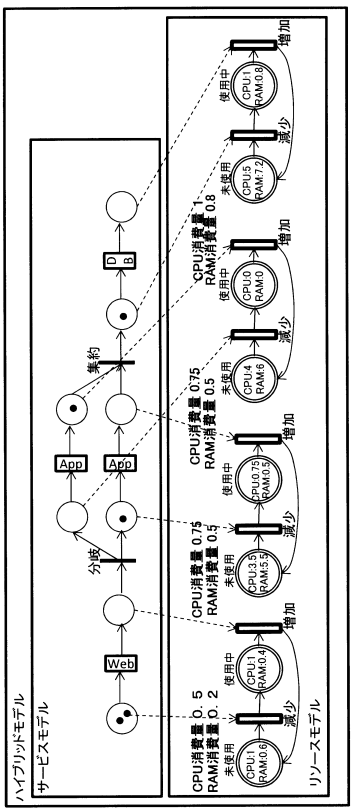
【図７】


サービスモデル
リクエスト
Web
分岐
App
App
集約
DB

【図８】


20
CPU消費量 0.5
RAM消費量 0.2
21
22
23
24
未使用
使用中
CPU:2
RAM:1
CPU:0
RAM:0

【図９】


ハイブリッドモデル
サービスモデル
リソースモデル
Web
分岐
App
App
集約
DB
CPU消費量 0.5
RAM消費量 0.2
CPU消費量 0.75
RAM消費量 0.5
CPU消費量 1
RAM消費量 0.8
未使用
使用中
増加
減少

【図１０】

ハブ＿サービス＿リソース配分テーブル

サービスID	ハブID	リソース量	リソース過不足量	リソース型
sv1	hub1	15	−2	CPU
sv1	hub1	21	4	RAM
sv2	hub1	20	1	CPU
sv2	hub1	18	−6	RAM
…	…	…	…	…

【図１１】

ハブ＿リソース配分テーブル

ハブID	ハブ＿リソース量	ハブ＿リソース過不足量	リソース型
hub1	70	−5	CPU
hub1	100	8	RAM

【図１２】

プール＿ハブ＿リソース配分テーブル

ハブID	プールID	ハブ＿リソース量	ハブ＿リソース過不足量	リソース型
hub1	pool1	70	−5	CPU

【図１３】

プール＿リソース容量テーブル

プールID	プール＿割当容量	プール＿リソース容量	リソース型	リソース単位
pool1	70	300	CPU	virtual_core

【図１４】

プール＿ハブ＿リソース配分テーブル

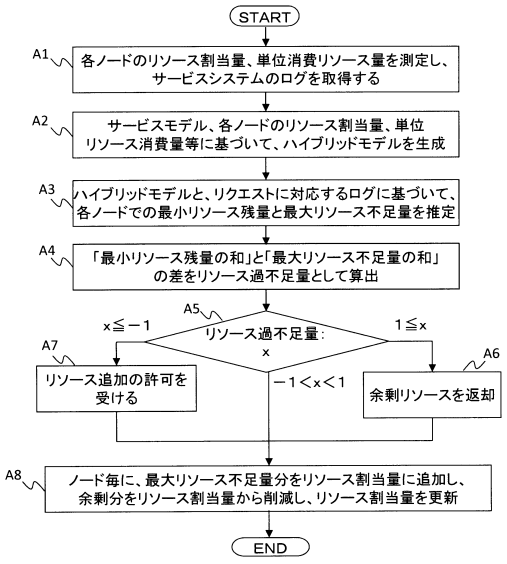
ハブID	プールID	ハブ＿リソース量	ハブ＿リソース過不足量	リソース型
hub1	pool2	100	8	RAM

【図１５】

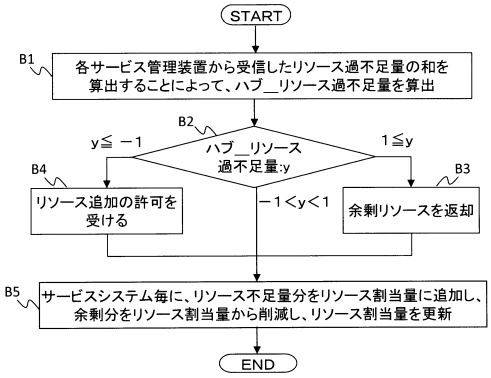
プール＿リソース容量テーブル

プールID	プール＿割当容量	プール＿リソース容量	リソース型	リソース単位
pool2	100	1000	RAM	GB

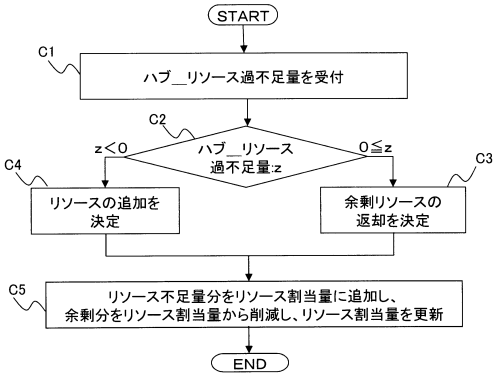
【図１６】



【図１７】



【図１８】



【図１９】

ノード_リソーステーブル

ノードID	リソース型	リソース割当量	リソース消費量	最小リソース残量	最大リソース不足量	サービスID
sv1_n1	CPU	2	0.5	0	−1	sv1
sv1_n2	CPU	4	0.75	1	0	sv1
sv1_n1	RAM	1	0.2	0	−1	sv1
sv1_n2	RAM	6	0.5	4	0	sv1
…	…	…	…	…	…	…

ノードID	ノード名	平均処理時間
sv1_n1	Web1	2sec
sv1_n2	App1	3sec
…	…	…

【図２０】

ノード_リソーステーブル

ノードID	リソース型	リソース割当量	リソース消費量	最小リソース残量	最大リソース不足量	サービスID
sv1_n1	CPU	3	0.5			sv1
sv1_n2	CPU	3	0.75			sv1
sv1_n1	RAM	2	0.2			sv1
sv1_n2	RAM	2	0.5			sv1
…	…	…	…	…	…	…

ノードID	ノード名	平均処理時間
sv1_n1	Web1	2sec
sv1_n2	App1	3sec
…	…	…

【図２１】

ハブ_リソース配分テーブル

ハブID	ハブ_リソース量	ハブ_リソース過不足量	リソース型
hub1	75		CPU
hub1	92		RAM

【図２２】

ハブ_サービス_リソース配分テーブル

サービスID	ハブID	リソース量	リソース過不足量	リソース型
sv1	hub1	17		CPU
sv1	hub1	17		RAM
sv2	hub1	19		CPU
sv2	hub1	24		RAM
…	…	…	…	…

【図２３】

プール_ハブ_リソース配分テーブル

ハブID	プールID	ハブ_リソース量	ハブ_リソース過不足量	リソース型
hub1	pool1	75		CPU

【図２４】

プール_リソース容量テーブル

プールID	プール_割当容量	プール_リソース容量	リソース型	リソース単位
pool1	75	300	CPU	virtual_core

【図２５】

プール_ハブ_リソース配分テーブル

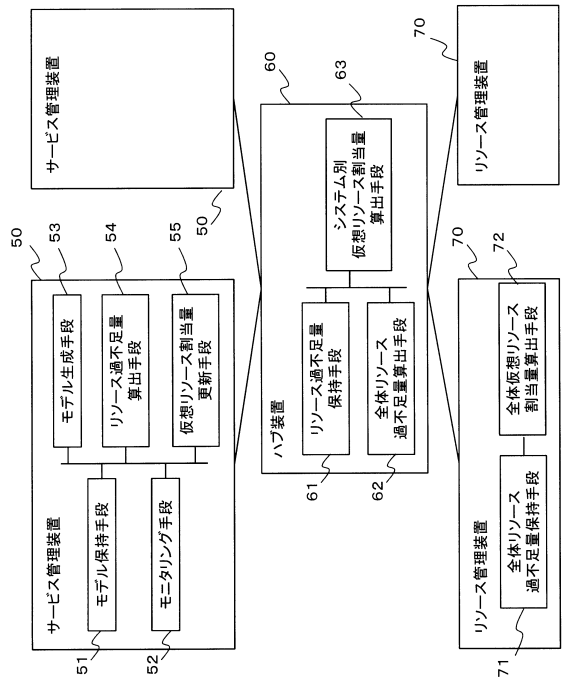
ハブID	プールID	ハブ_リソース量	ハブ_リソース過不足量	リソース型
hub1	pool2	92		RAM

【図２６】

プール_リソース容量テーブル

プールID	プール_割当容量	プール_リソース容量	リソース型	リソース単位
pool2	92	1000	RAM	GB

【図２７】



フロントページの続き

(56)参考文献 特開２０１２－１６８５８５（ＪＰ，Ａ）
特開昭６４－２１４５（ＪＰ，Ａ）
米国特許出願公開第２０１２／０２２１４５４（ＵＳ，Ａ１）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　　９／５０
Ｇ０６Ｆ　　９／４６