(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第6564013号
(P6564013)
(45)発行日　令和1年8月21日(2019.8.21)　(24)登録日　令和1年8月2日(2019.8.2)

(51)Int.Cl.　F I
G06F 9/50 (2006.01)　G06F 9/50 120Z
G06F 9/455 (2006.01)　G06F 9/455 150

請求項の数 15　(全 35 頁)

(21)出願番号　特願2017-500981(P2017-500981)
(86)(22)出願日　平成27年7月10日(2015.7.10)
(65)公表番号　特表2017-525035(P2017-525035A)
(43)公表日　平成29年8月31日(2017.8.31)
(86)国際出願番号　PCT/US2015/040048
(87)国際公開番号　WO2016/007922
(87)国際公開日　平成28年1月14日(2016.1.14)
審査請求日　平成30年2月5日(2018.2.5)
(31)優先権主張番号　62/023,076
(32)優先日　平成26年7月10日(2014.7.10)
(33)優先権主張国・地域又は機関
米国(US)
(31)優先権主張番号　62/054,901
(32)優先日　平成26年9月24日(2014.9.24)
(33)優先権主張国・地域又は機関
米国(US)

(73)特許権者　502303739
オラクル・インターナショナル・コーポレイション
アメリカ合衆国カリフォルニア州９４０６５レッドウッド・シティー，オラクル・パークウェイ５００
(74)代理人　110001195
特許業務法人深見特許事務所
(72)発明者　ティヤガラジャン，シバクマール
インド、５６００７５　カルナータカ、バンガロール、ジーバン・ビマ・ナガール、チマ・レディー・コロニー、ラア・レジデンシー、フラット・２０７

最終頁に続く

(54)【発明の名称】マルチテナントアプリケーションサーバ環境におけるリソースの分離および消費のためのシステムおよび方法

(57)【特許請求の範囲】
【請求項１】
アプリケーションサーバ環境におけるリソースの分離および消費のためのシステムであって、
複数のソフトウェアアプリケーションのデプロイおよび実行が可能であるアプリケーションサーバを含む１つ以上のコンピュータを備え、前記１つ以上のコンピュータは、
前記アプリケーションサーバ環境内で用いることができる複数のリソースと、
複数のパーティションと、を含み、
各前記パーティションは、
当該パーティション内にデプロイされる１つ以上のリソースグループと１つ以上の前記ソフトウェアアプリケーションを含み、
前記システムは、さらに、
各パーティションについて、当該パーティション内でデプロイされた前記ソフトウェアアプリケーションの実行時に当該パーティション内で消費される複数のリソースの量を監視するように構成可能なリソース消費管理モジュールを含み、
前記リソース消費管理モジュールは、前記実行時にリソースを予約するリソース予約処理、当該実行時に要求により消費されるリソースの数を決定するリソース制約処理、および後続するリソース消費のアクションを指定するリソース通知処理からなる群のうち少なくとも１つの処理を含む、システム。
【請求項２】

前記複数のリソースは複数の共有リソースを含み、前記リソース消費管理モジュールはさらに、前記リソース予約処理、前記リソース制約処理および前記リソース通知処理からなる群のうちの少なくとも１つの処理を呼出す命令を実施するように構成される、請求項１に記載のシステム。

【請求項３】

前記リソース制約処理は、前記複数のパーティションのうちのパーティションが予め定義された量を上回る前記複数の共有リソースを用いる場合、制約アクションを実行するように構成される、請求項２に記載のシステム。

【請求項４】

前記制約アクションは、減速、失敗およびシャットダウンからなる群から選択される、請求項３に記載のシステム。

【請求項５】

前記リソース通知処理は、前記複数のパーティションのうちのパーティションが予め定義された量を上回る前記複数の共有リソースを用いる場合、通知アクションを提供するように構成される、請求項２から４のいずれかに記載のシステム。

【請求項６】

前記リソース制約処理は、前記複数のパーティションのうち、公平共有値を上回る前記複数の共有リソースを用いて実行される前記ソフトウェアアプリケーションをデプロイするパーティションに対する制約アクションを実行するように構成され、前記制約アクションは減速を含み、前記減速は、前記複数のパーティションのうちの前記パーティションの前記公平共有値を減少させ、前記公平共有値は、前記複数のパーティションのうちの前記パーティションに割当てられたいくつかのワークインスタンスに関連付けられる、請求項２に記載のシステム。

【請求項７】

前記アプリケーションサーバ環境はマルチテナントアプリケーションサーバ環境を備え、前記複数のパーティションの各々は、複数の予め定義されたサービスレベルのうちの１つに関連付けることができ、前記リソース消費管理モジュールは、前記複数の予め定義されたサービスレベルの各々を用いて、当該サービスレベルに関連付けられた前記パーティションによるリソースの消費を調整するように構成される、請求項１から６のいずれかに記載のシステム。

【請求項８】

アプリケーションサーバ環境におけるリソースの分離および消費のための方法であって、

複数のソフトウェアアプリケーションのデプロイおよび実行が可能であるアプリケーションサーバを含む１つ以上のコンピュータにおいて、

前記アプリケーションサーバ環境内で用いることができる複数のリソースと、

複数のパーティションとを提供するステップと、

各前記パーティションは、

当該パーティション内にデプロイされる１つ以上のリソースグループと１つ以上の前記ソフトウェアアプリケーションを含み、

前記方法は、さらに、

各パーティションについて、当該パーティションでデプロイされた前記ソフトウェアアプリケーションの実行時に当該パーティション内で消費される前記リソースの量を監視するようにリソース消費管理モジュールを構成するステップを含み、

前記リソース消費管理モジュールは、前記実行時に、リソースを予約するリソース予約処理、当該実行時に要求により消費されるリソースの数を決定するリソース制約処理、および後続するリソース消費のアクションを指定するリソース通知処理からなる群のうち少なくとも１つの処理を含む、方法。

【請求項９】

デプロイ可能な前記複数のリソースは複数の共有リソースを含み、前記リソース消費管

理モジュールはさらに、前記リソース予約処理、前記リソース制約処理および前記リソース通知処理からなる群のうち少なくとも１つの処理を呼出す命令を実施するように構成される、請求項８に記載の方法。
【請求項１０】
前記複数のパーティションのうちのパーティションが、予め定義された量を上回る前記複数の共有リソースを用いる場合、制約アクションを実行するように前記リソース制約処理を構成するステップをさらに含む、請求項９に記載の方法。
【請求項１１】
前記制約アクションは、減速、失敗およびシャットダウンからなる群から選択される、請求項１０に記載の方法。
【請求項１２】
前記複数のパーティションのうちのパーティションが予め定義された量を上回る前記複数の共有リソースを用いる場合、通知アクションを提供するように前記リソース通知処理を構成するステップをさらに含む、請求項９または１０に記載の方法。
【請求項１３】
前記通知アクションは、ログイベントファイルをプッシュし、前記ログイベントファイルは、システムアドミニストレータによってアクセス可能である、請求項１２に記載の方法。
【請求項１４】
前記複数のパーティションの各々は、複数の予め定義されたサービスレベルのうちの１つに関連付けられ、前記方法は、さらに、
前記リソース消費管理モジュールを、前記複数の予め定義されたサービスレベルの各々を用いて、当該サービスレベルに関連付けられた前記パーティションによるリソースの消費を調整するように構成するステップを含む、請求項８から１０のいずれかまたは請求項１２に記載の方法。
【請求項１５】
機械読取り可能フォーマットのプログラム命令を含むコンピュータプログラムであって、前記プログラム命令は、コンピュータシステムによって実行されると、請求項８から１４のいずれかに記載の方法を前記コンピュータシステムに実行させる、コンピュータプログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
著作権表示
この特許文献の開示の一部は、著作権保護の対象となる題材を含んでいる。著作権の所有者は、特許商標庁の包袋または記録に掲載されるように特許文献または特許情報開示を誰でも複製できることに対して異議はないが、その他の点ではすべての如何なる著作権をも保有する。
【０００２】
発明の分野：
本発明の実施形態は、概して、アプリケーションサーバおよびクラウド環境に関し、特に、マルチテナントアプリケーションサーバ環境におけるリソースの分離および消費のためのシステムおよび方法に関する。
【背景技術】
【０００３】
背景：
ソフトウェアアプリケーションサーバは、その例として、Oracle WebLogic Server（ＷＬＳ）およびGlassfishを含んでおり、概して、エンタープライズソフトウェアアプリケーションを実行するための管理された環境を提供する。近年、クラウド環境において使用するための技術が開発されており、ユーザまたはテナントが、クラウド環境内でそれらの

アプリケーションを開発して実行することが可能になり、かつ、環境によって提供される分散型リソースを活用することが可能になっている。

【発明の概要】

【０００４】

概要：

一実施形態に従うと、アプリケーションサーバ環境におけるリソースの分離および消費のためのシステムおよび方法がこの明細書中に記載される。システムは、アプリケーションサーバ環境を含み当該アプリケーションサーバ環境が実行されている１つ以上のコンピュータにおいて、アプリケーションサーバ環境内で用いることができる複数のリソースと、１つ以上のパーティションとを提供することができる。各々のパーティションは、ドメインの管理および実行時間下位区分を備える。システムはまた、複数のリソースについての各パーティションの使用を監視するようにリソース消費管理モジュールを構成することができる。リソース消費管理モジュールは、リソース予約、リソース制約およびリソース通知からなる群のうち少なくとも１つのメンバを含み得る。

【図面の簡単な説明】

【０００５】

【図１】一実施形態に従った、アプリケーションサーバ、クラウドまたは他の環境においてマルチテナンシをサポートするためのシステムを示す図である。

【図２】一実施形態に従った、アプリケーションサーバ、クラウドまたは他の環境においてマルチテナンシをサポートするためのシステムをさらに示す図である。

【図３】一実施形態に従った、アプリケーションサーバ、クラウドまたは他の環境においてマルチテナンシをサポートするためのシステムをさらに示す図である。

【図４】一実施形態に従った、例示的なマルチテナント環境で使用されるドメイン構成を示す図である。

【図５】一実施形態に従った例示的なマルチテナント環境をさらに示す図である。

【図６】一実施形態に従った、アプリケーションサーバ環境においてリソースの分離および消費を示す図である。

【図７】一実施形態に従ったリソース消費管理インプリメンテーションを示す図である。

【図８】一実施形態に従った、リソース消費管理統合における対話を示すシーケンス図である。

【図９】一実施形態に従ったリソース消費管理インプリメンテーションを示す図である。

【図１０】一実施形態に従った、アプリケーションサーバ環境におけるリソースの分離および消費を示す図である。

【図１１】一実施形態に従った、アプリケーションサーバ環境におけるリソースの分離および消費のための方法についてのフローチャートである。

【発明を実施するための形態】

【０００６】

詳細な説明：

一実施形態に従うと、アプリケーションサーバ環境におけるリソースの分離および消費のためのシステムおよび方法がこの明細書中に記載される。システムは、アプリケーションサーバ環境を含み当該アプリケーションサーバ環境が実行されている１つ以上のコンピュータにおいて、アプリケーションサーバ環境内で用いることができる複数のリソースと、１つ以上のパーティションとを提供することができる。各々のパーティションは、ドメインの管理および実行時間下位区分を備える。システムはまた、複数のリソースについての各パーティションの使用を監視するようにリソース消費管理モジュールを構成することができる。リソース消費管理モジュールは、リソース予約、リソース制約およびリソース通知からなる群のうち少なくとも１つのメンバを含み得る。

【０００７】

アプリケーションサーバ（たとえば、マルチテナント（Multi-Tenant：ＭＴ））環境

図１は、一実施形態に従った、アプリケーションサーバ、クラウドまたは他の環境にお

いてマルチテナンシをサポートするためのシステムを示す。

【０００８】

図１に示されるように、一実施形態に従うと、アプリケーションサーバ（たとえばマルチテナント（ＭＴ））環境１００または他のコンピューティング環境は、ソフトウェアアプリケーションのデプロイメントおよび実行を可能にするものであって、アプリケーションサーバドメインを定義するために実行時に用いられるドメイン１０２の構成を含み、当該ドメイン１０２の構成に従って動作するように構成することができる。

【０００９】

一実施形態に従うと、アプリケーションサーバは、実行時に使用されるよう定義される１つ以上のパーティション１０４を含み得る。各々のパーティションは、グローバルユニークパーティション識別子（identifier：ＩＤ）およびパーティション構成に関連付けることができ、さらに、リソースグループテンプレートの参照１２６および／またはパーティション特有のアプリケーションもしくはリソース１２８とともに、１つ以上のリソースグループ１２４を含み得る。ドメインレベルのリソースグループ、アプリケーションおよび／またはリソース１４０も、任意にはリソースグループテンプレートの参照とともに、ドメインレベルで定義することができる。

【００１０】

各々のリソースグループテンプレート１６０は、１つ以上のアプリケーションＡ１６２、Ｂ１６４、リソースＡ１６６、Ｂ１６８および／または他のデプロイ可能なアプリケーションもしくはリソース１７０を定義することができ、リソースグループによって参照することができる。たとえば、図１に例示されるように、パーティション１０４におけるリソースグループ１２４は、リソースグループテンプレート１６０を参照する（１９０）ことができる。

【００１１】

概して、システムアドミニストレータは、パーティション、ドメインレベルのリソースグループおよびリソースグループテンプレート、ならびにセキュリティ領域を定義することができるとともに、パーティションアドミニストレータは、たとえば、パーティションレベルのリソースグループを作成するか、アプリケーションをパーティションにデプロイするかまたはパーティションについての特定の領域を参照することによって、それら自体のパーティションのアスペクトを定義することができる。

【００１２】

図２は、一実施形態に従った、アプリケーションサーバ、クラウドまたは他の環境においてマルチテナンシをサポートするためのシステムをさらに示す。

【００１３】

図２に示されるように、一実施形態に従うと、パーティション２０２は、たとえば、リソースグループテンプレート２１０の参照２０６を含むリソースグループ２０５と、仮想ターゲット（たとえば仮想ホスト）情報２０７と、プラグ接続可能なデータベース（pluggable database：ＰＤＢ）情報２０８とを含み得る。リソースグループテンプレート（たとえば２１０）は、たとえば、Ｊａｖａ（登録商標）メッセージサーバ（Java Message Server：ＪＭＳ）サーバ２１３、ストア・アンド・フォワード（store-and-forward：ＳＡＦ）エージェント２１５、メールセッションコンポーネント２１６またはＪａｖａデータベースコネクティビティ（Java Database Connectivity：ＪＤＢＣ）リソース２１７などのリソースとともに、複数のアプリケーションＡ２１１およびＢ２１２を定義することができる。

【００１４】

図２に例示されるリソースグループテンプレートが一例として提供される。他の実施形態に従うと、さまざまなタイプのリソースグループテンプレートおよび要素を提供することができる。

【００１５】

一実施形態に従うと、パーティション（たとえば２０２）内のリソースグループが、特

定のリソースグループテンプレート（たとえば２１０）を参照する（２２０）と、パーティション特有の情報２３０（たとえば、パーティション特有のＰＤＢ情報）を示すために、特定のパーティションに関連付けられた情報を、参照されたリソースグループテンプレートと組合わせて用いることができる。次いで、パーティション特有の情報は、パーティションによって使用されるリソース（たとえば、ＰＤＢリソース）を構成するようにアプリケーションサーバによって使用可能である。たとえば、パーティション２０２に関連付けられたパーティション特有のＰＤＢ情報は、そのパーティションによって使用されるべき適切なＰＤＢ２３８を備えたコンテナデータベース（container database：ＣＤＢ）２３６を構成する（２３２）ようにアプリケーションサーバによって使用可能である。

【００１６】

同様に、一実施形態に従うと、特定のパーティションに関連付けられた仮想ターゲット情報を用いて、そのパーティションによって使用されるべきパーティション特有の仮想ターゲット２４０（たとえば、ユニフォーム・リソース・ロケータ（uniform resource locator：ＵＲＬ）（たとえば、http://baylandurgentcare.com）によってアクセス可能にすることができるbaylandurgentcare.com）を定義する（２３９）ことができる。

【００１７】

図３は、一実施形態に従った、アプリケーションサーバ、クラウドまたは他の環境においてマルチテナンシをサポートするためのシステムをさらに示す。

【００１８】

一実施形態に従うと、config.xml構成ファイルなどのシステム構成を用いて、パーティションを定義する。当該パーティションは、そのパーティションに関連付けられたリソースグループについての構成エレメントおよび／または他のパーティションプロパティを含む。値は、プロパティ名／値の対を用いてパーティションごとに指定することができる。

【００１９】

一実施形態に従うと、複数のパーティションは、管理されたサーバ／クラスタ２４２内で、または、ＣＤＢ２４３にアクセス可能でありかつウェブ層２４４を介してアクセス可能である同様の環境内で、実行することができる。これにより、たとえば、ドメインまたはパーティションを（ＣＤＢの）ＰＤＢのうち１つ以上のＰＤＢに関連付けることが可能となる。

【００２０】

一実施形態に従うと、複数のパーティションの各々、この例においてはパーティションＡ２５０およびパーティションＢ２６０は、そのパーティションに関連付けられた複数のリソースを含むように構成することができる。たとえば、パーティションＡは、アプリケーションＡ１　２５２と、アプリケーションＡ２　２５４と、ＪＭＳ　Ａ　２５６と、さらには、ＰＤＢ　Ａ　２５９に関連付けられたデータソースＡ２５７とをともに含むリソースグループ２５１を含むように構成することができる。この場合、パーティションは仮想ターゲットＡ２５８を介してアクセス可能である。同様に、パーティションＢ２６０は、アプリケーションＢ１　２６２と、アプリケーションＢ２　２６４と、ＪＭＳ　Ｂ　２６６と、さらには、ＰＤＢ　Ｂ　２６９に関連付けられたデータソースＢ２６７とをともに含むリソースグループ２６１を含むように構成することができる。この場合、パーティションは仮想ターゲットＢ２６８を介してアクセス可能である。

【００２１】

上述の例のうちいくつかはＣＤＢおよびＰＤＢの使用を例示しているが、他の実施形態に従うと、他のタイプのマルチテナントのデータベースまたは非マルチテナントのデータベースをサポートすることができる。この場合、特定の構成は、たとえば、スキーマを使用するかまたはさまざまなデータベースを使用することによって、各々のパーティションのために提供することができる。

【００２２】

リソース

一実施形態に従うと、リソースは、環境のドメインにデプロイすることができるシステ

ムリソース、アプリケーションまたは他のリソースもしくはオブジェクトである。たとえば、一実施形態に従うと、リソースは、アプリケーション、ＪＭＳ、ＪＤＢＣ、ＪａｖａＭａｉｌ、ＷＬＤＦもしくはデータソースであり得るか、または、サーバ、クラスタもしくは他のアプリケーションサーバターゲットにデプロイすることができる他のシステムリソースもしくは他のタイプのオブジェクトであり得る。

【００２３】

パーティション

一実施形態に従うと、パーティションは、パーティション識別子（partition identifier：ＩＤ）および構成に関連付けられ得るドメインのうち実行時間および管理の下位区分またはスライスであるとともに、アプリケーションを含み得て、ならびに／または、リソースグループおよびリソースグループテンプレートを使用することによってドメイン全体に渡るリソースを参照し得る。

【００２４】

概して、パーティションは、それ自体のアプリケーションを含み、リソースグループテンプレートを介してドメイン全体に渡るアプリケーションを参照し、それ自体の構成を有し得る。パーティション可能なエンティティは、リソース、たとえば、ＪＭＳ、ＪＤＢＣ、ＪａｖａＭａｉｌ、およびＷＬＤＦリソースや、他のコンポーネント、たとえばＪＮＤＩネームスペース、ネットワークトラフィック、ワークマネージャ、セキュリティポリシーおよび領域などを含み得る。マルチテナント環境のコンテキストにおいては、システムは、テナントに関連付けられたパーティションの管理および実行時間のアスペクトへのアクセスをテナントに提供するように構成することができる。

【００２５】

一実施形態に従うと、パーティション内の各々のリソースグループは、任意には、リソースグループテンプレートを参照することができる。パーティションは、複数のリソースグループを有し得るとともに、それらの各々はリソースグループテンプレートを参照し得る。各々のパーティションは、パーティションのリソースグループが参照するリソースグループテンプレートにおいて指定されていない構成データについてのプロパティを定義することができる。これにより、パーティションが、リソースグループテンプレートで定義されたデプロイ可能なリソースをそのパーティションで使用されるべき特定の値にバインドするものとして機能することが可能となる。場合によっては、パーティションは、リソースグループテンプレートによって指定される構成情報を無効にすることができる。

【００２６】

一実施形態に従うと、パーティション構成は、たとえば、config.xml構成ファイルによって定義されるように、複数の構成エレメントを含み得る。複数の構成エレメントは、たとえば、「パーティション（partition）」（そのパーティションを定義する属性および子エレメントを含む）；「リソース・グループ（resource-group）」（パーティションにデプロイされるアプリケーションおよびリソースを含む）；「リソース・グループ・テンプレート（resource-group-template）」；（そのテンプレートによって定義されるアプリケーションおよびリソースを含む）；「ｊｄｂｃ・システム・リソース・無効化（jdbc-system-resource-override）」（データベース特有のサービス名、ユーザ名およびパスワードを含む）；ならびに、「パーティション・プロパティ（partition-properties）」（リソースグループテンプレートにおいてマクロ置換のために使用可能なプロパティキー値を含む）を含む。

【００２７】

始動後、システムは、構成ファイルによって提供される情報を用いて、リソースグループテンプレートから各々のリソースについてのパーティション特有の構成エレメントを生成することができる。

【００２８】

リソースグループ

一実施形態に従うと、リソースグループは、名前付けされ完全に修飾されたデプロイ可

能なリソースの集合であって、ドメインまたはパーティションのレベルで定義することができ、かつ、リソースグループテンプレートを参照することができる。リソースグループにおけるリソースは、完全に修飾されているものと見なされる。というのも、アドミニストレータが、それらのリソースを開始させるのに必要とされるかまたはそれらのリソースに接続するのに必要とされるすべての情報、たとえば、データソースに接続するためのクレデンシャル、またはアプリケーションについての目標情報、を提供しているからである。

【００２９】

システムアドミニストレータは、ドメインレベルで、またはパーティションレベルでリソースグループを公開することができる。ドメインレベルでは、リソースグループは、関連するリソースをグループ化するのに好都合な方法を提供する。システムは、グループ化されていないリソースと同じドメインレベルのリソースグループにおいて公開されたリソースを管理することができる。このため、リソースは、システム起動中に開始させたり、システムのシャットダウン中に停止させたりすることができる。アドミニストレータはまた、グループ内のリソースを個々に停止させるか、開始させるかまたは削除することができ、グループ上で動作させることによって暗黙的にグループ内のすべてのリソースに対して機能することができる。たとえば、あるリソースグループを停止させることにより、まだ停止されていないグループにおけるすべてのリソースを停止させ；リソースグループを始動させることにより、まだ始動させていないグループにおけるいずれのリソースも始動させ、リソースグループを削除することにより、グループに含まれるすべてのリソースを削除する。

【００３０】

パーティションレベルでは、システムまたはパーティションアドミニストレータは、任意のセキュリティ制限下で、或るパーティションにおいて０個以上のリソースグループを構成することができる。たとえば、ＳａａＳ使用事例においては、さまざまなパーティションレベルのリソースグループは、ドメインレベルのリソースグループテンプレートを参照することができる。ＰａａＳ使用事例においては、リソースグループテンプレートを参照しないが代わりにそのパーティション内でのみ利用可能にされるべきアプリケーションおよびそれらの関連するリソースを表わすパーティションレベルのリソースグループを作成することができる。

【００３１】

一実施形態に従うと、リソースグループ化を用いることで、アプリケーションと、それらアプリケーションがドメイン内で別個の管理ユニットとして使用するリソースとをともにグループ化することができる。たとえば、以下に記載される医療記録（ＭｅｄＲｅｃ）アプリケーションにおいては、リソースグループ化によりＭｅｄＲｅｃアプリケーションおよびそのリソースが定義される。複数のパーティションは、各々がパーティション特有の構成情報を用いて、同じＭｅｄＲｅｃリソースグループを実行することができ、このため、各々のＭｅｄＲｅｃインスタンスの一部であるアプリケーションが各々のパーティションにとって特有のものにされる。

【００３２】

リソースグループテンプレート

一実施形態に従うと、リソースグループテンプレートは、リソースグループから参照することができドメインレベルで定義されるデプロイ可能なリソースの集合であり、そのリソースを起動するのに必要な情報のうちいくらかは、パーティションレベル構成の仕様をサポートするように、テンプレート自体の一部として記憶されない可能性がある。ドメインは、リソースグループテンプレートをいくつ含んでもよく、それらの各々は、たとえば、１つ以上の関連するＪａｖａアプリケーションと、それらのアプリケーションが依存するリソースとを含み得る。このようなリソースについての情報のうちのいくらかは、すべてのパーティションにわたって同じであってもよく、他の情報はパーティションごとに異なっていてもよい。すべての構成がドメインレベルで指定される必要はなく、代わりに、

パーティションレベル構成が、マクロまたはプロパティ名／値の対を使用することによってリソースグループテンプレートで指定することができる。

【００３３】

一実施形態に従うと、特定のリソースグループテンプレートは、１つ以上のリソースグループによって参照可能である。概して、任意の所与のパーティション内では、リソースグループテンプレートは一度に１つのリソースグループによってのみ参照することができる。すなわち、同じパーティション内で同時に複数のリソースグループによって参照することはできない。しかしながら、異なるパーティションにおける別のリソースグループによって同時に参照することができる。リソースグループを含むオブジェクト、たとえばドメインまたはパーティションは、プロパティ名／値の割当てを用いて、任意のトークンの値をリソースグループテンプレートで設定することができる。システムは、参照するリソースグループを用いてリソースグループテンプレートを起動させると、それらのトークンを、リソースグループが含むオブジェクトにおいて設定された値と置換えることができる。場合によっては、システムはまた、静的に構成されたリソースグループテンプレートおよびパーティションを用いて、パーティション／テンプレートの組合せごとに実行時間構成を生成することができる。

【００３４】

たとえば、ＳａａＳ使用事例においては、システムは、同じアプリケーションおよびリソースを複数回を起動することができるが、そのうちの１回は、それらを用いるであろう各パーティションごとに起動され得る。アドミニストレータがリソースグループテンプレートを定義すると、これらは、どこか他のところで提供されるであろう情報を表わすためにトークンを用いることができる。たとえば、ＣＲＭ関連のデータリソースに接続する際に使用されるユーザ名は、リソースグループテンプレートにおいて＼＄｛ＣＲＭＤａｔａＵｓｅｒｎａｍｅ｝として示すことができる。

【００３５】

テナント

一実施形態に従うと、マルチテナント（ＭＴ）アプリケーションサーバ環境などのマルチテナント環境においては、テナントは、１つ以上のパーティションおよび／もしくは１つ以上のテナント認識型アプリケーションによって表現可能であるエンティティ、または１つ以上のパーティションおよび／もしくは１つ以上のテナント認識型アプリケーションに関連付けることができるエンティティである。

【００３６】

たとえば、テナントは、別個のユーザ組織、たとえばさまざまな外部会社、特定の企業内のさまざまな部門（たとえばＨＲおよび財務部）などを表わすことができ、それら各々は、異なるパーティションに関連付けることができる。テナントのグローバルユニークアイデンティティ（テナントＩＤ）は、特定の時点において特定のユーザを特定のテナントに関連付けるものである。システムは、たとえば、ユーザアイデンティティの記録を参照することによって、ユーザアイデンティティから、特定のユーザがどのテナントに属しているかを導き出すことができる。ユーザアイデンティティにより、ユーザが実行することを認可されているアクションをシステムが実施することが可能となる。ユーザアイデンティティは、ユーザがどのテナントに属し得るかを含むが、これに限定されない。

【００３７】

一実施形態に従うと、システムは、互いに異なるテナントの管理および実行時間を分離することを可能にする。たとえば、テナントは、それらのアプリケーションのいくつかの挙動、およびそれらがアクセスできるリソースを構成することができる。システムは、特定のテナントが別のテナントに属するアーティファクトを確実に管理することができないようにし、かつ、実行時に、特定のテナントの代わりに機能するアプリケーションがそのテナントに関連付けられたリソースのみを参照するが他のテナントに関連付けられたリソースは参照しないことを確実にすることができる。

【００３８】

一実施形態に従うと、テナント非認識型アプリケーションは、アプリケーションが応答している要求をどんなユーザが提示したかにかかわらず、アプリケーションが用いる如何なるリソースにもアクセス可能となるように明示的にテナントに対処する論理を含まないものである。対照的に、テナント認識型アプリケーションは、テナントに明示的に対処する論理を含む。たとえば、ユーザのアイデンティティに基づいて、アプリケーションは、ユーザが属するテナントを導き出すことができ、テナント特有のリソースにアクセスするためにその情報を用いることができる。

【００３９】

一実施形態に従うと、システムは、テナント認識型となるように明示的に書き込まれたアプリケーションをユーザがデプロイすることを可能にし、これにより、アプリケーション開発者は、現在のテナントのテナントＩＤを取得することができる。次いで、テナント認識型アプリケーションは、このテナントＩＤを用いて、アプリケーションの単一のインスタンスを用いている複数のテナントを処理することができる。

【００４０】

たとえば、単一の診療室または病院をサポートするＭｅｄＲｅｃアプリケーションは、２つの異なるパーティションまたはテナント（たとえばBayland Urgent Careテナントおよびvalley Healthテナント）に対して公開することができ、その各々は、基礎をなすアプリケーションコードを変更することなく、別個のＰＤＢなどの別個のテナント特有のリソースにアクセスすることができる。

【００４１】

例示的なドメイン構成およびマルチテナント環境

一実施形態に従うと、アプリケーションは、ドメインレベルでリソースグループテンプレートにデプロイすることができるか、または、パーティションに範囲指定されているかもしくはドメインに範囲指定されているリソースグループにデプロイすることができる。アプリケーション構成は、アプリケーション毎またはパーティション毎に指定されたデプロイメントプランを用いて無効化することができる。デプロイメントプランはまた、リソースグループの一部として指定することができる。

【００４２】

図４は、一実施形態に従った、例示的なマルチテナント環境で使用されるドメイン構成を示す。

【００４３】

一実施形態に従うと、システムがパーティションを始動させると、当該システムは、提供された構成に従って、それぞれのデータベースインスタンスに対して、各パーティションごとに１つずつ、仮想ターゲット（たとえば仮想ホスト）および接続プールを作成する。

【００４４】

典型的には、各々のリソースグループテンプレートは、１つ以上の関連するアプリケーションと、それらアプリケーションが依存するリソースとを含み得る。各々のパーティションは、それが参照するリソースグループテンプレートにおいて指定されていない構成データを提供することができるが、これは、場合によっては、リソースグループテンプレートによって指定されるいくつかの構成情報を無効にすることを含めて、パーティションに関連付けられた特定値に対するリソースグループテンプレートにおけるデプロイ可能なリソースのバインディングを行うことによって、実行可能である。これにより、システムは、各々のパーティションが定義したプロパティ値を用いて、パーティション毎にリソースグループテンプレートによってさまざまに表わされるアプリケーションを始動させることができる。

【００４５】

いくつかのインスタンスにおいては、パーティションが含み得るリソースグループは、リソースグループテンプレートを参照しないか、または、それら自体のパーティション範囲指定されたデプロイ可能なリソースを直接定義する。パーティション内で定義されるア

プリケーションおよびデータソースは、概して、そのパーティションにとってのみ利用可能である。リソースは、パーティション：<partitionName>/<resource JNDI name>、またはドメイン：<resource JNDI name>を用いて、パーティションの中からアクセスすることができるようにデプロイ可能である。

【００４６】

たとえば、ＭｅｄＲｅｃアプリケーションは、複数のＪａｖａアプリケーション、データソース、ＪＭＳサーバおよびメールセッションを含み得る。複数のテナントのためにＭｅｄＲｅｃアプリケーションを実行させるために、システムアドミニストレータは、テンプレートにおけるそれらのデプロイ可能なリソースを公開している単一のＭｅｄＲｅｃリソースグループテンプレート２８６を定義することができる。

【００４７】

ドメインレベルのデプロイ可能なリソースとは対照的に、リソースグループテンプレートにおいて公開されたデプロイ可能なリソースは、テンプレートにおいて完全には構成されない可能性があるか、または、いくつかの構成情報が不足しているので、そのままでは起動させることができない。

【００４８】

たとえば、ＭｅｄＲｅｃリソースグループテンプレートは、アプリケーションによって用いられるデータソースを公開し得るが、データベースに接続するためのＵＲＬを指定しない可能性がある。さまざまなテナントに関連付けられたパーティション、たとえば、パーティションＢＵＣ－Ａ２９０（Bayland Urgent Care：ＢＵＣ）およびパーティションＶＨ－Ａ２９２（Valley Health：ＶＨ）は、各々がＭｅｄＲｅｃリソースグループテンプレートを参照する（２９６，２９７）ＭｅｄＲｅｃリソースグループ２９３，２９４を含むことによって、１つ以上のリソースグループテンプレートを参照することができる。次いで、当該参照を用いて、Bayland Urgent Careテナントによって使用されるＢＵＣ－Ａパーティションに関連付けられた仮想ホストbaylandurgentcare.com３０４と、Valley Healthテナントによって使用されるＶＨ－Ａパーティションに関連付けられた仮想ホストvalleyhealth.com３０８とを含む各々のテナントのための仮想ターゲット／仮想ホストを作成する（３０２，３０６）ことができる。

【００４９】

図５は、一実施形態に従った例示的なマルチテナント環境をさらに示す。図５に示されるように、２つのパーティションがＭｅｄＲｅｃリソースグループテンプレートを参照している上述の例から引続いて、一実施形態に従うと、サーブレットエンジン３１０は、この例においてはBayland Urgent Careの医師テナント環境３２０およびValley Healthの医師テナント環境３３０といった複数のテナント環境をサポートするために用いることができる。

【００５０】

一実施形態に従うと、各々のパーティション３２１および３３１は、そのテナント環境についての入来トラフィックを受入れるための異なる仮想ターゲットと、異なるＵＲＬ３２２，３３２とを定義することができる。異なるＵＲＬ３２２，３３２は、パーティションと、この例ではBayland Urgent Careデータベースまたはvalley healthデータベースを含むそれぞれのリソース３２４、３３４とに接続するためのものである。同じアプリケーションコードが両方のデータベースに対して実行され得るので、データベースインスタンスは互換性のあるスキーマを用いることができる。システムがパーティションを始動させると、当該システムは、それぞれのデータベースインスタンスに対する接続プールおよび仮想ターゲットを作成することができる。

【００５１】

リソースの分離および消費

一実施形態に従うと、複数のテナントが単一のドメインを共有することを可能にすることによって顧客のコンピューティングインフラストラクチャの密度を改善させ、その利用を向上させる。しかしながら、複数のテナントまたはパーティションが同じドメイン内に

常駐する場合、リソースの割当てにおける公平性を促進し、共有リソースに対するアクセスの競合および／または干渉を防止し、かつ、複数のパーティションおよび／または関連するテナントのために一貫した性能を提供するために、ドメイン実行時間においてさまざまなパーティションがそれらのリソースにアクセスして当該リソースを使用することを判断し、管理し、分離しかつ監視することが必要になり得る。

【００５２】

一実施形態に従うと、このシステムは、システムアドミニストレータがＪＤＫ管理型リソースについてのリソース消費管理ポリシーを指定する（たとえば、制約、返還要求（recourse action）および通知を指定する）ことを可能にするために、ジャバ開発キット（たとえばＪＤＫ（Java Development Kit）８ｕ４０）によって提供されるリソース管理サポートを利用する。これらのリソースは、たとえばＣＰＵ、ヒープ、ファイルおよびネットワークを含み得る。この明細書中に記載される方法およびシステムはまた、共有リソースの消費を管理するためにアプリケーションサーバ環境内のコンテナおよびコンポーネントによって使用可能である軽量で規格ベースの包括的かつ拡張可能なリソース消費管理（resource consumption management：ＲＣＭ）フレームワークを提供することができる。

【００５３】

一実施形態に従うと、アプリケーションがアプリケーションサーバ環境内のさまざまなパーティションおよび／またはさまざまなテナントによってデプロイされる場合、アプリケーションが共有リソース（たとえば、ＣＰＵ、ネットワーク、ストレージなどの低レベルのリソース、またはデータソース、ＪＭＳ接続、ロギングなど高レベルのリソース）を使用しようとする際に生じる可能性のある概して２つの問題がある。これら２つの問題は、割当て中における競合／不公平性と、性能不定性および潜在的なサービスレベルアグリーメント（service level agreement：ＳＬＡ）違反とを含む。共有リソースに対して複数の要求があり、これにより結果として競合および干渉がもたらされる場合、結果として割当て中に競合および不公平性がもたらされる可能性がある。害のない理由（たとえば高トラフィックおよびＤＤｏＳ攻撃）、動作が不良であるかまたはバグの多いアプリケーション、悪質なコードなどのせいで異常なリソース消費要求が発生する可能性もある。このような異常な要求により、結果として、共有リソースの容量に過負荷がかかってしまい、このため、リソースに別のアプリケーションがアクセスするのが妨げられてしまう可能性がある。性能が変動することにより、潜在的なＳＬＡ違反につながるおそれがある。競合および不公平性により、結果として、さまざまなアプリケーション／コンシューマのための性能に違いが生じる可能性があり、結果として、ＳＬＡ違反につながるおそれがある。

【００５４】

この明細書中に記載されたさまざまなシステムおよび方法は、リソースの分離および消費のサポートに関連付けられる問題について論じる際にマルチパーティション／マルチテナントのアプリケーションサーバ環境を用いているが、当該システムおよび方法は、或るドメインにおける複数の共同常駐アプリケーションが（或るドメイン内における複数のパーティション内の複数のアプリケーションとは対照的に）互いと競合し、それらが共有リソースにアクセスすることによって、結果として競合が生じ、性能が予測不能なものとなってしまうような従来のデプロイメントに適用可能である。

【００５５】

一実施形態に従うと、システムアドミニストレータなどのアドミニストレータは、共有リソース（たとえば、ＣＰＵ、ヒープ、ファイルおよびネットワークなどのＪＤＫリソース）へのアクセスを監視し、コンシューマによるこれらのリソースの消費についてのポリシーを実施するためのメカニズムを備えている。たとえば、マルチパーティションシステムにおいては、システムアドミニストレータは、コンシューマが共有リソースを過剰消費せず、これにより、その共有リソースにアクセスする別のコンシューマにとってその共有リソースが不足することのないように、構成することができる。

【００５６】

一実施形態に従うと、システムアドミニストレータなどのアドミニストレータは、さま

ざまなコンシューマ／顧客に対する層状／差異化型サービスレベルを提供することができる。加えて、アドミニストレータは、任意には、（たとえば、パーティションにおいて実行されるであろう特定のワークロードまたは予測されたワークロードに対応するために、時刻に基づいて特定のパーティションのための別のポリシーを適用している）ビジネス特有のポリシーを構成することができ、かつ、共有リソースが消費される時点で上記ポリシーを実施させることができる。

【００５７】

一実施形態に従うと、アドミニストレータは、フレキシブルで拡張可能であり汎用のリソース消費フレームワークを備える。このリソース消費フレームワークは、さまざまなコンシューマによってアクセスおよび／または利用されるアプリケーションサーバ環境管理共有リソースのコンテナおよびコンポーネントを支援するために用いることができる。フレームワークは、加えて、ＲＣＭフレームワークのさまざまなユーザのためにさまざまなレベルで細分化されたポリシーを指定および実施することを可能にする。

【００５８】

一実施形態に従うと、以下の定義は、リソースの分離および消費のための方法およびシステムを記載または説明する際に用いることができる。

【００５９】

一実施形態に従うと、リソース消費管理のコンテキストにおいては、リソースという語は、アプリケーションサーバ環境などのシステム内のエンティティを意味するものであって、コンシューマ、パーティションおよび／またはテナント間で共有されるものを表わしており、量が制限されており、その不足分のせいで、結果として、リソースコンシューマのための性能が変化する可能性がある。

【００６０】

一実施形態に従うと、リソースコンシューマという語は、実行可能なエンティティであってもよく、そのリソース使用がリソース消費管理（Resource Consumption Management：ＲＣＭ）ＡＰＩによって制御されている。たとえば、マルチテナントアプリケーションサーバ環境内のリソースコンシューマはパーティションを含んでもよい。

【００６１】

一実施形態に従うと、リソースドメインは、リソースコンシューマのグループをリソースにバインドすることができ、このリソースのためにリソースコンシューマグループに対して共通のリソース管理ポリシーを課する。

【００６２】

リソース管理ポリシーは、一実施形態に従うと、予約、制約および通知を定義するものであって、リソースコンシューマによって消費されるリソースの数を制限するかまたは消費の割合を調節することができる。

【００６３】

一実施形態に従うと、制約は、リソースに対してリソース消費要求がなされたときに呼出されるコールバックであり得る。コールバックは、要求によって消費され得るリソースユニットの数を決定するそのリソースについてのポリシーを実現することができる。

【００６４】

通知は、一実施形態に従うと、リソース消費要求が処理された直後に呼出されるコールバックであり得る。通知は、後続のリソース消費（たとえばアカウンティング、ロギングなど）を実行するためのアクションを指定するリソース管理ポリシーを実現するために用いることができる。

【００６５】

一実施形態に従うと、リソース属性は、リソースのプロパティまたは属性を記述している。加えて、リソースディスペンサは、リソースを作成するコード／コンテナ／論理を意味する。さらに、ＲＭ可能化とは、ＲＣＭフレームワークがそのリソースの消費を管理することができるようにリソースディスペンサを変更する動作を意味し得る。

【００６６】

返還要求は、一実施形態に従うと、リソースコンシューマによるリソース消費が或る状態に達したときに実行されるアクションであり得る。返還要求は、通知であってもよく、または、リソースの使い過ぎを防ぐかもしくは使い過ぎに対処しようと試みるために予防措置を取ることであり得る。返還要求は、リソース消費要求がなされたスレッドとは異なるスレッドにおいて実行することができる。

【００６７】

図６は、一実施形態に従った、アプリケーションサーバ環境におけるリソースの分離および消費を示す。図６は、ドメイン６１０を含むアプリケーションサーバ環境６００を示す。システムアドミニストレータ６２０は、ドメイン内のリソース管理ポリシー６３０を構成し得る。リソース管理ポリシー６３０は、少なくともリソース予約６３１、リソース制約６３２およびリソース通知６３３を含むように構成することができる。ドメインは、加えて、パーティションＡ６４０およびパーティションＢ６５０を含む。パーティションＡおよびＢは、それぞれ、リソースグループ６４２および６５２を含み、リソースグループ６４２および６５２は、それぞれ、仮想ターゲットＡ６４５および仮想ターゲットＢ６５５に関連付けられている。図６に示されるように、パーティションＡまたはパーティションＢはまた、たとえば、これらパーティションが共有リソースの使用を必要とするアプリケーションを実行する際に、共有リソース６６０にアクセスすることもできる。共有リソース６６０は、ＣＰＵ、ヒープ、ネットワークおよびファイルなどのＪＤＫリソースを含み得るが、これらに限定されない。

【００６８】

一実施形態に従うと、リソース管理ポリシーは、リソース管理ポリシーによって構成される条件が共有リソースに関してパーティションＡまたはパーティションＢによって満たされる場合に、動作またはタスクを実行するように、システムアドミニストレータによって、または別の適切な許可を行なうものによって、構成することができる。これらの動作またはタスクは、制約または通知を含み得るが、これらに限定されない。制約の例は、パーティションによる共有リソースの使用の減速または停止を含む。通知の例は、アドミニストレータに対する通知の記録または提供を含むが、これらに限定されない。

【００６９】

一実施形態に従うと、図６に示されるアプリケーションサーバ環境はOracle WebLogic serverであり得る。WebLogicサーバは、ClassLoaderベースの分離（アプリケーションの共有ライブラリに属さないクラスが互いから分離される）と、WebLogicワークマネージャ特徴とを含み得る。WebLogicワークマネージャ特徴は、１セットのスケジューリングガイドラインを構成することによって、アプリケーションが如何にその作業の実行に優先順位を付けるかをアドミニストレータが構成することを可能にし得る。しかしながら、これらの２つの特徴は、アプリケーションに提供された分離の点ではかなり制限されてしまい、ＷＬＳ－ＭＴデプロイメントにおいては、すべてのＪＤＫリソースなどのすべてのリソースが分割されて、それらの消費がパーティションに対する性能分離を保証するように管理されることが望ましい。

【００７０】

ソフト分離およびハード分離

一実施形態に従うと、分離された実行環境と共有された実行環境との相違はバイナリではなく、共有インフラストラクチャおよび完全分離型（shared-infrastructure-and-completely-isolated）の実行環境と、非分離型（non-isolated）の実行環境との間に一連の選択肢が存在し得る。

【００７１】

一実施形態に従うと、同じアプリケーションサーバ環境におけるさまざまなパーティションにおいてシステムアドミニストレータが非信頼型テナントからのコードをプロビジョニングするために、システムは、分離された実行環境（たとえば、ＭＶＭ／分離保護、分離されたＶＭ実行時間－［ＧＣ、ヒープ、スレッド、システムクラス］、デバッグおよび診断など）のためのサポートを提供することができる。このため、当該システムは、パー

ティションにおいて起こる動作を互いからサンドボックスする（sandbox）ことができる。これは実行分離とも称することができる。これはハード分離の一形式と見なされてもよい。

【００７２】

ＪＤＫ　８ｕ４０によって公開されたＲＭ　ＡＰＩからのサポートにより、システムはまた、よりソフトな形式の分離を提供することができる。この場合、いくつかの共有されたＪＤＫリソースへのアクセスが分離され、これらの分離は返還要求によって維持される。これは性能分離と称され得る。

【００７３】

ワークマネージャ（Work Manager：ＷＭ）との関係

一実施形態に従うと、アプリケーションサーバ（たとえばWebLogicサーバ）ワークマネージャー（ＷＭ）は、１セットのスケジューリングガイドライン（最大／最少および容量制約、応答時間要求クラス）を構成することによって、アプリケーションがその作業の実行に如何に優先順位を付けるかを、アドミニストレータが構成することを可能にし得る。これにより、ＷＭが、アプリケーションに割当てられたスレッドを管理し、それらのスレッドに沿ってスケジューリングワークインスタンスを管理し、アグリーメントの維持を支援することが可能になり得る。

【００７４】

一実施形態に従うと、ＷＭは、そのスケジューリングポリシーに関するワークインスタンスの実行のために割当てられた時間を用いる。ワークインスタンスのための経過時間は、ＣＰＵ時間と非ＣＰＵ時間との組合せ（Ｉ／Ｏ上での待機、システムコール特有のコールなど）であってもよく、いくつかの使用事例（たとえば、Ｉ／Ｏ向けのワークロード）に対するポリシーを指定するためのより適切な基準である。

【００７５】

リソース消費管理（ＲＣＭ）による拡張

一実施形態に従うと、ＷＭパーティションの公平共有では、公平共有分を演算する際にパーティションのワークインスタンスを実行するための経過時間を用いることができる。WebLogicサーバなどのアプリケーションサーバ環境においてパーティションが並べて配置された状態では、ＣＰＵレベルにおいても同様に確実に性能分離することが重要である。

【００７６】

たとえば、パーティションＰ１がＩ／Ｏモーストリ・ワークロード（I/O-mostly workload）を有し、Ｐ２がＣＰＵインテンシブ・ワークロード（CPU-intensive workload）を有する場合、Ｐ１およびＰ２はともに同じＷＭの公平共有分を有する。ワークインスタンスがＰ１によって提示され、同時に完了されるべきであると想定すると、Ｐ１およびＰ２は、ＷＭによって同様の態様でスケジューリングされるだろう（たとえば、Ｐ１およびＰ２のために同数のワークインスタンスがスケジューリングされて実行されるだろう）。しかしながら、Ｐ２のワークロードがＣＰＵインテンシブであるので、Ｐ２の方がＰ１よりも多く共有ＣＰＵを使用する。これは望ましくないかもしれない。

【００７７】

一実施形態に従うと、ＪＤＫ　ＲＭ　ＡＰＩなどのＡＰＩによって提供されるＣＰＵ時間使用通知は、リソースコンテキストに蓄積されたスレッドごとのＣＰＵ時間を構成している。これにより、システムアドミニストレータが、ポリシーを特定して、ＣＰＵレベルで並置パーティション間の性能分離を達成するための別の直交的方法が提供される。ＣＰＵ時間リソースのために利用可能な公平共有ポリシーとＷＭパーティションの公平共有ポリシーとはともに、さまざまな結果を達成するためにシステムアドミニストレータによって有意義に採用され得る。Ｉ／Ｏ関連の性能分離のために、ファイルおよびネットワークリソースレベルでの公平共有ポリシーも、ＲＣＭ特徴によって利用可能である。

【００７８】

一実施形態に従うと、ＷＭは、それが管理するスレッドに対してのみ、そのポリシーを実施することができる。リソースコンテキストの一部でありＷＭによって管理されないス

レッド（たとえば、パーティションに常駐するスレッド大量生成型アプリケーション）が存在する可能性もあり、ＲＣＭポリシーはそれらを管理するために用いることができる。スタックスレッド状況に加えてＪＤＫ　ＲＭ　ＡＰＩサポートによって、システムは、暴走してデッドロックとなったスレッドを判定してマーク付けし、システムアドミニストレータがそれらスレッドに対する措置を講じることを可能にし得る。

【００７９】

ルールベースの融通性と監視／通知との関係

一実施形態に従うと、WebLogic診断フレームワーク（WebLogic Diagnostics Framework：ＷＬＤＦ）などの診断フレームワークの観察および通知コンポーネントは、システムアドミニストレータが、サーバおよびアプリケーション状態を監視し、基準に基づいて通知を送信することを可能にする。これらの観察は、リソースが消費された後にシステムの状態についてのメトリクスを監視する際に機能する。ＲＣＭポリシーは消費要求中にポリシーを実施することを可能にする。ＲＣＭポリシーは、アドミニストレータがリソースコンシューマによってそれらの共有リソースの使用を正確に具体化するのを助ける。

【００８０】

一実施形態に従うと、ルールベースの融通性特徴は、ＷＬＤＦ観察／通知特徴を基礎にして、システムアドミニストレータが、スケーリング管理アクションを行うためにクラスタにわたってリソース使用を監視する複雑なルールを構築することを可能にする。ルールベースの融通性は、共有されたコンポーネントに対する個々のコンシューマによる個々のリソース消費要求に焦点を合わせて、ミクロレベルで複数の作業に影響を及ぼすアクションを監視および実行するために「マクロ」レベルで（たとえば、ＷＬＳクラスタ全体にわたる、またはサービスにわたる）履歴傾向の使用に焦点を合わせることができ、かつ、典型的には、結果として、そのリソースの特定的なさらに他の使用（たとえば現在のＶＭだけに影響するアクション）に変化をもたらすポリシー（返還要求）に焦点を合わせることができる。

【００８１】

一実施形態に従うと、ＪＤＫ　ＲＭ　ＡＰＩは、リソースコンテキスト（たとえばパーティション）ごとに、ＪＤＫによって管理されたリソースのためにいくつかのリソース消費メトリクスに関する情報を提供することができる。これらは、対応するPartitionRuntimeMBeanについての属性として表面化させることができるように、パーティション範囲指定された監視特徴に渡すことができる。

【００８２】

パーティションのライフサイクルイベントと開始／停止のサポートとの関係

一実施形態に従うと、ＲＣＭフレームワークは、スレッドにおける正確なリソースコンテキストを確立することができる。このため、そのスレッドのコンテキストにおいて消費されるすべてのリソースが、インプリメンテーションによって、そのリソースコンテキストに対して適切な割合を占めることになる。ＲＣＭフレームワークは、WebLogicサーバイベント特徴などのアプリケーションサーバ特徴によって提供されるパーティション開始／停止イベントを用いる（可能であれば、可能化／不可能化する）ことにより、新しいパーティションをいつ開始または停止させるかを認識し、新しいリソースコンテキストを作成または削除することができる。

【００８３】

一実施形態に従うと、パーティションファイルシステムが仮想ファイルシステムを実現しない可能性があるので、java.[n]ioを介してパーティションのファイルシステムにアクセスすることができ、アプリケーションサーバは、それに対するファイルＩ／Ｏ操作を阻止することができない可能性がある。しかしながら、ＲＣＭフレームワークはＪＤＫファイル記述子およびファイル関連リソースを、直接、システムアドミニストレータに対して公開することができる。

【００８４】

協同メモリ管理（Co-operative Memory Management：ＣＭＭ）との関係

一実施形態に従うと、協同メモリ管理（ＣＭＭ）特徴は、システムのさまざまな部品をメモリ圧縮に反応させることによって、メモリ不足状況に対処するために用いることができる。メモリ圧縮は外部から判断することができる。次いで、システムに常駐するアプリケーション、ミドルウェアおよびＪＶＭはメモリ圧縮レベルに反応し得る。たとえば、高メモリ圧縮状況においては、使用頻度の最も低いアプリケーションはアイドリングにされる可能性／休止状態にされる可能性があり、ＪＤＢＣ／ＪＭＳおよびコヒーレンスサブシステムはそれらのキャッシュサイズを小さくする可能性があり、ＪＶＭはそのヒープサイズなどを小さくする可能性がある。

【００８５】

一実施形態に従うと、ＣＭＭ特徴は、マクロ（マシン）レベルでメモリ圧縮シナリオに対処するために用いることができ、高メモリ状況の発生後に反応し得る。マシンにおける高メモリ圧縮は、マシンにおける外部の不規則なＪＶＭまたはプロセスの結果である可能性もあり、メモリ圧縮の「ソース」を制御したり、（割当てられて保持されたメモリ容量）の点で異常なメモリ使用を制御したりするための措置を取る必要はない。ポリシーは、「ミクロの」（ＷＬＳサーバＪＶＭ）レベルで動作可能であり、ヒープの不規則な「リソースコンシューマ」の使用を積極的に制御することができる。したがって、ＣＭＭ特徴およびシステムアドミニストレータによるＲＣＭポリシーの適用は、本来補足的なものである。

【００８６】

リソース管理サポート

一実施形態に従うと、ＪＤＫ８ｕ４０などの開発キットにおいて提供されるリソースマネージャＡＰＩは、リソース計測およびリソース消費の通知からリソースポリシーの実施を分離することができる。ＲＭ　ＡＰＩは、リソース消費マネージャなどの外部マネージャが、アプリケーションドメインにおけるリソースコンシューマスレッドをリソースコンテキスト（resourcecontext）に関連付けることを可能にし得る。このリソースコンテキスト（resourcecontext）は、リソース消費要求を明らかにするために、開発キットのためのコンテキストを提供することができる。ＲＭ　ＡＰＩはまた、リソース消費マネージャなどの外部マネージャが、リソースメータ（resourcemeters）を介してリソースコンテキスト（resourcecontext）によって（リソースＩＤ（resourceIds）によって識別される）特定のリソースの消費についての情報を取得するための関心事項を登録することを可能にし得る。

【００８７】

一実施形態に従うと、ＪＤＫなどの開発キットによって提供される標準的なリソースメータ（resourcemeter）インプリメンテーションは：simplemeter（リソースコンテキストについての正常なリソース割当ての単純なカウント）；boundedmeter（上限が設けられた状態でのカウント）；notifyingmeter（リソース割当てよりも前に、リソースの構成済み粒状区域についての、リソースマネージャからの承認をカウントして要求する）；throttlemeter（消費を毎秒ごとに特定の割合に制限する）である。

【００８８】

一実施形態に従うと、単純な境界付けされたメータなどのいくつかのメータがプルの態様で情報を取得する。リソース消費要求を受領するか減速させるかまたは拒否するためのポリシーをリソースマネージャに実施させるために、notifyingmeterなどの他のメータは、リソースマネージャに事前消費通知コールバックを送信することができる。

【００８９】

ＲＭ対応のＪＤＫリソース

一実施形態に従うと、ＪＤＫによって管理されたリソースの例は、ＲＭ対応可能にすることにより、システムアドミニストレータがそれらリソースに関するリソース管理ポリシーを指定することを可能にし得るものであって、オープンファイル記述、ヒープ、スレッド、ＣＰＵ時間などを含む。加えて、以下のリソースは、リソース消費メトリクス：ファイル記述子、ファイル（たとえばオープンファイルカウント、送信されたバイト、受信さ

れたバイト、合計）、ソケットおよびデータグラム、ヒープ、スレッド、アクティブなスレッドカウント、ＣＰＵ時間、として利用可能にすることができる。

【００９０】

一実施形態に従うと、ＣＰＵ時間測定値は、各々のResourceContextにおいてアクティブなスレッドをサンプリングしてメータに更新を送信することができるスレッドを監視するＪＤＫなどの開発キットによって定期的に実行することができる。

【００９１】

一実施形態に従うと、Ｇ１ガーベッジコレクタは領域ベースのアロケータであってもよい。ガーベッジコレクションが実行され、オブジェクトがコピーされると、ガーベッジコレクタは、オブジェクトが同じリソースコンテキストを有する領域にコピーされることを確実にすることができる。

【００９２】

リソース消費管理－リソース、トリガー

一実施形態に従うと、システムアドミニストレータは、リソース・コンシューマごとにＲＭ対応のリソース周囲のＲＣＭポリシーを指定することができる。リソースに対して、システムアドミニストレータは、１つ以上のトリガー（たとえば、リソースの最大使用が４００単位に限定される）と、トリガーがその値に到達したときに実行されなければならないアクションとを指定することができる。これらのアクションは、結果として、リソース消費要求が（同期して）なされていた同じスレッドにおいてアクティビティを発生させる可能性があるか、または、リソース消費要求が（非同期的に）なされたのとは異なるスレッドにおいて実行される可能性がある。トリガー／アクションの組合せは、システムアドミニストレータがリソースコンシューマによってリソースの使用を具体化し、制御し、制限することを可能にし得る。

【００９３】

一実施形態に従うと、システムアドミニストレータが指定することができるいくつかのアクションが存在する。これらアクションは、通知、減速、失敗およびシャットダウンを含むが、これらに限定されない。通知アクションは、通知を情報の更新としてシステムアドミニストレータに提供する。減速アクションは、リソースが消費される割合を調節する（たとえば減速させる）ことができる。失敗アクションは１つ以上のリソース消費要求に対する失敗であり得る。失敗アクションは、リソースの使用が所望の限度を下回ると終了し、これによりリソース消費を再び可能にすることもできる。シャットダウンアクションは、ＳＩＧＴＥＲＭに相当するもの（たとえば終了を要求するためにプロセスに送信される信号）であってもよく、コンシューマがクリーンアップできるようにした状態でリソース消費を停止させようとする試みであってもよい。

【００９４】

一実施形態に従うと、ＲＣＭは、ＪＳＲ－２８４ＡＰＩに基づいたWebLogicサーバＲＣＭフレームワークであり、WebLogicコンテナ、コンポーネントおよびツールによって使用されるべき軽量のＲＣＭ　ＳＰＩ（Service Provider Interface：サービスプロバイダインターフェイス）およびＡＰＩを提供する。

【００９５】

ここで、一実施形態に従ったリソース消費管理インプリメンテーションを示す図７を参照する。図示されるように、リソース消費管理インプリメンテーション７００は、リソースコンシューマ７０５、リソースドメイン７１０、リソース属性７１５、リソース７２０、リソースディスペンサ７２５、制約７４０、通知７５０、トリガー７６０、およびＪＤＫファイルＩ／Ｏサブシステム７３０を含み得る。図７に示されるように、トリガー７６０は、たとえば図６に示されるようなリソース消費管理ポリシー６３０内においてシステムアドミニストレータによって設定され得る。

【００９６】

リソースコンシューマ７０５、たとえばパーティション、がリソースを消費し始めると、それらのリソースの使用が監視される。たとえば、消費されたリソースがＣＰＵ、ヒー

プ、ファイルおよびネットワークを含み得ると想定すると、リソース消費管理インプリメンテーションにより、消費されたリソースが監視され、システムアドミニストレータによって設定／構成／指定されるトリガー７６０に対して比較される。

【００９７】

たとえば、図７に示されるように、システムアドミニストレータによって設定されるトリガーは、リソースの使用が７０以上（たとえば、ＣＰＵ時間の７０％）になったときに、通知をシステムアドミニストレータに記録する／表示させるためのものである。ＣＰＵ時間がリソースコンシューマに関して７０％を越えると、リソース消費管理ポリシーはログ７５５をファイルへとプッシュすることができる。ログファイルは、リソースコンシューマがトリガー内でシステムアドミニストレータによって設定されるようなしきい値を越えたことを示している。

【００９８】

一実施形態に従うと、システムアドミニストレータが「通知する」ことをトリガーにおける返還要求タイプとして指定すると、ＷＬＳ　ＲＣＭフレームワークは、トリガーにおいて指定されている使用に到達したことについての標準的メッセージを記録することができ、さらに、所要の補足属性を備えたメッセージ（たとえば、現在の使用、前回の使用、パーティションに関して到達した割当て分の通知）をメッセージの一部としてログ記録する。システムアドミニストレータは、標準的ログメッセージに従うように観察ルールを構成するためにＷＬＤＦシステムモジュールをデプロイして、通知を送信するためにフレームワークにおける通知機能を用いてもよい。

【００９９】

加えて、図７に示されるように、リソースコンシューマのリソースの使用が１００以上である（たとえば、オープンファイル記述子である）場合、制約がなされる。たとえば、リソースコンシューマ（たとえばパーティション）が１００の限度を上回ると想定する。次いで、システムアドミニストレータによって設定されるトリガーに基づいて、制約が呼出されることとなり、たとえば、要求されたファイル７４５を開くことに失敗するという例外を投入することができる。

【０１００】

一実施形態に従うと、制約の別の例として減速が挙げられる。減速は、パーティションワークマネージャの公平共有分の減少を伴う可能性があり、これにより、ワークマネージャがスケジューリングするパーティションのためのワークインスタンスの数が少なくなるだろう。加えて、ワークマネージャはまた、パーティションの同時実行されるアクティブなワークインスタンスの最大数を減じるようにパーティションの最大スレッド制約を低減させることができる。減速はまた、アプリケーションによって作成されたスレッドの優先度を下げることができる。減速アクション／制約は、パーティションをさらに減速させるために複数回と呼出されてもよい。これにより、結果として、パーティションワークマネージャの公平共有、最大スレッド制約およびアプリケーションスレッドの優先度についての値をさらに小さくすることができる。これは、たとえば、公平共有ポリシーサポートの場合に行うことができる。同様に、減速条件は取消されてもよく、これにより結果として、パーティションワークマネージャの公平共有および最大スレッド制約を、もともと構成されていた値に戻すこととなり、さらに、アプリケーションスレッドについての通常の優先度を回復させることにもなるだろう。

【０１０１】

一実施形態に従うと、シャットダウンパーティションアクションは、トリガーが対処された後の返還要求として指定することができる。システムアドミニストレータが返還要求のうちの１つとしてシャットダウンを指定すると、指定された割当て分に違反するリソース消費を伴うパーティションがシャットダウンさせられることとなる。

【０１０２】

一実施形態に従うと、パーティションがシャットダウンすると、適切なクリーンアップが行なわれることとなり、対応するJDK ResourceContextsを閉じることができる。パーテ

ィションおよびそのアプリケーションは、パーティションがシャットダウンさせられると、アクセス可能ではなくなるだろう。

【０１０３】

リソースプラグ接続可能性

一実施形態に従うと、リソースインプリメンタは、ＳＰＩを用いて、それらのリソース属性をリソース消費管理フレームワークに登録することができる。アドミニストレータは、これらのリソース属性に関するリソース管理ポリシーを定義することができる。リソース属性により、リソースインプリメンタは、リソースのさまざまな特徴（使い捨て可能であること、制限されていること、予約可能であること、粒度、測定値の単位など）を記述し得る。リソースインプリメンタはまた、粒度、およびデフォルトの最大測定遅延、およびリソースのレート管理期間を決定することができる。リソースインプリメンタは、測定の性能影響を制御するようにこれらの値を選択することができる。

【０１０４】

一実施形態に従うと、リソースインプリメンタは、ある量のリソースが消費され得るかどうかをチェックするために、ＲＣＭフレームワークにｃｏｎｓｕｍｅ（）コールを挿入し得る。これにより、ＲＣＭフレームワークが、リソースへのアクセスを要求しているリソースコンシューマのためにリソースに関するリソース管理ポリシーをチェックして実施することが可能となる。使い捨て可能なリソースの場合、リソースインプリメンタは、再利用のために使用可能となるように、リソースがリソースコンテナによってこれ以上使用されないことを示すために、ＲＣＭフレームワークにｒｅｌｉｎｑｕｉｓｈ（）コールを挿入し得る。

【０１０５】

一実施形態に従うと、ＲＣＭフレームワークは、指定された量がリソースコンシューマに提供され得るかどうかをチェックするために、適切なリソースドメインおよびその関連するポリシーを調べることができる。

【０１０６】

リソースコンシューマプラグ接続可能性

一実施形態に従うと、ＲＣＭフレームワークは、新しいリソースコンシューマタイプがＳＰＩを介して差し込まれることを可能にし得る。リソースコンシューマはたとえばパーティションであり得る。リソースディスペンサは、リソースコンシューマごとにリソースの使用を決定し、分離し、区別することができる。たとえば、リソースコンシューマタイプがパーティションである場合、かつ、共有のデータソース上のオープン接続の数だけリソースがアクセスされている場合、データソースコンテナは、或るパーティションからの接続要求（およびその後、その要求が達成されたこと）を別のパーティションによる接続要求から区別することができる。リソースの使用が固有のリソースコンシューマに明確に割当てることができない場合、分離および公平性が保証されない可能性がある。

【０１０７】

Ｊａｖａ開発キットのリソースマネージャＡＰＩ

一実施形態に従うと、上述されたＲＭ対応のＪＤＫリソースは、JSR-284 ResourceAttributeによって表わすことができる。リソースを表わすプロキシResourceDispenserが実現可能であり、ＪＤＫ　ＲＭ　ＡＰＩとインターフェースをとることができる。

【０１０８】

一実施形態に従うと、ＷＬＳ　ＲＣＭインプリメンテーションは、パーティション始動イベントを受信することができ、始動したパーティションのためのResourceContextを作成することができる。JDK ResourceMetersは、そのパーティションについて指定されたＲＣＭポリシー構成に基づいて監視される必要のあるＪＤＫリソースごとに作成することができる。次いで、これらのJDK ResourceMetersは、パーティションについてのJDK ResourceContextに関連付けることができる。

【０１０９】

一実施形態に従うと、ＷＬＳ　ＲＣＭインプリメンテーションは、各々のJDK Resource

Meter（たとえば、NotifyingMeter）にJDK ResourceApprover（リスナー）を登録することができる。このリスナーはすべての事前消費割当て要求に基づきＪＤＫによって呼び戻すことができ、さらに、リスナーは、これを、ＲＣＭフレームワークにおける適切なJSR-284 ResourceDomainに委任することができる。ＲＣＭフレームワークは、セクション４．２．８に記載されたそのリソースコンテキストについてのリソース消費管理ポリシーを実現する。

【０１１０】

ここで、一実施形態に従った、リソース消費管理統合における対話を示すシーケンス図を例示する図８を参照する。図８に示されるシーケンスは、リソースを消費するであろう動作を実行するアプリケーション８０５から開始される。次いで、ＪＤＫ８１０が量を要求する。通知カウンタ（NotifyingCounter）８１５は、資源要求を実行し、次いで、リソースアプルーバ（ResourceApprover）８２０は消費動作を実行する。リソースドメイン（ResourceDomain）８２５は、パーティションの現在の提案された使用に関するインジケータを含み得る事前消費（preConsume）を実行する。次いで、システムアドミニストレータによって構成される制約８３０は、ポリシーを実行することができる。ポリシー８３５は、その設定に従って、アプリケーションによって要求されるリソースの量を戻すか、または、０のリソースを戻すことによってリソースのための要求を拒否するだろう。この決定は、システムアドミニストレータによって構成され得るポリシーに基づいている。

【０１１１】

スレッドにおける正確なコンテキストの確立

一実施形態に従うと、ＪＤＫ　ＲＭインプリメンテーションが、スレッドに関連付けられたResourceContext（リソースコンテキスト）へのスレッドにおいて生じるリソース消費の原因となるので、システムは、リソースのミスアカウントを防ぐためにスレッド上で正確なResourceContextを確立する必要があるかもしれない。ＷＬＳ　ＲＣＭであり得るＲＣＭは、ResourceContextのために用いられるべきパーティションを決定するために現在のスレッドのコンポーネント呼出しコンテキスト（Component Invocation Context：ＣＩＣ）を用いることができる。ＷＬＳ　ＲＣＭインプリメンテーションは、ＣＩＣがスレッドにおいて変化する場合、通知を受信するためのComponentInvocationContextChangeListenerを登録することができる。このＣＩＣ変更通知が受信され、ＣＩＣ変更がパーティションスイッチ（たとえばグローバルな遷移への／からのクロスパーティションまたはパーティション）によるものである場合、ＷＬＳ　ＲＣＭインプリメンテーションは、現在のResourceContextをスレッドからアンバインドし、新しいパーティションに関するResourceContextをバインドすることができる。

【０１１２】

ここで、一実施形態に従ったリソース消費管理インプリメンテーションを示す図９を参照する。図示されるように、リソース消費管理インプリメンテーション７００は、リソースコンシューマ７０５、リソースドメイン７１０、リソース属性７１５、リソース７２０、リソースディスペンサ７２５、制約７４０、通知７５０、トリガー７６０、およびＪＤＫファイルＩ／Ｏサブシステム７３０を含み得る。図示されるように、トリガー７６０は、たとえば、図６に示されるようにリソース消費管理ポリシー６３０内でシステムアドミニストレータによって設定され得る。加えて、リソース消費管理インプリメンテーションは、現在のスレッドのコンポーネント呼出しコンテキスト９２０を参照することによって設定することができるリソースコンテキスト（resourcecontext）９１０に関連付けられる。これにより、制約または通知が、トリガーされた時に、アクティブなリソース消費に関連付けられることを確実にすることができる。

【０１１３】

一実施形態に従うと、スレッドが新しいスレッドを大量に生成すると、ＪＤＫ　ＲＭＡＰＩは親スレッドにおいて確立されたResourceContextを自動的に引き継ぐことができる。

【０１１４】

ＪＤＫリソースメトリクス

一実施形態に従うと、さまざまなコンポーネントは、パーティション特有のリソース使用メトリクスを得ることができる。ＲＣＭインプリメンテーションは、ＡＰＩを介してパーティション特有のリソース消費メトリクスを取得するために、監視用のＡＰＩを公開することができる。

【０１１５】

ＣＰＵの利用

一実施形態に従うと、ＪＤＫリソース管理ＡＰＩは、リソースコンテキストのリソースコンシューマによって消費されたＣＰＵ時間の測定をサポートする。マルチパーティション環境においては、各々のパーティションがリソースコンテキストにマップ可能であり、スレッドはパーティションの代わりに機能するようにリソースコンテキストにバインドされるだろう。ＣＰＵ時間メトリクスを用いれば、ＲＣＭフレームワークは、パーティションによって利用されるＣＰＵ時間を測定することができる。しかしながら、絶対数（たとえば１時間３０分）でＣＰＵ使用をリソースとして表わしても、システムアドミニストレータにとって有用ではないだろう。なぜなら、ＣＰＵが非制限（無限）のリソースであるからであり、かつ、システムアドミニストレータが絶対的なＣＰＵ使用時間に換算してパーティションについての限度／割当て分を指定するには、実際には限定的な値（または非直観的）となるからである。このために、ＣＰＵ使用から導き出すことができるＣＰＵ利用リソースタイプはパーセント値（１～１００）によって表わすことができる。ＣＰＵ利用百分率は、システムプロセスに対して利用可能なＣＰＵに関して、パーティションによって利用されるＣＰＵの百分率を示している。この値は、パーティションおよびＣＰＵの負荷係数によりＣＰＵ消費百分率を用いて計算される。

【０１１６】

一実施形態に従うと、ＣＰＵ消費百分率は、ＷＬＳプロセスによって、ＣＰＵの総消費に対するパーティションのＣＰＵ消費に基づいて計算することができる。しかしながら、ＷＬＳプロセスの総ＣＰＵ消費に対するパーティションのＣＰＵ利用の共有分を考慮しても、ポリシーを設定する際にシステムアドミニストレータが参照するおよび／または基準にする有用な基準が常に得られるとは限らないだろう。

【０１１７】

たとえば、１つのパーティションだけがＷＬＳプロセスにアクティブに関与している場合、かつ、ＷＬＳプロセスが実行されているマシンに対する負荷が軽い場合、そのパーティションについてのＣＰＵ利用として、総ＷＬＳプロセスのＣＰＵ使用に対するパーティションのＣＰＵ使用の単比例を処理することにより、結果として、過剰表示となり（そのパーティションについてのＣＰＵ利用値が非常に高くなり）、そのパーティションが不必要に不公平に処理されることとなるだろう。

【０１１８】

追加の例として、２つのパーティションが１つのドメインにあると想定する。パーティション－１に対するＣＰＵ消費百分率は９５であり、パーティション－２に対するＣＰＵ消費百分率は４である。また、ＷＬＳプロセスが実行されているマシンに対する負荷は軽い。この場合であっても、総ＷＬＳプロセスのＣＰＵに対するパーティションのＣＰＵ使用の単比例を処理することにより、結果として、パーティション－１が不必要に不公平に処理されることとなるだろう。

【０１１９】

一実施形態に従うと、システムに過度に負荷が加えられていない上述のシナリオを、アプリケーションサーバがＣＰＵを過度に使用しているかまたは外部プロセスがＣＰＵを過度に使用しているせいでシステムに重い負荷が加えられる可能性のある他のシナリオから区別するために、追加の負荷係数をＣＰＵ消費百分率に適用することができる。この追加の係数はＣＰＵ負荷であり、１つ以上のＡＰＩを用いて計算することができる。

【０１２０】

一実施形態に従うと、サーバにおいてアクティブなパーティションが１つしかない状況

では、負荷係数はProcessCPULoadおよびCONTENTION_THRESHOLDに基づき得る。CONTENTION_THRESHOLD値は、たとえば０．８（またはＣＰＵの８０％）であり得る。

【０１２１】

一実施形態に従うと、サーバにおいてアクティブなパーティションが２つ以上ある状況においては、ＲＣＭは、ProcessCPULoad値が８０を超える（CONTENTION_THRESHOLD）場合に、ProcessCPULoad値を用いて負荷係数を計算することができる。これは、システム（たとえばＷＬＳ）プロセスがその期間にわたって８０％を超えるＣＰＵを利用していることを示している。

【０１２２】

一実施形態に従うと、サーバにおいてアクティブなパーティションが２つ以上ある状況においては、かつ、ProcessCPULoad値が０．８未満（またはＣＰＵの８０％）である場合には、これは、システム（たとえばＷＬＳ）プロセスがＣＰＵインテンシブではないことを示し得る。このような場合、ＲＣＭは、SystemCPULoadが０．８（またはＣＰＵの８０％）よりも大きいかどうかをチェックすることができる。０．８（ＣＰＵの８０％）よりも大きい場合、SystemCPULoadを用いて負荷係数を計算することができる。これはまた、ＣＰＵインテンシブであるシステムにおいて他のプロセスが存在し、ＪＶＭプロセスには制限されたＣＰＵが提供されることを示している。

【０１２３】

一実施形態に従うと、サーバにおいてアクティブなパーティションが２つ以上ある上述のいずれかの状況においては、ＣＰＵ使用がCONTENTION_THRESHOLDを上回る場合、これは、システムに十分に負荷が加えられており、パーティションがＪＶＭプロセスのために利用可能なＣＰＵに対して（たとえば、リソース競合が存在している）他のパーティションを犠牲にしてＣＰＵを消費している可能性があることを示している。

【０１２４】

一実施形態に従うと、サーバにおいてアクティブなパーティションが２つ以上ある状況においては、ＲＣＭは、ProcessCPULoadまたはSystemCPULoadがそれらのCONTENTION_THRESHOLDを上回ったかどうかに基づいて、ProcessCPULoadまたはSystemCPULoadを負荷係数と見なすことができる。

【０１２５】

一実施形態に従うと、サーバにおいてアクティブなパーティションが２つ以上ある状況においては、ProcessCPULoadおよびSystemCPULoadがともにそれらのCONTENTION_THRESHOLD未満である場合、負荷係数はProcessCPULoadおよびCONTENTION_THRESHOLDに基づくだろう。

【０１２６】

一実施形態に従うと、ＣＰＵ負荷係数は、ＣＰＵに対する競合の判断を支援することができ、これにより、ＪＤＫリソース管理ＡＰＩに基づいて導き出されたＣＰＵ消費メトリクス値をさらに定量化することができる。

【０１２７】

一実施形態に従うと、ＣＰＵ利用値を用いて、通知、減速、シャットダウンおよび公平共有のようなさまざまなポリシーおよびリコースを構成することができる。ＣＰＵ時間消費についてのＪＤＫからの通知が消費後のコールであるので、要求ごとの失敗返還要求をＣＰＵ利用リソースタイプのためにサポートすることができない。

【０１２８】

一実施形態に従うと、ＣＰＵ利用値の減衰平均が或る期間にわたって一貫してしきい値を上回っている場合、通知、減速、シャットダウンなどの返還要求が取られる。スパイク（突発的な消費の急上昇／急降下）を回避／無視するのに役立つ一般的なポーリングメカニズムベースのアルゴリズムは、保持されたヒープおよびＣＰＵ利用リソースタイプの両方によって使用され得る。

【０１２９】

一実施形態に従うと、公平共有ポリシーは、サーバにおけるすべてのパーティションの

公平共有値に基づいて機能するＣＰＵ利用リソースタイプのためにサポートされる。
【０１３０】
保持されたヒープ
一実施形態に従うと、ＡＰＩ（たとえば、ＪＤＫリソース管理ＡＰＩ）は、特定のリソースコンテキストにおいて保持されたヒープおよびヒープ割当ての測定をサポートすることができる。これらのリソースタイプの両方のためのコールバックは消費後に行なわれる。このために、ヒープ消費を制御するために事前に先回りしてアクションを実行することができない。
【０１３１】
一実施形態に従うと、ヒープ割当ての場合、ＪＤＫ　ＲＭは、特定のリソースコンテキストのためにヒープ割当ての現在累計高のためにコールバックを提供する。ＪＤＫからのヒープ割当て通知は、単調に増大し得るものであって、一般には、そのリソースコンテキストによる使用時には実際のヒープを反映していない。ＷＬＤＦルールは、ヒープ割当て済みのパーティションによって範囲指定されたメトリクスを使用することによってヒープ割当てを監視するように構成することができる。
【０１３２】
一実施形態に従うと、保持されたヒープの場合、コールバックはガーベッジコレクタアクティビティの結果としてＪＤＫから発信可能である。ヒープ保持通知は、関連する精度レベル（たとえば信頼度係数）を有し得る。ヒープ使用はガーベッジコレクションが十分になされた後に高い信頼度で測定することができ、これにより、結果として、高精度の通知を得ることができる。その後、ヒープ使用の測定の信頼度が低下する。イベント完了を同時にマーク付けするＧ１ガーベッジコレクションは高い信頼度の値をもたらす可能性があり、これにより、高精度の通知がもたらされるだろう。ガーベッジコレクションがより少なければ、測定値に対する信頼度が低くなり、これにより、通知の精度がより低くなるだろう。次のガーベッジコレクションサイクルまで、ヒープ使用についてＪＤＫによって報告される値は、非直線的に低下する信頼度係数を有するだろう。保持されたヒープ使用情報の精度は、（リソースコンテキストのためのガーベッジコレクション後に残されたすべてのオブジェクトにタグ付けし、サイズを合計している点で）正確であり、かつ、高価であり得るか、または、（リソースコンシューマに割当てられたＧ１領域をカウントする点で）より粗くかつより安価である。ガーベッジコレクションサイクル間の信頼度係数は、過去の割当て履歴に基づき得るものであって、定常的な割当てを想定し得る。
【０１３３】
一実施形態に従うと、マルチパーティション環境においては、各々のパーティションは、リソースコンテキストにマップすることができ、スレッドは、パーティションの代わりに作業を実行する際に適切なリソースコンテキストにバインドされるだろう。これは、リソースが保持されたヒープを読取ることを可能にし、そのリソースコンテキストに対する消費の割合が正確になる。保持されたヒープがＪＤＫからコールバックされると、ＲＣＭフレームワークは、パーティションによって保持されているヒープの量を測定することができる。上述のとおり、高精度の通知を用いて、ヒープ保持型リソースタイプのためのＲＣＭポリシーを実施することができる。通知、減速、シャットダウンのような返還要求は、ＪＤＫからの単一の通知に基づいて呼出される必要はない。しかしながら、それらのアクションは、ＪＤＫについてのヒープ保持型通知の減衰平均が、一定期間にわたって一貫して、構成されたしきい値を上回っているときに、呼出すことができる。デフォルト期間を設定することができる。たとえば、このメカニズムに基づいてアクションをトリガーするために用いられるデフォルト期間は１００秒である。スパイク（突発的な消費の急上昇／急降下）を回避／無視するのに役立つ一般的なポーリングメカニズムベースのアルゴリズムは、保持されたヒープおよびＣＰＵ利用リソースタイプの両方によって使用され得る。
【０１３４】
一実施形態に従うと、サーバにおけるすべてのパーティションの公平共有値に基づいて

作用するヒープ保持型リソースタイプのために、公平共有ポリシーがサポートされる。
【０１３５】
ポリシー評価、返還要求／通知
一実施形態に従うと、パーティション（たとえばマルチパーティション環境内のパーティション）はリソースマネージャに関連付けることができる。いずれのリソースマネージャもパーティションに関連付けられていなければ、パーティションは如何なるリソース消費管理ポリシーにも準拠せず、リソースのその使用は制約されない。
【０１３６】
一実施形態に従うと、パーティションのために定義されたリソースマネージャごとに、ＲＣＭフレームワークは、ResourceIdを表わす適切なResourceAttributeのためにJSR-284 ResourceDomainをインスタンス化することができる。ＪＳＲ２８４の制約および通知は、ユーザ／システムアドミニストレータによって指定されたトリガーに基づいて、このResourceDomainと対照して登録される。
【０１３７】
一実施形態に従うと、システムアドミニストレータは、リソースの使用を管理するために１つ以上のトリガーを指定することができる。各々のトリガーは名前、値の対およびアクションを含み得る。リソース消費レベルがシステムアドミニストレータによって設定された値に到達すると、一連のアクションを達成するために、システムアドミニストレータによってトリガーが指定され得る。ＷＬＳ　ＲＣＭフレームワークは、使用が任意の値に達すると、関連するアクションを実行する。
【０１３８】
一実施形態に従うと、アクション要素によって指定された返還要求タイプは、通知、減速、失敗、シャットダウンといった４つのタイプのうちの１つであり得る。
【０１３９】
一実施形態に従うと、システムアドミニストレータによって指定される返還要求タイプを実現するためにＷＬＳ　ＲＣＭフレームワークによって実行される特定のセットのアクティビティは、ユーザによって定義され得るかまたは既存のマトリックスから選択され得る。
【０１４０】
たとえば、一実施形態に従うと、システムアドミニストレータは、単に２０００個のファイル記述子を用いるためにだけパーティションＰ１を必要とし得る。使用が特定値に達すると、システムアドミニストレータは、そのパーティションがそれ以上のファイル記述子を使用することを防ぐことを所望するだろう。このようなトリガーは以下のように表現される可能性もある：
【０１４１】
【数１】

```
<trigger>
<value>2000</value>
<action>fail</action>
</trigger>
```

【０１４２】
指定された失敗アクションは、ファイルオープン要求に対するIOExceptionの投入にマップすることができる。ＷＬＳ　ＲＣＭフレームワークは、パーティションＰ１についてのFILE_OPEN ResourceTypeを表わしているResourceAttributeについてのResourceDomainに制約を登録することによって、ファイルオープン要求を阻止することができる。
【０１４３】
一実施形態に従うと、システムによって実行される返還要求のうちのいくつかが消費の動作中に行なわれてもよい（たとえば、オープンファイル記述子リソースのために設定さ

れたＦＡＩＬトリガーにより、結果として、新しいファイルの生成中にIOExceptionが投入されることになる）。加えて、いくつかのアクションは、現在の消費要求の後に実行することができる（たとえば、ＣＰＵおよびヒープ使用についての通知が非同期的であり、すなわち、それらのリソースに基づいたトリガーに対する（結果として、関連するパーティションをシャットダウンすることとなる）「シャットダウン」返還要求が非同期的に実行される）。

【０１４４】

一実施形態に従うと、同期的なアクションはＪＳＲ－２８４制約により実現することができ、非同期的なアクションはＪＳＲ－２８４通知により実現することができる。

【０１４５】

公平共有ポリシーのインプリメンテーション

一実施形態に従うと、マルチテナントアプリケーションサーバ環境（たとえば、マルチパーティション環境におけるさまざまなパーティション）において、システムアドミニストレータがさまざまなリソースコンシューマに対してＲＭ対応の共有リソースを「公平に」割当てることを定性的に保証し、かつ定量的に確実にすることが望ましい。

【０１４６】

一実施形態に従うと、同じ公平共有値を有する２つのリソースドメインに対するリソースの割当ては、そのリソース割当てが不均等／不均一である場合、言い換えれば、一方のドメインのコンシューマのうちのいくつかに対するリソース割当てが他方のドメインと比べて偏っている場合、不公平であると見なされる。２つのリソースドメインは、さまざまなリソース要求パターンを有する可能性があり、いつでも即時に公平な共有を保証することが困難になる可能性がある。しかしながら、特に、パーティション間のリソース使用をそのリソースにとってのシステムの上限にまで推し進めているリソースにとって、両方のリソースドメインが競合／競争している場合には、リソース割当てをそれらの構成された公平共有の比率で維持することが依然として所望される。

【０１４７】

一実施形態に従うと、リソース割当てにおける公平性は、個々のリソースドメインからのリソース消費要求の変化を明らかにするために一定期間にわたって計算することができる。時間をかけて公平性を計算して実施することができるので、公平性は、必ずしも、特定の時間窓における分布の（公平な共有分が等しい場合の）均一性を示唆するものではない。公平共有のポリシーインプリメンテーションは、特定のリソースコンシューマがウィンドウにおけるリソースのうちそれらの共有分を使用しなかった場合には、そのコンシューマのために指定された公平共有分よりも大きなリソースの共有分を一時的に割当てることを可能にする。しかしながら、時間とともに、ユーザによって指定された公平共有分と一致するように割当てを調整することができる。

【０１４８】

公平共有のポリシーが一時的に過剰に割当てできる程度は、リソースのタイプにも基づき得る。たとえば、ＣＰＵ利用などの「非制限の」リソースの場合、公平共有のインプリメンテーションは、それらの共有分だけ遅れをとっているコンシューマにリソースを提供するために、その直近時点まで非常にアクティブであったリソースコンシューマに十分に不足を生じさせることができる。しかしながら、ヒープ割当て／保持されたヒープなどの「制限された」、「使い捨て可能な」および「取り消し不可能な」リソースの場合、公平共有のポリシーインプリメンテーションは、（許可されたいずれの割当ても後には無効にすることができないので）それらの共有分だけ遅れをとっていたリソースコンシューマからのすべてのヒープ割当て要求を許可しない可能性がある。

【０１４９】

一実施形態に従うと、公平共有のポリシーはシステムアドミニストレータに以下の保証を提供することができる。リソースに関する「競合」がある場合、および、「一定期間にわたって」２つのリソースドメインによる「均一な負荷」がある場合、２つのドメインに割当てられたリソースは、それらの２つのドメインのためにシステムアドミニストレータ

によって構成された公平共有分ごとに「大まか（roughly）」に共有される。

【０１５０】

一実施形態に従うと、公平共有のポリシーは、ＣＰＵ利用およびヒープ保持型リソースタイプのためにサポートされる。これらの両方のリソースタイプに関して、ＪＤＫからの通知が消費後に提供され得る。これが行なわれると、公平共有のポリシーは、リソース消費のための要求時に正確に適用することができなくなる。保持されたヒープの場合、ＪＤＫからの通知はガーベッジコレクション（garbage collection：ＧＣ）サイクルにリンクすることができ、ヒープ消費要求がなされる時点より後に行なわれる可能性がある。ＣＰＵ利用の場合、この利用は、固定された時間窓の後に決定することができる。

【０１５１】

一実施形態に従うと、リソース消費のための要求は、パーティションのワークマネージャによってパーティションのために行なわれている作業に間接的にリンクさせることができる。したがって、パーティションのために実行されているワークインスタンスをパーティションのリソース消費に基づいて制御することができれば、そのリソースのための公平共有が依然として保証され得る。

【０１５２】

一実施形態に従うと、保持されたヒープの場合、事前に割当てられた如何なるヒープもシステムによって強制的に無効にすることができないので、公平共有のポリシーは、パーティションのワークマネージャの公平共有を制御することによって、ヒープの割当てを制御する。

【０１５３】

ＲＣＭ構成

一実施形態に従うと、リソース消費マネージャ（resource consumption manager：ＲＣＭ）ポリシーおよび返還要求は、システムアドミニストレータなどのアドミニストレータによって構成することができる。システムアドミニストレータはまた、再使用可能な定義を生成し、パーティションのためにカスタマイズされたポリシーを生成することができる。

【０１５４】

一実施形態に従うと、システムアドミニストレータは、複数のパーティションにわたって次に再使用することができるリソースマネージャ定義を生成することができる。たとえば、ＳａａＳ使用事例においては、ドメインを管理するシステムアドミニストレータは、特定の「クラス」の顧客のために生成するすべての新しいパーティション（たとえばブロンズ、シルバー、ゴールドのクラス）に適用することが所望され得る１セットのＲＣＭポリシーを有してもよい。たとえば、ブロンズの顧客のパーティションは、ヒープおよびＣＰＵ使用に関するいくつかの管理されたリソースポリシーを有してもよい（たとえば、ヒープは２ＧＢに制限される）。新しいブロンズの顧客が登録され、代わりにパーティションＰ２が生成されると、システムアドミニストレータが、ドメインレベルでブロンズのリソース管理ポリシーを生成することが可能となり、ブロンズのパーティションの生成中にそれを指し示すことが可能となる。次いで、ブロンズのリソースマネージャに含まれているすべてのポリシー構成が、Ｐ２のリソース管理ポリシーに適用される。リソースマネージャが追加される各々のパーティションは、その構成において定義されている管理されたリソースポリシーの「コピー」を得る。このため、たとえば、ブロンズのリソースマネージャがＰ２パーティションおよびＰ３パーティションに適用された。Ｐ２およびＰ３の各々には２ＧＢのヒープ使用が許可される。

【０１５５】

一実施形態に従うと、システムアドミニストレータは、パーティションのためにカスタマイズされるポリシーを生成することができる。たとえば、統合の使用事例においては、システムアドミニストレータは、特定の部門のために生成されたパーティションについての固有のリソース消費管理ポリシーを指定することができる。そのパーティションについての新しいパーティション範囲指定されたリソース管理ポリシーを生成することが可能で

ある。たとえば、システムアドミニストレータは会社のＣＥＯのためのパーティションＣＥＯを作成する。次いで、システムアドミニストレータはまた、システムにおいて定義された他のパーティションとは異なるそのパーティションにのみ適用可能である（たとえば、比較的高い公平共有分をパーティションに与える）特定の管理されたリソースポリシーを定義することもできる。

【０１５６】

図１０は、一実施形態に従った、アプリケーションサーバ環境におけるリソースの分離および消費を示す。図１０は、ドメイン６１０を含むアプリケーションサーバ環境６００を示す。ドメインは、パーティションＡ６４０およびパーティションＢ６５０を含み得る。パーティションＡおよびＢは、それぞれ、仮想ターゲットＡ６４５および仮想ターゲットＢ６５５に関連付けられたリソースグループ６４２および６５２を含む。図６に示されるように、パーティションＡまたはパーティションＢはまた、たとえば、パーティションが共有リソースの使用を必要とするアプリケーションを実行する際に、共有リソース６６０にアクセスすることができる。共有リソース６６０は、ＣＰＵ、ヒープ、ネットワークおよびファイルなどのＪＤＫリソースを含み得るが、これらに限定されない。

【０１５７】

上述のとおり、システムアドミニストレータは、パーティション特有のリソース消費管理ポリシー１００５および１０１５をそれぞれ定義することができる。これらのパーティション特有のリソース消費管理ポリシーは、パーティション特有のリソース予約１００６および１０１６をそれぞれ含み、パーティション特有のリソース予約１００７および１０１７をそれぞれ含み、かつ、パーティション特有のリソース通知１００８および１０１８をそれぞれ含むように、構成することができる。

【０１５８】

一実施形態に従うと、パーティション特有のリソース消費管理ポリシーは、リソース管理ポリシーによって設定された構成済み条件が共有リソースに関してパーティションＡまたはパーティションＢのいずれかによって満たされたときに、動作またはタスクを実行するために、システムアドミニストレータによって、または適切な許可を有する別のものによって、構成することができる。これらの動作またはタスクは、制約または通知を含み得るがこれらに限定されない。制約の例は、パーティションによる共有リソースの使用の減速または停止を含む。通知の例は、アドミニストレータへの通知の記録または提供を含むがこれらに限定されない。

【０１５９】

以下は、指定されたパーティションについてのリソース管理ポリシーのための定義の具体的な一例である：

【０１６０】

【数２－１】

```
<domain>
...
   <!--Define RCM Configuration -->
   <resource-management>
      <resource-manager>
         <name>Gold</name>
         <file-open>
            <trigger>
               <name>Gold2000</name>
               <value>2000</value><!-- in units-->
               <action>shutdown</action>
            </trigger>
            <trigger>
               <name>Gold1700</name>
               <value>1700</value>
               <action>slow</action>
            </trigger>
            <trigger>
               <name>Gold1500</name>
               <value>1500</value>
               <action>notify</action>
            </trigger>
         </file-open>
         <heap-retained>
            <trigger>
               <name>Gold2GB</name>
               <value>2097152</value>
               <action>shutdown</action>
            </trigger>
            <fair-share-constraint>
               <name>FS-GoldShare</name>
               <value>60</value>
            </fair-share-constraint>
         </heap-retained>
      </resource-manager>
      <resource-manager>
         <name>Silver</name>
         <file-open>
            <trigger>
               <name>Silver1000</name>
               <value>1000</value><!-- in units-->
```

【０１６１】

【数２－２】

```
                <action>shutdown</action>
            </trigger>
            <trigger>
                <name>Silver700</name>
                <value>700</value>
                <action>slow</action>
            </trigger>
            <trigger>
                <name>Silver500</name>
                <value>500</value>
                <action>notify</action>
            </trigger>
        </file-open>
    </resource-manager>
</resource-management>
<partition>
    <name>Partition-0</name>
    <resource-group>
        <name>ResourceTemplate-0_group</name>
        <resource-group-template>ResourceTemplate-0</resource-
group-template>
    </resource-group>
    ...
    <partition-id>1741ad19-8ca7-4339-b6d3-78e56d8a5858</partition-
id>
    <!-- RCM Managers are then targeted to Partitions during
partition creation time or later by system administrators -->
    <resource-manager-ref>Gold</resource-manager-ref>
...
</partition>
...
</domain>
```

【０１６２】

一実施形態に従うと、すべてのＲＣＭ要素の動的な再構成をサポートすることができる。いくつかの状況においては、リソース消費管理ポリシーの再構成のためにパーティションおよびサーバを再始動させる必要はない。

【０１６３】

リソースの分離および消費

ここで、一実施形態に従った、アプリケーションサーバ環境におけるリソースの分離および消費のための方法についてのフローチャートを示す図１１を参照する。例示的な方法は、アプリケーションサーバ環境を含み当該アプリケーションサーバ環境が実行されている１つ以上のコンピュータにおいて、アプリケーションサーバ環境内で用いることができ

る複数のリソースと、各々がドメインの管理および実行時間下位区分を備える１つ以上のパーティションとを提供するステップ１１１０から開始可能である。当該方法は、複数のリソースについての各パーティションの使用を監視するためにリソース消費管理モジュールを構成するステップ１１２０に続き得る。リソース消費管理モジュールは、リソース予約、リソース制約およびリソース通知からなる群のうち少なくとも１つのメンバを含む。

【０１６４】

この発明は、この開示の教示に従ってプログラミングされた１つ以上のプロセッサ、メモリおよび／またはコンピュータ読取可能記憶媒体を含む、１つ以上の従来の汎用または特化型デジタルコンピュータ、コンピューティング装置、マシン、またはマイクロプロセッサを使用して都合よく実現されてもよい。ソフトウェア技術の当業者には明らかであるように、この開示の教示に基づいて、適切なソフトウェアコーディングが、熟練したプログラマによって容易に準備され得る。

【０１６５】

実施形態によっては、本発明は、本発明のプロセスのうちいずれかを実行するためにコンピュータをプログラムするのに使用できる命令が格納された非一時的な記録媒体または（１つまたは複数の）コンピュータ読取可能な媒体であるコンピュータプログラムプロダクトを含む。この記録媒体は、フロッピー（登録商標）ディスク、光ディスク、ＤＶＤ、ＣＤ－ＲＯＭ、マイクロドライブ、および光磁気ディスクを含む、任意の種類のディスク、ＲＯＭ、ＲＡＭ、ＥＰＲＯＭ、ＥＥＰＲＯＭ、ＤＲＡＭ、ＶＲＡＭ、フラッシュメモリデバイス、磁気もしくは光カード、ナノシステム（分子メモリＩＣを含む）、または、命令および／もしくはデータを格納するのに適した任意の種類の媒体もしくはデバイスを含み得るものの、これらに限定されない。

【０１６６】

本発明のこれまでの記載は例示および説明を目的として提供されている。すべてを網羅するまたは本発明を開示された形態そのものに限定することは意図されていない。当業者には数多くの変更および変形が明らかであろう。実施の形態は、本発明の原理およびその実際の応用を最もうまく説明することによって他の当業者がさまざまな実施の形態および意図している特定の用途に適したさまざまな変形を理解できるようにするために、選択され説明されている。本発明の範囲は添付の特許請求の範囲およびそれらの同等例によって規定されるものと意図されている。

【図１】

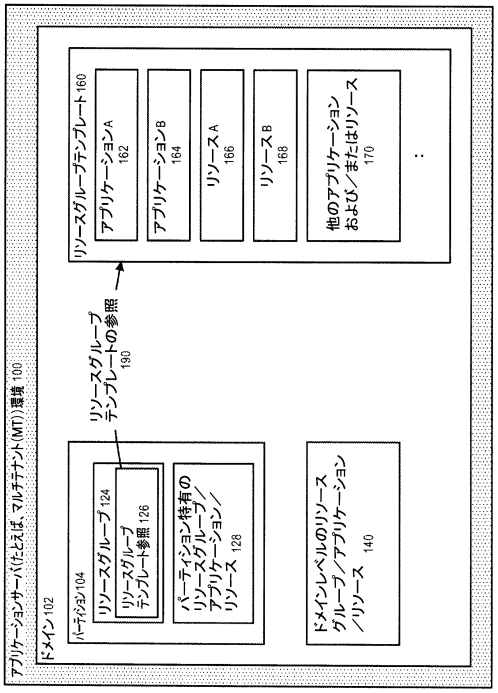

FIGURE 1

【図２】

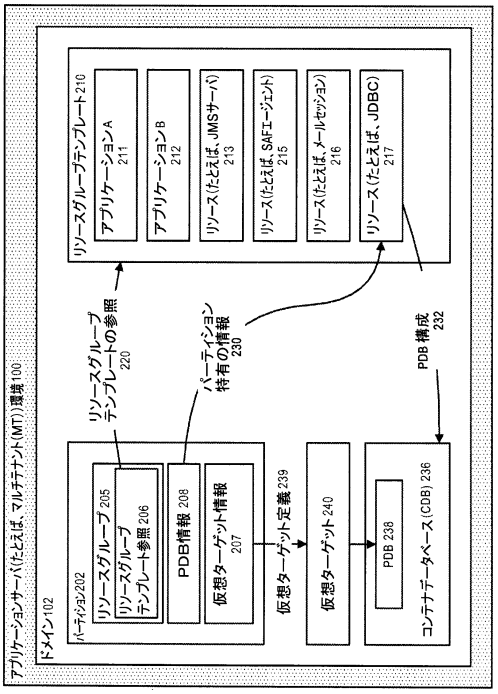

FIGURE 2

【図３】

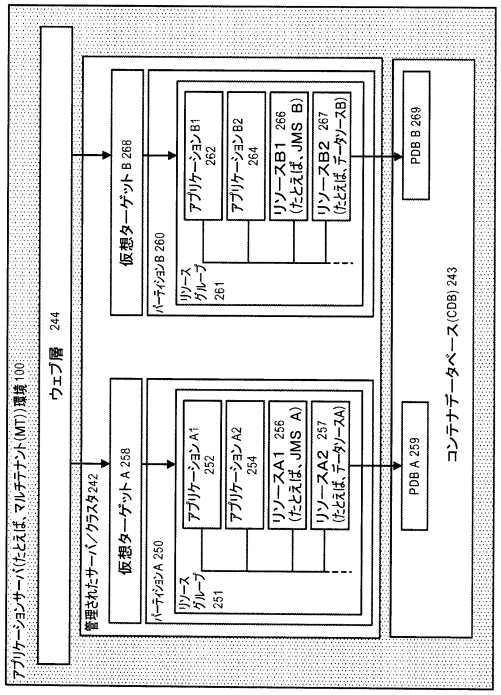

FIGURE 3

【図４】

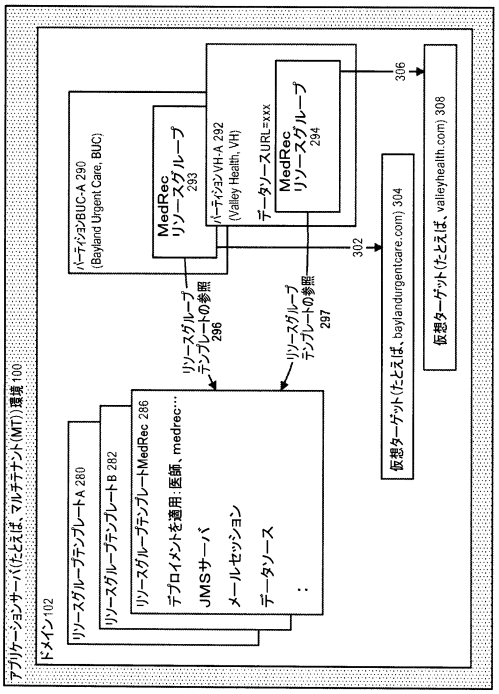

FIGURE 4

【図５】

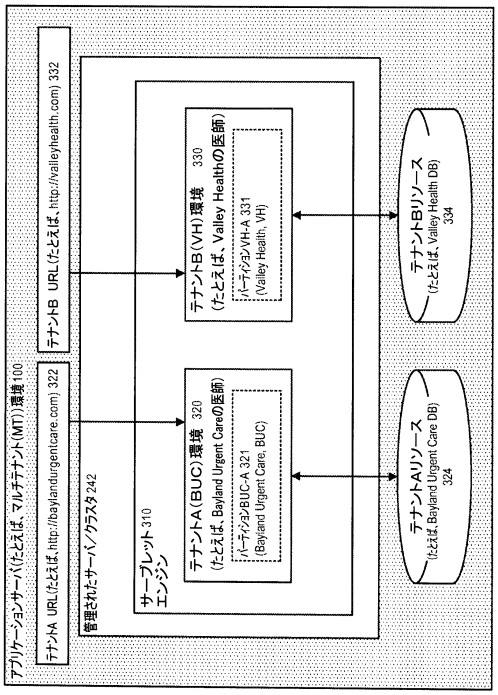

FIGURE 5

【図６】

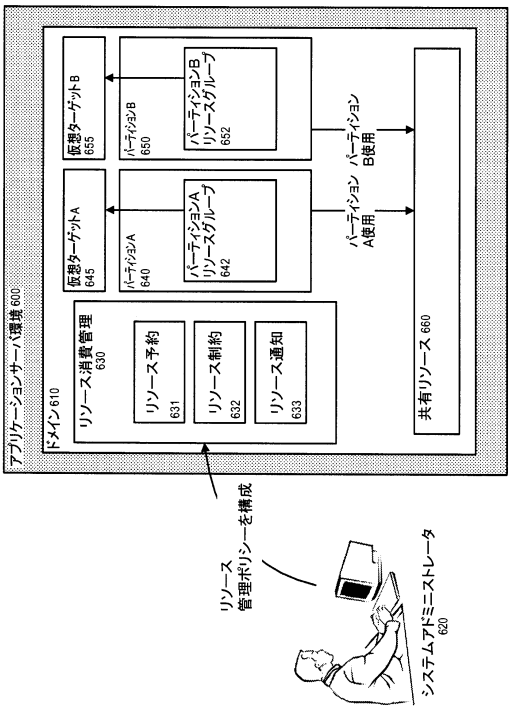

FIGURE 6

【図７】

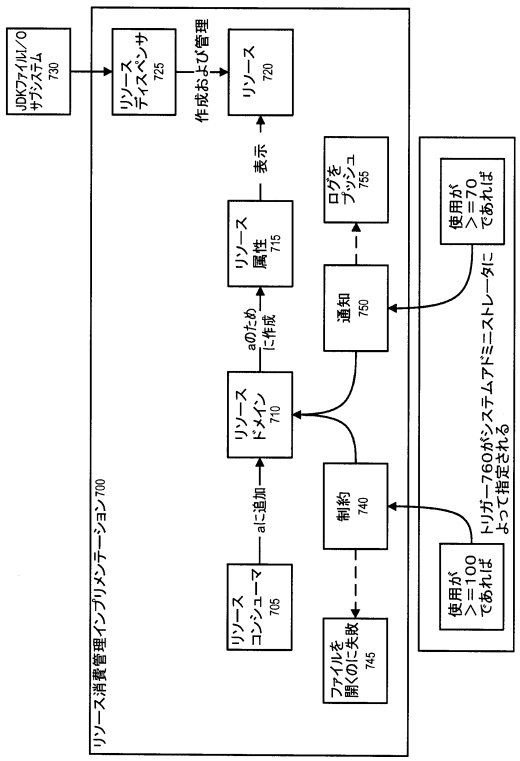

FIGURE 7

【図８】

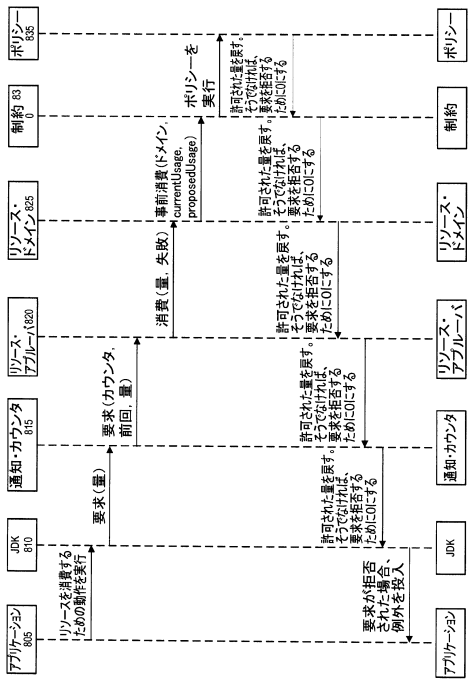

FIGURE 8

【図９】

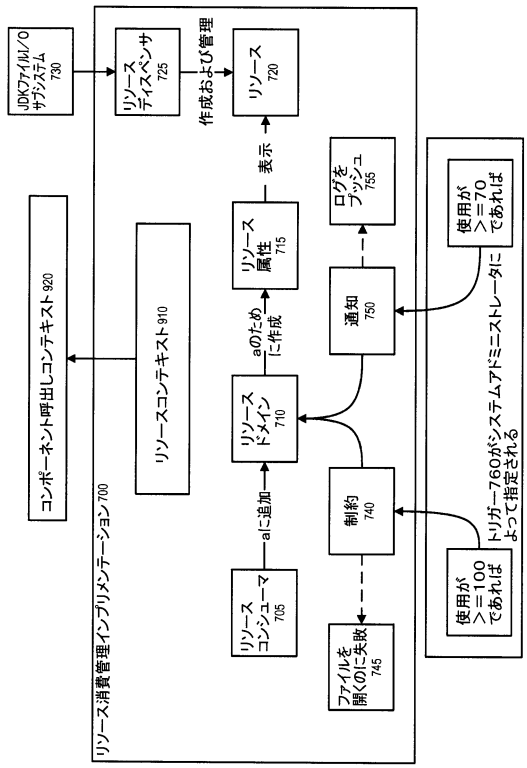

FIGURE 9

【図１０】

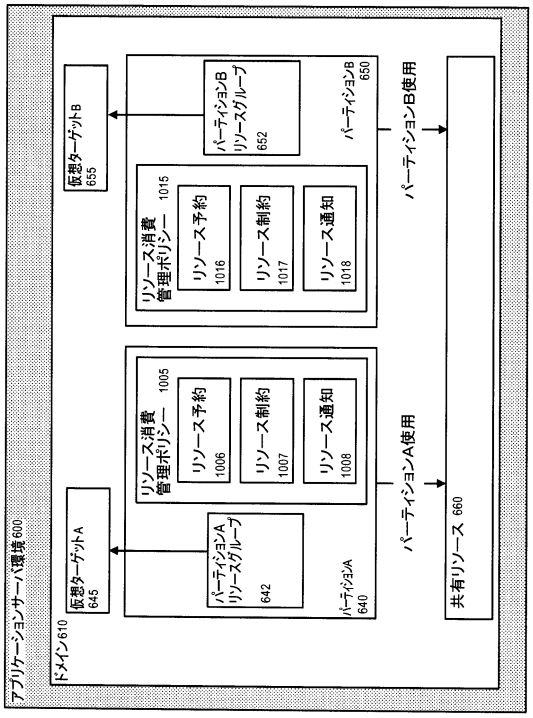

FIGURE 10

【図１１】

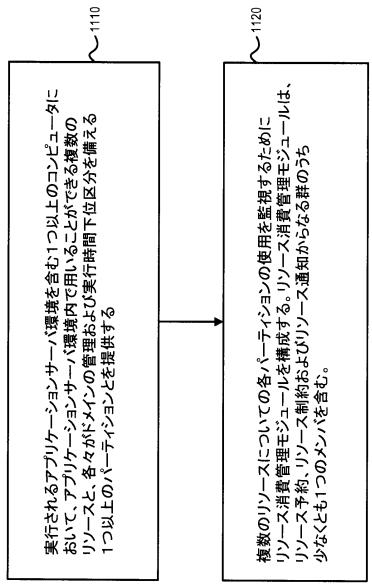

FIGURE 11

フロントページの続き

(31)優先権主張番号 14/795,427
(32)優先日 平成27年7月9日(2015.7.9)
(33)優先権主張国・地域又は機関
米国(US)

(72)発明者 ラム, ジャガディシュ
インド、５６００３７ カルナータカ、バンガロール、チナパナハリ、ネクスト・トゥ・エイ・イー・シィ・エス・レイアウト、ロハン・ミヒラ・アパートメンツ、ビィ－１０３
(72)発明者 サクセナ, クシティズ
インド、５６００３７ カルナータカ、バンガロール、ブルックフィールズ、エイ・イー・シィ・エス・レイアウト、シュリラム・スプルシ・アパートメンツ、シィ－００８
(72)発明者 スリバスタバ, ラウル
インド、５６００２９ カルナータカ、バンガロール、チッカ・アダゴディ、セカンド・クロス・ロード、ナンバー・１８、プレスティージ・セント・ジョンズ・ウッズ、プレスティージ・レキシントン
(72)発明者 フェイゲン, ローレンス
アメリカ合衆国、０７０６９ ニュー・ジャージー州、ウォッチャング、レッドモント・ロード、１０４
(72)発明者 メータ, ナマン
インド、５６００２９ カルナータカ、バンガロール、チッカ・アダゴディ、セカンド・クロス・ロード、ナンバー・１８、プレスティージ・セント・ジョンズ・ウッズ、プレスティージ・レキシントン
(72)発明者 スブラマニアン, プラサド
インド、５６００２９ カルナータカ、バンガロール、チッカ・アダゴディ、セカンド・クロス・ロード、ナンバー・１８、プレスティージ・セント・ジョンズ・ウッズ、プレスティージ・レキシントン

審査官 井上 宏一

(56)参考文献 特開２００３－２２３３３５（ＪＰ，Ａ）
特開２００９－１７６０９７（ＪＰ，Ａ）
特開２００１－１３４４５２（ＪＰ，Ａ）

(58)調査した分野(Int.Cl., ＤＢ名)
Ｇ０６Ｆ ９／４５５－９／５４