(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号
特許第7148570号
(P7148570)

(45)発行日　令和4年10月5日(2022.10.5)

(24)登録日　令和4年9月27日(2022.9.27)

(51)国際特許分類

		FI		
G06F	*9/46 (2006.01)*	G06F	9/46	430
G06F	*9/48 (2006.01)*	G06F	9/48	300Z

請求項の数　11　外国語出願　（全15頁）

(21)出願番号　特願2020-117139(P2020-117139)
(22)出願日　令和2年7月7日(2020.7.7)
(62)分割の表示　特願2017-552147(P2017-552147)の分割
原出願日　平成27年10月23日(2015.10.23)
(65)公開番号　特開2020-184365(P2020-184365A)
(43)公開日　令和2年11月12日(2020.11.12)
審査請求日　令和2年8月4日(2020.8.4)

(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弁理士法人深見特許事務所
(72)発明者　ルブ，シャンチェン
中華人民共和国、１００１９３ ベイジン、ハイディアン・ディストリクト、ヂョングアンツン・ソフトウェア・パーク、ビルディング・ナンバー・２４、オラクル・ビルディング
(72)発明者　ジン，ジム・ヨンシュン
中華人民共和国、１００１９３ ベイジ
最終頁に続く

(54)【発明の名称】　アプリケーションサーバを並列起動するためのシステムおよび方法

(57)【特許請求の範囲】
【請求項１】
トランザクション処理環境において、アプリケーションサーバを並列起動するための方法であって、
複数のサーバエントリおよび複数のグループエントリをメモリにロードするステップを含み、前記複数のサーバエントリの各々は、複数のサーバのうち１つのサーバに関連し、前記複数のグループエントリの各々は、複数のサーバグループのうち１つのサーバグループに関連しており、
前記複数のサーバエントリと前記複数のグループエントリとの間の依存関係マップを作成するステップと、
前記複数のサーバの中に依存関係を有しないサーバをチェックするステップと、
前記依存関係を有しないサーバを含む実行可能リストを作成するステップと、
前記実行可能リストに基づいて、前記複数のサーバの中に依存関係を有しないサーバを各々並列起動するステップとを含み、
前記複数のサーバの中に依存関係を有しないサーバを各々並列起動するステップは、
共有メモリ（ＳＨＭ）モードで前記トランザクション処理環境を実行する間、ブートプロセスモジュールが、前記複数のサーバ内の依存関係を持たないサーバの各々を並列に起動するステップと、
前記ブートプロセスモジュールが、前記複数のサーバ内の依存関係を持たないサーバの各々と通信するためのパイプを生成するステップと、

前記複数のサーバ内の依存関係を持たないサーバの各々が、前記パイプを介して前記ブートプロセスモジュールに応答を送信するステップとを含み、
前記方法はさらに、
前記複数のサーバの中に依存関係を有しないサーバを各々初期化するステップと、
起動したサーバの各々からの応答情報を受信するステップと、
前記応答情報を１つ受信するごとに、依存関係のあるサーバが起動可能か否かを判定するステップと、
前記依存関係のあるサーバが起動可能であることに基づいて、前記依存関係のあるサーバを前記実行可能リストに含めるステップと、
前記依存関係マップおよび前記実行可能リストに従って、残りの未起動サーバを起動するステップとを含む、方法。
【請求項２】
前記依存関係マップを作成するステップは、
前記複数のサーバエントリおよび前記複数のグループエントリの各々の依存関係属性をチェックするステップと、
前記チェック中に発見した各々の依存関係属性に基づいて、前記依存関係マップを作成するステップとを含む、請求項１に記載の方法。
【請求項３】
前記依存関係マップの依存関係ループをチェックするステップと、
依存関係ループを発見した場合、前記発見した依存関係ループを回避するように、前記依存関係マップを再作成するステップとをさらに含む、請求項１または２に記載の方法。
【請求項４】
前記作成された依存関係マップに従って残りの未起動サーバを起動する前に、並列起動された前記複数のサーバの中に依存関係を有しないサーバの各々が起動されたかをチェックするステップをさらに含む、請求項１～３のいずれかに記載の方法。
【請求項５】
前記複数のサーバの各々は、Tuxedoアプリケーションサーバを含む、請求項１～４のいずれかに記載の方法。
【請求項６】
トランザクション処理環境において、アプリケーションサーバを並列起動するためのシステムであって、
１つ以上のマイクロプロセッサと、
前記１つ以上のマイクロプロセッサ上で動作する処理装置とを備え、前記処理装置は、動作すると、
複数のサーバエントリおよび複数のグループエントリをメモリにロードするステップを実行し、前記複数のサーバエントリの各々は、複数のサーバのうち１つのサーバに関連し、前記複数のグループエントリの各々は、複数のサーバグループのうち１つのサーバグループに関連しており、
前記複数のサーバエントリと前記複数のグループエントリとの間の依存関係マップを作成するステップと、
前記複数のサーバの中に依存関係を有しないサーバをチェックするステップと、
前記依存関係を有しないサーバを含む実行可能リストを作成するステップと、
前記実行可能リストに基づいて、前記複数のサーバの中に依存関係を有しないサーバを各々並列起動するステップとを実行し、
前記複数のサーバの中に依存関係を有しないサーバを各々並列起動するステップは、
共有メモリ（ＳＨＭ）モードで前記トランザクション処理環境を実行する間、ブートプロセスモジュールが、前記複数のサーバ内の依存関係を持たないサーバの各々を並列に起動するステップと、
前記ブートプロセスモジュールが、前記複数のサーバ内の依存関係を持たないサーバの各々と通信するためのパイプを生成するステップと、

前記複数のサーバ内の依存関係を持たないサーバの各々が、前記パイプを介して前記ブートプロセスモジュールに応答を送信するステップとを含み、
前記複数のサーバの中に依存関係を有しないサーバを各々初期化するステップと、
起動したサーバの各々からの応答情報を受信するステップと、
前記応答情報を１つ受信するごとに、依存関係のあるサーバが起動可能か否かを判定するステップと、
前記依存関係のあるサーバが起動可能であることに基づいて、前記依存関係のあるサーバを前記実行可能リストに含めるステップと、
前記作成された依存関係マップに従って、残りの未起動サーバを起動するステップとをさらに実行する、システム。

【請求項７】
前記依存関係マップを作成するステップは、
前記複数のサーバエントリおよび前記複数のグループエントリの各々の依存関係属性をチェックするステップと、
前記チェック中に発見した各々の依存関係属性に基づいて、前記依存関係マップを作成するステップとを含む、請求項６に記載のシステム。

【請求項８】
前記依存関係マップの依存関係ループをチェックするステップと、
依存関係ループを発見した場合、前記発見した依存関係ループを回避するように、前記依存関係マップを再作成するステップとをさらに含む、請求項６または７に記載のシステム。

【請求項９】
前記作成された依存関係マップに従って残りの未起動サーバを起動する前に、並列起動された前記複数のサーバの中に依存関係を有しないサーバの各々が起動されたかをチェックするステップをさらに含む、請求項６～８のいずれかに記載のシステム。

【請求項１０】
前記複数のサーバの各々は、Tuxedoアプリケーションサーバを含む、請求項６～９のいずれかに記載のシステム。

【請求項１１】
１または複数のプロセッサに請求項１～５のいずれかに記載の方法を実行させるためのプログラム。

【発明の詳細な説明】
【技術分野】
【０００１】
著作権表示
この特許文書の開示の一部は、著作権保護の対象となるものを含んでいる。著作権者は、特許商標庁の特許ファイルまたは記録に掲載された特許文書または特許開示の複製に対しては異議を唱えないが、その他の場合、全ての著作権を留保する。

【０００２】
発明の分野
本発明は、一般にコンピュータシステムに関し、特にトランザクション処理環境に関する。

【背景技術】
【０００３】
背景
Tuxedoアプリケーションを起動（boot）するための従来のメカニズム（すなわち、tmboot）は、通常、所定の順序でTuxedoアプリケーションサーバを１つずつ起動する。大規模／複雑なアプリケーションの場合、起動時間が長くなり、望ましくない。

【発明の概要】
【０００４】

概要

アプリケーションサーバを並列起動するためのシステムおよび方法が提供される。例示的な方法において、各サーバおよびサーバグループエントリに依存関係属性を関連付けることができる。この方法は、依存関係属性に基づいて、依存関係マップを作成することができる。この方法は、呼び出されると、依存関係を有しないサーバを各々並列起動することができる。依存関係マップに基づいて、残りのサーバおよびサーバグループを起動することができる。

【発明が解決しようとする課題】

【０００５】

従来、Tuxedoによって、シーケンスパラメータに従ってアプリケーションサーバを起動することができる。この場合、管理者が（アプリケーションを使用するようにサーバを起動するために）tmbootを呼び出したときに、アプリケーションサーバは、割り当てられたシーケンス番号／パラメータに従って起動される。これによって、サーバの起動シーケンスが遅くなる可能性がある。

【課題を解決するための手段】

【０００６】

本開示に記載のシステムおよび方法は、グループおよびサーバにDEPENDSON属性を追加
する。アプリケーションサーバは、DEPENDSONパラメータで指定された順序で起動される
。これによって、アプリケーションサーバを並列起動することができ、より迅速且つより効率的に起動プロセスを実行することができる。

【図面の簡単な説明】

【０００７】

【図１】一実施形態に従って、トランザクション処理環境を例示する図である。

【図２】一実施形態に従って、アプリケーションサーバの並列起動を例示する図である。

【図３】トランザクション処理環境においてアプリケーションサーバを並列起動するための例示的な方法を示すフローチャートである。

【発明を実施するための形態】

【０００８】

詳細な説明

本発明は、限定ではなく例示として添付の図面に示される。添付図面の図において、同様の参照符号が類似する要素を示す。なお、本開示において、一実施形態、１つの実施形態またはいくつかの実施形態の言及は、必ずしも同一の実施形態に限定されず、少なくとも１つの実施形態を意味する。

【０００９】

特定の実現例を説明する場合、これらの特定の実現例は、例示のみの目的で提供されていることを理解すべきである。当業者なら、本発明の範囲および精神から逸脱することなく、他の要素および構成を使用できることを理解するであろう。

【００１０】

以下の本発明の説明は、トランザクションミドルウェアマシン環境の例として、Tuxedo（登録商標）環境を使用する。制限することなく、他の種類のトランザクションミドルウェアマシン環境を使用できることは、当業者にとって明らかであろう。

【００１１】

場合によって、本発明の完全な説明を提供するように、多くの具体的な詳細を記載する。しかしながら、これらの具体的な詳細がなくても、本発明を実施できることは、当業者にとって明らかであろう。また、場合によって、本発明を不明瞭にしないように、周知の特徴は、詳細に記載されない。例えば、詳細な説明において、例示として、ＸＡ分散トランザクション環境を使用する。本発明は、制限することなく、他の種類の分散トランザクション環境に適用できることは、当業者にとって明らかであろう。

【００１２】
図面および詳細な説明において、共通の参照番号を使用して同様の要素を示す。したがって、他の箇所で図示された要素を説明する場合、図面に使用されている参照番号は、必ずしもこの図面に対応する詳細な説明に記載する必要がない。
【００１３】
本開示は、トランザクション処理環境において、アプリケーションサーバを並列起動するためのシステムおよび方法を説明する。
【００１４】
トランザクションミドルウェア環境
トランザクションミドルウェアシステムは、迅速に用意することができ且つオンデマンドで拡張することができる大規模並列処理インメモリグリッドを含む完全なＪａｖａ（登録商標）ＥＥアプリケーションサーバ複合体を提供するために、６４ビットプロセッサ技術などの高性能ハードウェア、高性能大容量のメモリ、冗長インフィニバンド（登録商標）およびイーサネット（登録商標）ネットワーク、並びにWebLogic（登録商標）スイートなどのアプリケーションサーバまたはミドルウェア環境の組み合わせを含む。本発明のシステムは、アプリケーションサーバグリッド、ストレージエリアネットワーク、およびインフィニバンド（ＩＢ）ネットワークを形成するフルラック、ハーフラックまたはクォーターラックもしくは他の構成として展開されることができる。ミドルウェアマシンソフトウェアは、アプリケーションサーバ、ミドルウェアおよび他の機能、例えば、WebLogic Server、JRockitまたはHotspot JVM、Oracle（登録商標）Linux（登録商標）あまたはSolaris、およびOracle（登録商標）VMを提供することができる。本発明のシステムは、ＩＢネットワークを介して互いに通信する複数の計算ノード、ＩＢスイッチゲートウェイ、およびストレージノードまたはストレージユニットを含むことができる。ラック構成として実装される場合、ラックの未使用部分は、空のままにしてもよく、詰め物で充填されてもよい。
【００１５】
例えば、「Sun Oracle Exalogic」または「Exalogic」などのシステムにおいて、本発明のシステムは、Oracle（登録商標）ミドルウェアＳＷスイートまたはWebLogicなどのミドルウェアまたはアプリケーションサーバソフトウェアをホストするために、展開容易な解決案である。本開示に説明するように、トランザクションミドルウェアシステムは、ミドルウェアアプリケーションをホストするために必要とされる１つ以上のサーバ、ストレージユニット、ストレージネットワーク用のＩＢファブリック、および全ての他の要素を含む「グリッドボックス」（grid in a box）である。例えば、共有キャッシュアーキテクチャを備えるクラスタデータベースであり且つクラウドアーキテクチャの構成要素であり得るOracleリアルアプリケーションクラスタ（ＲＡＣ）エンタープライズデータベースおよびExalogicオープンストレージを使用する大規模並列グリッドアーキテクチャを活用することによって、全ての種類のミドルウェアアプリケーションに高性能を提供することができる。このシステムは、線形Ｉ／Ｏ拡張性によって改善した性能を提供し、使用および管理が簡単であり、ミッションクリティカルな可用性および信頼性を提供する。
【００１６】
本発明の一実施形態によれば、Tuxedoシステムのようなトランザクションミドルウェアシステムは、複数のプロセッサを備える高速マシン、例えばExalogicミドルウェアマシン、および高性能ネットワーク接続、例えばインフィニバンド（ＩＢ）ネットワークを利用することができる。
【００１７】
Oracle（登録商標）Tuxedoシステムは、高性能分散型ビジネスアプリケーションの構築、実行および管理を可能にするソフトウェアモジュールのセットであり、トランザクションミドルウェアとして多くの複層アプリケーション開発ツールに使用されている。Tuxedoは、分散トランザクション処理を管理するために、分散コンピューティング環境に使用

することができるミドルウェアプラットフォームである。Tuxedoは、無制限の拡張性および標準ベースの相互運用性を提供しながら、エンタープライズレガシーアプリケーションのロックを解除し、エンタープライズレガシーアプリケーションをサービス指向型アーキテクチャに拡張するための有効なプラットフォームである。

【００１８】

また、Tuxedoは、リクエスト、イベント、システムプロセス間のアプリケーションキュー、およびアプリケーションサービスを効率的にルーティングする、配布するおよび管理するためのサービス指向型インフラストラクチャを提供する。Tuxedoは、事実上無制限に拡張できるため、ピークトランザクションボリュームを効率的に管理することによって、ビジネスの機敏性を向上させ、ＩＴ組織がビジネスの需要および処理量の変化に迅速に対応できるようにする。Oracle（登録商標）Tuxedoは、アクセスプロトコルに関係なく、複数のデータベース間のトランザクションを最適化し、全ての加入リソースの間のデータ整合性を保証する。このシステムは、トランザクション加入者を追跡し、拡張コミットプロトコルを監督することによって、全てのトランザクションコミットおよびロールバックを適切に処理することを保証する。

【００１９】

さらに、Oracle（登録商標）Tuxedoシステムは、２フェイスコミット（２ＰＣ）処理をサポートするためのＸＡ標準、X/Open（登録商標）ATMI API、X/Open（登録商標）「分散トランザクション処理：ＴＸ（トランザクション境界設定）仕様」、および言語国際化を行うためのX/Open（登録商標）移植性ガイド（ＸＰＧ）標準を含むオープングループのX/Open（登録商標）標準に準拠することができる。トランザクションアプリケーションサーバは、ＸＡ標準を使用する場合、ＸＡサーバとして呼ばれてもよい。例えば、Tuxedoグループに属する各Tuxedoアプリケーションサーバは、OPENINFO属性を使用するように構成することができる。Tuxedoグループ内の全てのＸＡサーバは、OPENINFO属性を使用して、リソースマネージャ（ＲＭ）との接続を形成することができる。

【００２０】

Tuxedoアプリケーションサーバの順次起動

従来、Tuxedoによって、シーケンスパラメータに従ってアプリケーションサーバを起動することができる。この場合、管理者が（アプリケーションを使用するようにサーバを起動するために）tmbootを呼び出すときに、アプリケーションサーバは、割り当てられたシーケンス番号／パラメータに従って起動される。

【００２１】

アプリケーションサーバは、シーケンスパラメータ（SEQUENCEパラメータ）によって指定された順序でまたは設定ファイル（例えば、UBBCONFIG）内のサーバエントリの順序で
起動することができる。２つ以上のサーバが同様のSEQUENCEパラメータを有する場合、tmbootは、これらのサーバを並行起動して、全てののサーバの初期化が完了するまで操作を続行しない。

【００２２】

あるサーバが起動できなかった場合、tmbootは、診断結果を主要イベントログに書き込み、起動できなかったサーバが親サーバである（すなわち、起動できなかったサーバの後に起動されるサーバが正常に動作するためには起動できなかったサーバに依存する）場合を除いて、操作を続行することができる。

【００２３】

このように、起動および／または初期化できなかったサーバが起動プロセス内の他のサーバを起動するための前提条件である場合、またはサーバを確認できなかった場合、アプリケーションサーバの起動プロセスに望ましくない遅延が生じる可能性がある。

【００２４】

依存関係に基づくTuxedoアプリケーションサーバの並列起動

図１は、トランザクション処理環境１００を例示する図である。より具体的には、図１

は、トランザクション処理環境１００内のTuxedoアプリケーションサーバを並列起動するためのプロセスを示している。

【００２５】

一実施形態によれば、トランザクション処理環境１００内で、管理者１１０は、tmbootなどのコマンドおよび起動マップ１１５を用いて、１つまたは複数のアプリケーションサーバ１３０、例えばTuxedoアプリケーションサーバを起動することができる。アプリケーションサーバは、起動され初期化された後、１つ以上のデータベース１４０と通信することができ、アプリケーション１２０に利用されることができる。

【００２６】

一実施形態によれば、起動マップ１１５は、２つ以上のアプリケーションサーバ１３０を並行起動させる起動プランを含むことができ、サーバとサーバグループとの間の依存関係を取り込むことができる。

【００２７】

一実施形態によれば、Tuxedoアプリケーションサーバを並列起動するためのオプションをtmbootに追加することができる。このオプションは、アプリケーションサーバを並列モードで起動するよう指定するために、tmbootに追加された「-p num」オプションであってもよい。また、tmbootコマンドを使用するときに、Tuxedoアプリケーションサーバの起動順序を指定するように、新しいキーワード「DEPENDSON」をUBBCONFIG内のグループおよびサーバに追加することもできる。

【００２８】

一実施形態によれば、tmbootに追加された「-p num」オプションは、アプリケーションサーバを並列モードで起動するように指定することができる。「-p」は、アプリケーションサーバを並列起動することを意味する。「num」は、同時に起動できるサーバの数を意
味する。例えば、「-p 1」を指定したtmbootは、「-p」オプションを指定しないtmbootと同様である。num=0の場合、システムは、初期の数字、例えば１６を使用することができ
る。

【００２９】

一実施形態によれば、本開示に記載の方法およびシステムは、キーワードDEPENDSON属
性をグループおよびサーバに追加することができる。アプリケーションサーバは、DEPENDSONパラメータまたはSEQUENCEパラメータにより指定された順序でまたは設定ファイル（
例えば、UBBCONFIG(5)）内のサーバエントリの順序で起動することができる。DEPENDSON
パラメータおよびSEQUENCEパラメータが同時に指定された場合、DEPENDSONパラメータが
優先する。

【００３０】

一実施形態によれば、本開示に記載の方法およびシステムは、tmbootにオプション「-p〈数字〉」を追加することができる。UBBCONFIGに設定されたDEPENDSON関係を有しない全
てのアプリケーションサーバは、並列して起動する。依存関係に従った順序でサーバを起動し且つ並列に動作するために、tmbootは、各サーバが起動した状態まで待機しなければならない。

【００３１】

一実施形態によれば、SHM（共有メモリ）モードで動作する場合、tmbootは、並列に動作するサーバを全て起動し、起動されたサーバからの応答を待つ。これらのサーバは、パイプを介して応答をtmbootプロセスに送信し、tmbootプロセスは、応答を受け取り、次

のジョブを実行する。
【００３２】
一実施形態によれば、MP（マルチプロセス）モードで動作する場合、tmbootおよびtlistenは、互いに情報を交換するもう１つのスレッドを作成することができる。このローカルスレッドは、リモートマシンから送信されるメッセージを待つことができる。リモートで起動されたサーバがある場合に、続行するようにメインスレッドに通知する。リモートtlistenスレッドは、起動されたサーバからの応答を収集するように準備を整える。１つのサーバが起動または失敗した場合、ネットワークメッセージを介してローカルスレッドに通知する。
【００３３】
図２は、一実施形態に従って、アプリケーションサーバの並列起動を例示する図である。
【００３４】
一実施形態によれば、図２は、いくつかのサーバ、例えば、サーバＳ１１（２０２）、Ｓ１２（２０９）、Ｓ１３（２０４）、Ｓ１４（２０３）、Ｓ２１（２０６）、Ｓ２２（２１１）、Ｓ２３（２１０）、Ｓ２４（２０５）の依存関係および並列起動マップを示している。また、依存関係および並列起動マップは、Ｇ１（２１２）、Ｇ２（２１３）、Ｇ３（２１６）およびＧ４（２１８）を含むいくつかのサーバグループを含む。
【００３５】
一実施形態によれば、図２の依存関係および並列起動マップ２００は、サーバとグループとの間の依存関係に基づいて構築することができる。
【００３６】
一実施形態によれば、図２に示すように、サーバグループの数は複数であってもよい。すなわち、サーバグループは、Ｇ１、Ｇ２およびＧ３（３つ）を含んでもよい。
【００３７】
Ｇ１は、サーバＳ１１、Ｓ１２（DEPENDSON S11）、Ｓ１３およびＳ１４を含む。表記
（DEPENDSON）を用いて、Ｓ１２が正常に機能するように起動され且つ初期化されたＳ１
２に依存することを示すことができる。
【００３８】
Ｇ２は、サーバＳ２１、Ｓ２２（DEPENDSON S21）、Ｓ２３（DEPENDSON G1.S13、S24）およびＳ２４を含む。ここでは、表記（DEPENDSON）を用いて、Ｓ２２が正常に機能する
ように起動され且つ初期化されたＳ２１に依存することを示すことができる。同様に、Ｓ３２は、正常に機能するように起動され且つ初期化された（Ｇ１の）Ｓ１３およびＳ２４に依存する。
【００３９】
Ｇ３（DEPENDSON G1、G2）は、サーバＳ３１およびＳ３２を含むことができる。ここでは、Ｇ３が正常に機能するために、グループＧ１およびＧ２を完全に起動して初期化する必要がある。
【００４０】
Ｇ４は、Ｓ４１、Ｓ４２（DEPENDSON S41, G3）、およびＳ３を含むことができる。ここ
ここでは、表記（DEPENDSON）を用いて、Ｇ４が正常に機能するように起動され且つ初期化
されたＳ４１およびＧ３を示すことができる。
【００４１】
一実施形態によれば、Ｓ１１、Ｓ１３、Ｓ１４、Ｓ２１、Ｓ２４、Ｓ４１、Ｓ４３を並行起動するように、図２の並列起動マップ２００を構築し、サーバを起動する（２０１）ことができる。Ｓ１１を起動した後、Ｓ１２を起動することができる。Ｓ２１を起動した

後、Ｓ２２を起動することができる。Ｓ１３を起動した後、Ｓ２４を起動し、その後Ｓ２３を起動することができる。Ｇ１（Ｓ１１、Ｓ１２、Ｓ１３、Ｓ１４）およびＧ２（Ｓ２１、Ｓ２２、Ｓ２３、Ｓ２４）を起動した後、サーバＳ３１、Ｓ３２を並列起動することができる。Ｇ３（Ｓ３１、Ｓ３２）およびＳ４１を起動した後、Ｓ４２を起動することができる。Ｓ４２およびＳ４３を起動した後、Ｇ４を起動し、初期化する。

【００４２】

一実施形態によれば、全てのサーバを起動した後、管理者によって開始されたtmbootタスクを完了することができる。

【００４３】

図３は、トランザクション処理環境においてアプリケーションサーバを並列起動するための例示的な方法を示すフローチャートである。方法は、ステップ３１０から始まる。ステップ３１０において、方法は、複数のサーバエントリおよび複数のグループエントリをメモリにロードすることができる。複数のサーバエントリの各々は、複数のサーバのうち１つのサーバに関連し、複数のグループエントリの各々は、複数のサーバグループのうち１つのサーバグループに関連する。ステップ３２０において、方法は、複数のサーバと複数のグループエントリとの間の依存関係マップを作成することができる。ステップ３３０において、方法は、引き続き、複数のサーバの中に依存関係を有しないサーバをチェックすることができる。ステップ３４０において、方法は、複数のサーバの中に依存関係を有しないサーバを各々並列起動することができる。ステップ３５０において、方法は、引き続き、複数のサーバの中に依存関係を有しないサーバを各々初期化することができる。ステップ３６０において、方法は、作成された依存関係マップに従って、残りの未起動サーバを起動することができる。

【００４４】

起動プロセス

一実施形態によれば、tmbootが実行されると、例示的なプロセスは、サーバおよびグループエントリをTUXCONFIGからメモリにロードすることができる。次に、プロセスは、グ
ループおよびサーバのDEPENDSON属性に従って、依存関係マップ内のサーバノードとグル
ープノードとの間の依存関係を設定することができる。次に、プロセスは、ループおよび他のDEPENDSONエラーが存在するか否かをチェックすることができる。次に、プロセスは
、map_startノードからサーバを実行可能なリストに追加することができる。次に、プロセスは、非同期モードで利用可能なリスト内のサーバを起動し、パイプから応答情報を取得する。プロセスは、１つの応答を受け取ると、従属サーバを起動できるか否かをチェックする。起動できる場合、従属サーバを実行可能なリストに追加し、実行可能なリスト内のプロセスの起動を続行する。最後に、全てのサーバを起動した後、tmbootが終了する。

【００４５】

依存関係マップの作成

一実施形態によれば、例示的なプロセスは、例えば以下の例示的な手順に従って、依存関係マップを作成することができる。

【００４６】

【数１】

グループ名およびサーバ名に従って、サーバ／グループアレイのエントリをソートする。
For サーバ／グループアレイの各エントリに対して
　最後のエントリに従って、同名のサーバをスキップする。
　　If DEPENDSON リストを有する場合
　　　DEPENDSON リストを解析する
　　　For DEPENDSON リスト内の各 DEPENDSON 項目に対して
　　　　サーバ／グループアレイ内の項目を検索する
　　　　サーバと DEPENDSON 項目との間の依存関係を設定する
　　　　If DEPENDSON 項目がグループである場合、サーバをグループ内の全てのサーバに依存するように設定する
　　　End For
　　End If
　　同名サーバの数および最初の同名サーバを設定する
End For
設定ファイルから依存関係を読み取り、設定する
For サーバ／グループアレイの各エントリに対して
　If エントリがサーバである場合、サーバグループをグループ内の全てのサーバに依存するように設定する
End For

【００４７】
【数２】

For サーバ／グループアレイの各エントリに対して
　　If サーバが何も依存していない場合、map_start に依存するように設定する
　　If サーバが何にも依存されていない場合、map_end に依存するように設定する
End For
依存関係ループをチェックする

【００４８】
ローカルモードでの並列起動
一実施形態によれば、プロセスは、ローカルモードで並列起動をサポートすることができる。以下、このようなプロセスの一例を示す。
【００４９】
【数３】

「-p」オプションを指定した tmboot
依存関係マップを作成する
実行可能なリストを作成する
proc 応答を処理する
For 起動される各サーバに対して
起動されるサーバがサーバ依存サーバであるか否かをチェックし、サーバ依存サーバではない場合、警告を示す
処理を開始する（応答を待たず、sync_proc を呼び出すことによって、ローカルマシンまたはリモートマシンにおいて処理を開始する）
応答待ちリストを追加する
処理応答を処理する
End For

【００５０】
Unix（登録商標）の場合
tmbootプロセスは、匿名パイプを作成し、サーバプロセスを分岐し、パイプｆｄをサーバプロセスに送信する。
【００５１】
サーバプロセスは、tpsrvinitを完了し、パイプを介して応答をtmbootに送信する。
tmbootは、非停止（non-block）モードでパイプを読み込み、応答を取得した場合、依存関係マップをチェックして、次に起動するサーバを判断する。
【００５２】
Windows（登録商標）の場合
tmbootプロセスは、名前付きパイプを作成し、サーバプロセスを分岐する。サーバは、tpsrvinitを完了した後、名前付きパイプを開放して接続し、名前付きパイプを介して応答をtmbootに送信する。
【００５３】
tmbootは、１つのスレッドを作成し、パイプ情報を待つ。応答を取得した場合、依存関係マップをチェックして、次に起動するサーバを決定する。
【００５４】
SHMモード
一実施形態によれば、SHMモードで並列起動をサポートすることができる。SHMモードにおいて、tmbootは実行可能なリスト内の全てのサーバを起動し、起動されたサーバからの応答を待つ。サーバは、パイプを介して応答をtmbootプロセスに送信する。tmbootプロセスは、応答を受け取り、次のジョブを実行する。Tmbootは、非同期モードで実行可能なリスト内のサーバを起動し、起動されたサーバからの応答を収集するように準備を整える。サーバが１つの応答を受信した場合、依存先のサーバの準備が整っているか否かをチェックする。準備が整っている場合、そのサーバを実行可能なリストに追加して起動させる。
【００５５】
MPモードでの並列起動
一実施形態によれば、MPモードで並列起動をサポートすることができる。MPモードにおいて、tmbootおよびtlistenは、互いに情報を交換するもう１つのスレッドを作成できる。このローカルスレッドは、リモートマシンから送信されるメッセージを待つことができる。リモートで起動されたサーバがある場合に、続行するようにメインスレッドに通知する。リモートtlistenスレッドは、起動されたサーバからの応答を収集するように準備を整える。１つのサーバが起動または失敗した場合、ネットワークメッセージを介してロー

カルスレッドに通知する。

【００５６】

本発明の多くの特徴は、ハードウェア、ソフトウェア、ファームウェア、またはそれらの組み合わせに実行する、またはそれらを使用して実行する、またはそれらの援助で実行することができる。したがって、（例えば、１つ以上のプロセッサを含む）処理システムを用いて、本発明の特徴を実施することができる。

【００５７】

本発明の多くの特徴は、コンピュータプログラム製品に実装され、またはこのコンピュータプログラム製品を使用して実装され、またはこのコンピュータプログラム製品の支援によって実装されることができる。このコンピュータプログラム製品は、本明細書に記載の特徴のいずれかを実行するように処理システムをプログラムするために使用できる命令を格納する記憶媒体またはコンピュータ可読媒体である。この記憶媒体は、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装置を含むことができるが、これらに限定されない。

【００５８】

本発明の特徴は、機械可読媒体のいずれかに格納された場合、処理システムのハードウェアを制御し、本発明の結果を利用して処理システムと他の機構との相互作用を可能にするソフトウェアおよび／またはファームウェアに組み込むことができる。そのようなソフトウェアまたはファームウェアは、アプリケーションコード、装置ドライバ、オペレーティングシステムおよび実行環境／コンテナを含むことができるが、これらに限定されない。

【００５９】

本発明の特徴は、例えば、特定用途向け集積回路（ＡＳＩＣ）などのハードウェア要素を使用して、ハードウェアで実施することもできる。本開示に記載の機能を実行するためのハードウェアステートマシンの実装は、当業者には明らかであろう。

【００６０】

さらに、本発明は、本開示の教示に従ってプログラムされた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可読記憶媒体を含む１つ以上の従来の汎用または専用デジタルコンピュータ、コンピューティング装置、機械、またはマイクロプロセッサを使用して都合よく実施することができる。ソフトウェア分野の当業者には明らかであるように、熟練したプログラマは、本開示の教示に基づいて、適切なソフトウェアのコーディングを容易に準備することができる。

【００６１】

上記で本発明のさまざまな実施形態を説明したが、これらの実施形態は、限定ではなく例示として提示されることを理解すべきである。本発明の精神および範囲から逸脱することなく、形態および詳細のさまざまな変更を行うことができることは、当業者にとって明らかであろう。

【００６２】

本発明は、特定の機能の実行およびそれらの関係を示す機能性構成ブロックを用いて説明された。これらの機能性構成ブロックの境界は、通常、説明の便宜上、本開示に任意に定義される。代替的には、別々の要素によって実行されるように示された機能は、１つの要素によって実行されてもよい。特定の機能およびそれらの関係が適切に実行される限り、代替的な境界を定義してもよい。したがって、これらの代替的な境界のいずれも、本発明の範囲および精神内に含まれる。

【００６３】

本発明の上記記載は、例示および説明を目的として提供される。本発明を網羅的なものまたは開示された形態そのものに限定することを意図していない。本発明の広さおよび範囲は、上記の例示的な実施形態のいずれかによって限定されるべきではない。数多くの変

更および変形は、当業者にとって明らかであろう。修正例および変形例は、開示された特徴の任意可能な組み合わせを含む。実施形態は、本発明の原理およびその実際の応用を最も良く説明することによって、他の当業者がさまざまな実施形態および考えられる特定の用途に適したさまざまな変形に応じて本発明を理解できるように、選択され説明される。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物によって規定されると意図する。

【図面】

【図１】

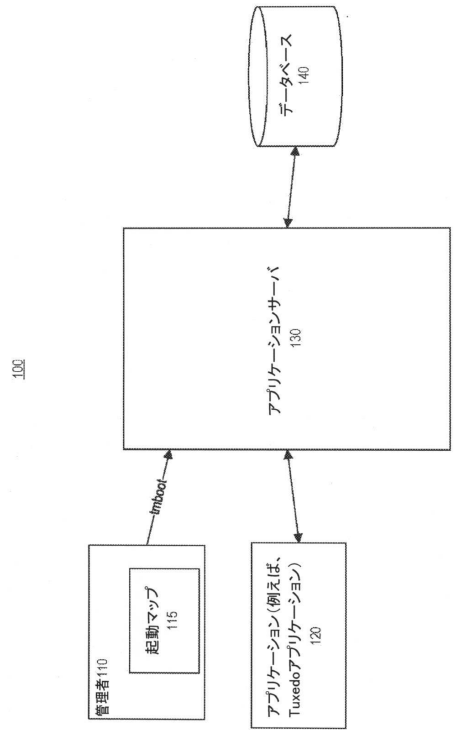

FIGURE 1

【図２】

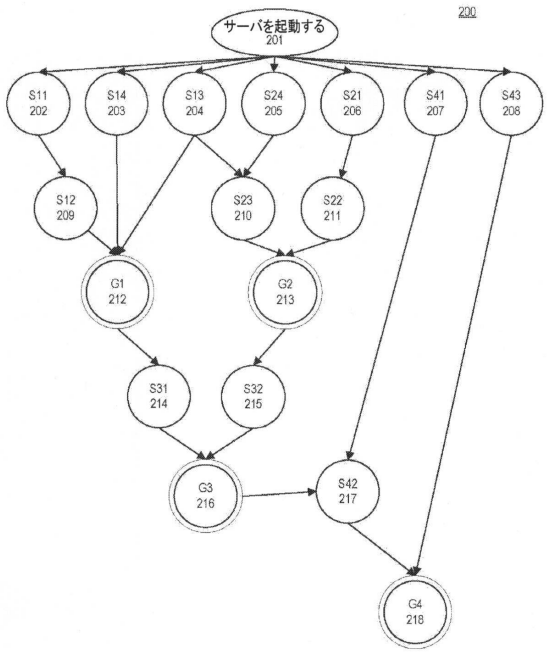

FIGURE 2

【図３】

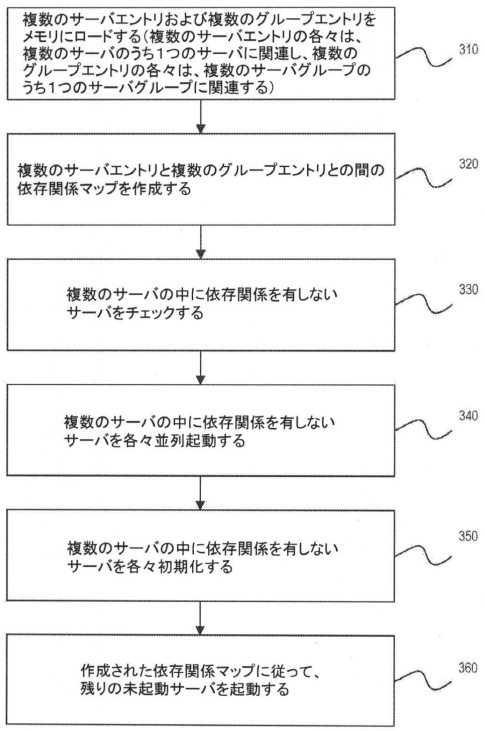
複数のサーバエントリおよび複数のグループエントリをメモリにロードする（複数のサーバエントリの各々は、複数のサーバのうち1つのサーバに関連し、複数のグループエントリの各々は、複数のサーバグループのうち1つのサーバグループに関連する）
310
複数のサーバエントリと複数のグループエントリとの間の依存関係マップを作成する
320
複数のサーバの中に依存関係を有しないサーバをチェックする
330
複数のサーバの中に依存関係を有しないサーバを各々並列起動する
340
複数のサーバの中に依存関係を有しないサーバを各々初期化する
350
作成された依存関係マップに従って、残りの未起動サーバを起動する
360

Figure 3

フロントページの続き

ン、ハイディアン・ディストリクト、ヂョングアンツン・ソフトウェア・パーク、ビルディング・ナンバー・２４、オラクル・ビルディング
(72)発明者　リトル，トッド・ジェイ
アメリカ合衆国、６００６７－６６７５　イリノイ州、パラタイン、ウエスト・イリノイ・アベニュ、１１５５
(72)発明者　リ，シャンドン
中華人民共和国、１００１９３　ベイジン、ハイディアン・ディストリクト、ヂョングアンツン・ソフトウェア・パーク、ビルディング・ナンバー・２４、オラクル・ビルディング

審査官　三坂 敏夫

(56)参考文献
特開２０１４－０１０７７２（ＪＰ，Ａ）
国際公開第２００６／０４３３２１（WO，Ａ１）
特開平０６－１４９７６５（ＪＰ，Ａ）
米国特許第０８８１９６７３（ＵＳ，Ｂ１）
特開平０７－３３４４６６（ＪＰ，Ａ）
特開２００７－１７９２５２（ＪＰ，Ａ）
Timothy E. Carone，ミドルウェアと３層クライアント／サーバーシステム開発，ドクター・ドブス・ジャーナル／日本版 1997年3月号，日本，株式会社翔泳社，1997年03月01日，第6巻，第33頁-第39頁
平 初＆できるシリーズ編集部，できるPRO Red Hat Enterprise Linux7，第1版，株式会社インプレス，2015年07月01日，第116頁-第121頁
TPシリーズ V1.1 運用手引書 B2WN-0022-01-00，富士通株式会社，1996年05月，第1頁 第26頁-第27頁 第120頁-第125頁

(58)調査した分野　(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　９／４６－４８
Ｇ０６Ｆ　９／４４５
Ｇ０６Ｆ　１１／３０