(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第5052955号
(P5052955)
(45)発行日　平成24年10月17日(2012.10.17)　(24)登録日　平成24年8月3日(2012.8.3)

(51)Int.Cl.　　F I
*G06F 9/48 (2006.01)*　　G06F 9/46 452J
*G06F 11/00 (2006.01)*　　G06F 9/06 630C

請求項の数 8　(全 44 頁)

(21)出願番号　特願2007-124311 (P2007-124311)
(22)出願日　平成19年5月9日(2007.5.9)
(65)公開番号　特開2008-282130 (P2008-282130A)
(43)公開日　平成20年11月20日(2008.11.20)
審査請求日　平成22年2月10日(2010.2.10)

前置審査

(73)特許権者　000005108
株式会社日立製作所
東京都千代田区丸の内一丁目６番６号
(74)代理人　100114236
弁理士　藤井　正弘
(74)代理人　100075513
弁理士　後藤　政喜
(72)発明者　川本　真一
東京都国分寺市東恋ヶ窪一丁目２８０番地
株式会社日立製作所　中央研究所内
(72)発明者　中村　友洋
東京都国分寺市東恋ヶ窪一丁目２８０番地
株式会社日立製作所　中央研究所内

最終頁に続く

(54)【発明の名称】アプリケーションの高可用運用方法、オンラインバージョン変更方法及び計算機システム

(57)【特許請求の範囲】
【請求項１】
処理要求を受け付ける第１のアプリケーションと第２のアプリケーションを入れ替えるアプリケーションの高可用制御方法であって、
予め設定したアプリケーションから識別子の異なる前記第１のアプリケーションと、前記第２のアプリケーションを生成する手順と、
前記複数のアプリケーションの実行に関する情報をセッション情報として生成し、前記セッション情報を前記第１のアプリケーションと第２のアプリケーションで共有する設定を行う手順と、
前記処理要求を、前記第１のアプリケーションへ転送して、前記第１のアプリケーションで前記共有されたセッション情報を用いて処理する手順と、
所定の条件を満たしたときには、前記第２のアプリケーションを起動し、前記第２のアプリケーションが起動した後に、新たに受け付けた前記処理要求を、前記第２のアプリケーションへ転送し、前記共有されたセッション情報を用いて前記第２のアプリケーションで処理する手順と、
前記第２のアプリケーションが起動した後に、前記第１のアプリケーションで前記処理要求の処理が完了すると、前記第１のアプリケーションを終了する手順と、を含み、
前記セッション情報を前記第１のアプリケーションと第２のアプリケーションで共有する設定を行う手順は、
前記第１のアプリケーションで使用されるセッション情報をセッション保存部に保存し

て、前記第２のアプリケーションがセッション情報を読み出すと、前記セッション保存部に保存された第１のアプリケーションが書き込んだセッション情報を得ることを特徴とするアプリケーションの高可用制御方法。

【請求項２】

前記識別子の異なる第１のアプリケーションと、第２のアプリケーションを生成する手順は、

前記処理要求の転送先を第１のアプリケーションから第２のアプリケーションへ切り替えるリクエスト転送部を生成する手順を含み、

前記リクエスト転送部が新たな処理要求を前記第２のアプリケーションへ転送する手順を実行することを特徴とする請求項１に記載のアプリケーションの高可用制御方法。

【請求項３】

前記識別子の異なる第１のアプリケーションと、第２のアプリケーションを生成する手順は、

前記予め設定したアプリケーションから識別子を含む第１の部分と、所定の処理を記述したプログラムを含む第２の部分を抽出する手順と、

前記第１の部分の識別子に第１のアプリケーションを示す新たな識別子を付加し、当該第１の部分と前記第２の部分を結合したものを第１のアプリケーションとして生成する手順と、

前記第１の部分の識別子に第２のアプリケーションを示す新たな識別子を付加し、当該第１の部分と前記第２の部分を結合したものを第２のアプリケーションとして生成する手順と、

を含むことを特徴とする請求項１に記載のアプリケーションの高可用制御方法。

【請求項４】

処理要求を受け付けて当該処理要求を前記予め設定したアプリケーションへ転送する処理を、さらに含み、

前記処理要求を前記予め設定したアプリケーションへ転送する処理は、

前記転送した処理要求に対する処理が完了したか否かを監視して、予め設定した時点以前に転送した処理要求に対する処理が全て完了しときに、完了を通知する手順と、

を含むことを特徴とする請求項１に記載のアプリケーションの高可用制御方法。

【請求項５】

処理要求を受け付けて当該処理要求を前記予め設定したアプリケーションへ転送する処理を、さらに含み、

前記転送した処理要求に対する処理が完了したか否かを監視して、予め設定した時点以前に転送した処理要求を記録する手順と、

前記予め設定した時点から所定時間経過したときに、前記記録した処理要求を消去する手順と、

前記記録した処理要求を消去した後に、完了を通知する手順と、

を含むことを特徴とする請求項１に記載のアプリケーションの高可用制御方法。

【請求項６】

前記第１のアプリケーションは、受け付けた処理要求を処理し、この処理結果を中継する手順と、

前記中継した処理結果を前記処理要求の送信元へ転送する手順と、

を含むことを特徴とする請求項１に記載のアプリケーションの高可用制御方法。

【請求項７】

処理要求を受け付ける旧アプリケーションを新アプリケーションへ入れ替えるアプリケーションのオンラインバージョン変更方法であって、

予め設定したアプリケーションから識別子の異なる前記新アプリケーションと、前記旧アプリケーションを生成する手順と、

前記複数のアプリケーションの実行に関する情報をセッション情報として生成し、前記セッション情報を前記旧アプリケーションと新アプリケーションで共有する設定を行う手

順と、
前記処理要求を、前記旧アプリケーションへ転送して、前記旧アプリケーションで前記共有されたセッション情報を用いて処理する手順と、
所定の条件を満たしたときには、前記新アプリケーションを起動し、前記新アプリケーションが起動した後に、新たに受け付けた前記処理要求を、前記新アプリケーションへ転送し、前記共有されたセッション情報を用いて前記新アプリケーションで処理する手順と、
前記新アプリケーションが前記起動した後に、前記旧アプリケーションで前記処理要求の処理が完了すると、前記旧アプリケーションを終了する手順と、を含み、
前記セッション情報を前記旧アプリケーションと新アプリケーションで共有する設定を行う手順は、
前記旧アプリケーションで使用されるセッション情報をセッション保存部に保存して、前記新アプリケーションがセッション情報を読み出すと、前記セッション保存部に保存された旧アプリケーションが書き込んだセッション情報を得ることを特徴とするアプリケーションのオンラインバージョン変更方法。
【請求項８】
メモリ上に第１のアプリケーションと第２のアプリケーションを配備する配備部と、
受け付けた処理要求を前記第１のアプリケーションと第２のアプリケーションのいずれか一方に転送するリクエスト転送部と、を備えた計算機システムにおいて、
前記第１のアプリケーションと第２のアプリケーションを切り替えるリプレース管理部と、
予め設定したアプリケーションから識別子の異なる前記第１のアプリケーションと、前記第２のアプリケーションを生成し、前記第１のアプリケーションまたは第２のアプリケーションの実行を管理するアプリケーション管理部と、を有し、
前記リプレース管理部は、
所定の条件を満たしたときには、前記配備部に第２のアプリケーションを起動させた後に、処理要求の転送先を前記第１のアプリケーションから前記第２のアプリケーションに切り替える指令を前記リクエスト転送部へ送出し、
前記第１のアプリケーションの処理が完了した後に、前記第１のアプリケーションを破棄するよう前記配備部へ指令し、
前記アプリケーション管理部は、
前記複数のアプリケーションの実行に関する情報をセッション情報として生成し、前記第１のアプリケーションで使用されるセッション情報をセッション保存部に保存して、前記第２のアプリケーションがセッション情報を読み出すと、前記セッション保存部に保存された第１のアプリケーションが書き込んだセッション情報を取得することで、前記セッション情報を前記第１のアプリケーションと第２のアプリケーションで共有する設定を行い、
前記処理要求を、前記第１のアプリケーションへ転送して、前記第１のアプリケーションで前記共有されたセッション情報を用いて処理し、前記所定の条件を満たしたときには、前記第２のアプリケーションが起動した後に、新たに受け付けた前記処理要求を、前記第２のアプリケーションへ転送し、前記共有されたセッション情報を用いて前記第２のアプリケーションで処理することを特徴とする計算機システム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、インターネットなどのネットワーク上でアプリケーションサービスを実行するアプリケーションサーバの運用方法の改良に関する。
【背景技術】
【０００２】
近年情報システムはありとあらゆるところに使用され、情報システムに障害が発生する

とその影響は非常に大きく、社会問題となっている。情報システムはサーバやストレージなどのハードウェアと、ＯＳやアプリケーションなどのソフトウェアに分けられる。ハードウェア障害に対しては、ディスクや電源の二重化や、複数のサーバを用いたクラスタ構成により、障害が発生してもそれを隠蔽する技術が普及している。ソフトウェアの障害は、主にバグに起因する。ＯＳは通常長い年月をかけてバグフィックスが行われており、バグによる障害発生は少ない。一方、アプリケーションはＷｅｂ技術の進歩により、Ｗｅｂアプリケーションとして実装する場合が増えている。こうしたＷｅｂアプリケーションを使用し、インターネットなどのネットワーク上でアプリケーションサービスを提供するＷｅｂサイトでは、Ｗｅｂサイト間の競争が激化しており、各Ｗｅｂサイトとも顧客の嗜好に合わせ迅速な機能修正や追加に追われている。迅速な機能の修正や追加は、すなわち開発とテストを限られた時間内で行うことを意味しており、必ずしも十分なテストができるとは限らない。従って、実際に顧客にサービスを提供する実運用を開始した後のＷｅｂアプリケーションにおいても、バグが残っている可能性が高く、それらのバグがシステムに障害を起し、サービスが停止する可能性がある。従って、情報システムの信頼性向上には、Ｗｅｂアプリケーションの信頼性向上が重要な課題となっている。前述の通り、Ｗｅｂアプリケーションの障害は、主にＷｅｂアプリケーションのバグに起因している。しかし、開発工数の制限などにより１００％バグを根絶することは不可能であると考えられる。従って、Ｗｅｂアプリケーションにバグが存在していても極力サービスを継続させることのできる運用方法に注目が集まっている。

【０００３】

上記バグによってＷｅｂアプリケーションが停止または異常の要因としては、メモリリークを代表とするリソースリークが知られている。

【０００４】

バグによって発生した障害の多くはＷｅｂアプリケーションを再起動することで一時的に解決することが知られていることから、Ｗｅｂアプリケーションを運営するサイトの中には、定期的にＷｅｂアプリケーションを再起動して障害を未然に防ぐ手法が知られている。

【０００５】

しかし、上記の手法では一時的にアプリケーションサービスを中断することになってしまう。このため、アプリケーションサービスを継続しながらアプリケーションのリソースリークを防止するものとして、アプリケーションサーバをクラスタ構成として、所定時間ごとにフェールオーバを実行し、旧現用系のＯＳやアプリケーションをリセット（再起動）することで、リソースリークによるアプリケーションの停止を未然に防ぐ技術が知られている（例えば、特許文献１）。

【０００６】

あるいは、アプリケーションの実行中に、同一のアプリケーションを他のメモリ空間で新たに起動し、現在提供中のアプリケーションサービスを新たに起動したアプリケーションに移行させる技術が知られている（例えば、特許文献２）。

【０００７】

また、新たに起動したアプリケーションを使用する際の性能低下を抑制する技術として、新しいアプリケーションに対する処理要求の量を初めは少量とし、時間の経過と共に増やして行く技術が知られている（例えば、特許文献３）。

【特許文献１】特開２００１－１８８６８４号
【特許文献２】特開２００２－２５９１４２号
【特許文献３】特開２００５－９２８６２号

【発明の開示】

【発明が解決しようとする課題】

【０００８】

しかしながら、上記特許文献１と２では、現用系のアプリケーションから新たに起動したアプリケーションへ移行する際に、現用系のアプリケーションを終了してから、新たに

起動した代替系のアプリケーションへ移行することになる。この現用系から代替系アプリケーションへの切替の際に、アプリケーションサービスが一時中断すると共に、アプリケーションの処理能力は一時的に低下するという問題がある。

【０００９】

また、特許文献１では、現在提供中のアプリケーションサービスで使用しているアプリケーション、ＯＳ、ハードウェアとはまったく異なる新規インスタンスのアプリケーション、ＯＳ，ハードウェアで運用を継続することになる。また、特許文献２では、従来サービスを提供していたアプリケーションと異なる新規のアプリケーションで運用を継続する。このため、新しく起動した代替系アプリケーションに切り替わった直後は、サーバのＣＰＵのキャッシュメモリやディスクキャッシュなど、計算機上の各種のキャッシュに、アプリケーション実行に必要な情報が存在せず、キャッシュミスが頻発し、アプリケーションの実行性能が低下するという問題がある。このようにキャッシュに必要な情報が存在しない状態をＣｏｌｄ状態と呼ぶ。

【００１０】

このキャッシュのＣｏｌｄ状態の問題に対し、特許文献３では、新しく起動したアプリケーションに対する処理要求を最初は少なくし、時間と共に増加させていく事で、Ｃｏｌｄ状態のキャッシュを序所に暖め、ヒット率を向上する。これによって、代替系のアプリケーションを新規に起動した場合でも性能の低下をほとんど起こさないようにできる。しかし、この場合は、現用系から代替系アプリケーションへの移行に時間がかかり、場合によっては現用系アプリケーションで空きメモリが枯渇し、障害が発生する可能性がある。

【００１１】

そこで、アプリケーション全体を新規に起動した代替系のアプリケーションに入れ替えて、現用系アプリケーションを破棄し、メモリを解放するのではなく、リークしたリソースを含むアプリケーションの「一部」を解放するために、アプリケーションの一部を新規に起動して代替系とし、障害の発生を抑止しつつ、実行環境の一部がまだＣＰＵのキャッシュメモリ等に残っていることによりキャッシュミス率の増加を抑え、性能低下を最小限に抑えることが課題である。

【課題を解決するための手段】

【００１２】

本発明のアプリケーションはＷｅｂアプリケーションとする。Ｗｅｂアプリケーションは、ＪＡＶＡ（登録商標） Ｖｉｒｔｕａｌ Ｍａｃｈｉｎｅ（ＪＶＭ）やＪＶＭの上で動作するアプリケーションサーバ等のミドルウェア上で動作する。従来のアプリケーションは、ＯＳ上で動作することから、Ｗｅｂアプリケーションにおけるアプリケーションとは、ＪＶＭやアプリケーションサーバを含む概念と言える。しかし、ＪＶＭやアプリケーションサーバはソフトウェアベンダが提供するミドルウェアであり、出荷前に十分にデバッグ等がなされ、リソースリークのような不良はほとんど無いと言える。一方、Ｗｅｂアプリケーションは、上述のようにリソースリーク等のソフトウェアバグを含む可能性がある。そこで、Ｗｅｂアプリケーションのみを代替系のＷｅｂアプリケーションに入れ替える。すなわち本発明は、処理要求を受け付ける第１のアプリケーションと第２のアプリケーションを入れ替えるアプリケーションの高可用制御方法であって、予め設定したアプリケーションから識別子の異なる前記第１のアプリケーションと、前記第２のアプリケーションを生成する手順と、前記複数のアプリケーションの実行に関する情報をセッション情報として生成し、前記セッション情報を前記第１のアプリケーションと第２のアプリケーションで共有する設定を行う手順と、前記処理要求を、前記第１のアプリケーションへ転送して、前記第１のアプリケーションで前記共有されたセッション情報を用いて処理する手順と、所定の条件を満たしたときには、前記第２のアプリケーションを起動し、前記第２のアプリケーションが起動した後に、新たに受け付けた前記処理要求を、前記第２のアプリケーションへ転送し、前記共有されたセッション情報を用いて前記第２のアプリケーションで処理する手順と、前記第２のアプリケーションが前記起動した後に、前記第１のアプリケーションで前記処理要求の処理が完了すると、前記第１のアプリケーションを終

了する手順と、を含み、前記セッション情報を前記第１のアプリケーションと第２のアプリケーションで共有する設定を行う手順は、前記第１のアプリケーションで使用されるセッション情報をセッション保存部に保存して、前記第２のアプリケーションがセッション情報を読み出すと、前記セッション保存部に保存された第１のアプリケーションが書き込んだセッション情報を得る。

【００１３】

また、予め設定したアプリケーションから識別子の異なる前記第１のＷｅｂアプリケーションと、前記第２のＷｅｂアプリケーションを生成する。

【発明の効果】

【００１４】

したがって、本発明は、既に実行していた第１のＷｅｂアプリケーションを終了する際に、第１のＷｅｂアプリケーションが使用していたリークしたメモリを解放することができ、メモリリークによるＷｅｂアプリケーションの障害発生を予防することができる。さらに、Ｗｅｂアプリケーションのみを第二のＷｅｂアプリケーションに入れ替え、ＪＶＭやアプリケーションサーバは入れ替えずにそのまま使用し続けるため、ＣＰＵやディスクキャッシュ等にＪＶＭやアプリケーションサーバのコードを残すことができるので、第２のＷｅｂアプリケーションへ入れ替えた直後のキャッシュヒット率の低下が少なく、性能の劣化を抑制することが可能となる。

【００１５】

さらに、第１のＷｅｂアプリケーションから第２のＷｅｂアプリケーションへの入れ替えを、処理要求の受け付けを停止することなく実行できるので、計算機システムの処理能力を低下させることなく円滑な入れ替えを実現できる。

【発明を実施するための最良の形態】

【００１６】

以下、本発明の一実施形態を添付図面に基づいて説明する。

【００１７】

図１は、本発明を適用する計算機システムの構成図である。アプリケーションサービスを提供するＷｅｂサイト１は、ネットワーク２を介してクライアント３に接続されている。Ｗｅｂサイト１は、クライアント３から処理の実行要求を受け付けると、Ｗｅｂ層、アプリケーション層及びデータベース層の３つの階層からなるＷｅｂ３階層アプリケーション（ビジネスシステム）によって所定の処理（例えば、ビジネスロジック）を実行し、実行結果をクライアント３へ送信する。

【００１８】

Ｗｅｂ層は、クライアント３のＷｅｂブラウザが送信したＨＴＴＰによる処理要求を受け付けるＷｅｂサーバ計算機（以下、Ｗｅｂサーバとする）４が複数配置される。データベース層は、データや管理情報などを管理するデータベース管理システム（以下、ＤＢＭＳ）を実行するデータベースサーバ計算機６が複数配置される。そして、アプリケーション層は、Ｗｅｂサーバ４が受け付けた処理要求に対して、データベースサーバ計算機６からデータを取得し、取得したデータに所定の加工などを行ってＷｅｂサーバ４へ応答するアプリケーションサーバ計算機５を複数備える。

【００１９】

なお、Ｗｅｂサーバ４、アプリケーションサーバ計算機５及びデータベースサーバ計算機６は、図示はしないが、ＣＰＵ、メモリ、ストレージ装置を備えるもので、各サーバ計算機はネットワークを介して相互に接続される。このネットワークには、各サーバ４～６を制御するための管理端末７が接続される。管理端末７は、ＣＰＵ、メモリ、ストレージ装置、及び表示装置を備え、管理者からの操作を受け付けて各サーバ４～６へ指令することができる。

【００２０】

図２は、Ｗｅｂ３階層アプリケーション（業務システム）の各層のソフトウェア構成を示すブロック図である。

【００２１】
Ｗｅｂサーバ４は、ＣＰＵやメモリ及びストレージ装置からなるハードウェア４１上でオペレーティングシステム（以下、ＯＳ）４２を実行する。そして、ＯＳ４２上では、Ｗｅｂサーバ４３が実行されており、このＷｅｂサーバ４３は静的なコンテンツ４４と、アプリケーションサーバ計算機５から送られてきた動的なコンテンツをクライアント３に提供する。
【００２２】
アプリケーションサーバ計算機５は、ＣＰＵやメモリ及びストレージ装置からなるハードウェア５１上でＯＳ５２を実行する。そして、ＯＳ５２上では、第１のミドルウェアとしてＪＡＶＡ（登録商標）仮想マシン５３が実行されており、このＪＡＶＡ（登録商標）仮想マシン５３上で第２のミドルウェアとしてアプリケーションサーバ５４が実行される。アプリケーションサーバ５４は、Ｗｅｂサーバ４３が受け付けた処理要求に応じたＷｅｂアプリケーション５５を実行する。Ｗｅｂアプリケーション５５は、所定のビジネスロジックなどを処理し、データベースサーバ計算機６のデータベースサーバ６３に対してデータの読み書きを要求し、取得したデータに所定の処理を施してからＷｅｂサーバ４３に転送する。Ｗｅｂサーバ４３はアプリケーションサーバ５４から処理要求に対する応答を受信するとクライアント３が要求した処理の実行結果をクライアント３へ送信する。
【００２３】
データベースサーバ計算機６は、ＣＰＵやメモリ及びストレージ装置からなるハードウェア６１上でＯＳ６２を実行する。そして、ＯＳ６２上では、データベースサーバ６３が実行されており、このデータベースサーバ６３はＷｅｂアプリケーション５５から要求されたデータについてデータベース６４で読み書きを実行する。
【００２４】
なお、上記図１、図２において、Ｗｅｂ層、アプリケーション層、データベース層は異なる計算機で実行される例を示したが、これら３つの層が同一の計算機で実行されても良い。あるいは、Ｗｅｂ層とアプリケーション層が統合されたものであってもよい。また、Ｗｅｂ層、アプリケーション層、データベース層の各サーバ４～６は、ファイルシステムを備えたメモリ及びストレージ装置に接続されており、ファイルやデータなどを格納する。
【００２５】
図３は、アプリケーションサーバ５４の機能を示すブロック図である。図３では、アプリケーションサーバ５４がＷｅｂサーバ４からアプリケーションＡｐｐを実行する処理要求を受け付けた例を示している。アプリケーションサーバ５４は、アプリケーションサーバ計算機５に接続されたストレージ装置またはメモリのファイルシステム５６から、管理者の指示に従ってＷｅｂアプリケーションＡｐｐ．ｅａｒファイルを読み込んで、このファイルＡｐｐ．ｅａｒから２つのＷｅｂアプリケーションＡｐｐ１．ｅａｒと、ＷｅｂアプリケーションＡｐｐ２．ｅａｒを生成する。そしてアプリケーションサーバ５４は、後述するように実行するＷｅｂアプリケーションＡｐｐ１からＷｅｂアプリケーションＡｐｐ２へ入れ替えることで、リソースの解放を実行する。この際に、ＷｅｂアプリケーションＡｐｐ１とＡｐｐ２は同じＯＳ上のＪＶＭ上のアプリケーションサーバに配備されており、従って、ＯＳやＪＶＭやアプリケーションサーバのコードはＷｅｂアプリケーションの入れ替え時もＣＰＵなどのキャッシュに残り、従って従来技術でアプリケーション全体を起動して入れ替える場合と比べて、キャッシュミスの影響を最小化して性能低下を抑制できると共に、リソースリークによる障害の発生を予防することができる。
【００２６】
次に、アプリケーションサーバ５４の機能要素について説明する。
【００２７】
アプリケーションマネージャ５４１は、サービスを提供するＷｅｂアプリケーション５５のオリジナルであるＷｅｂアプリケーションのファイルＡｐｐ．ｅａｒから、現用系ＷｅｂアプリケーションＡｐｐ１．ｅａｒと代替系ＷｅｂアプリケーションＡｐｐ２．ｅａ

ｒ及びリクエストスイッチ５４４（Ａｐｐ．ｗａｒ）をメモリまたは記憶装置上のファイルシステム５６上に生成する。なお、現用系ＷｅｂアプリケーションＡｐｐ１．ｅａｒと代替系ＷｅｂアプリケーションＡｐｐ２．ｅａｒは、図２に示したＷｅｂアプリケーション５５として機能する。また、リクエストスイッチＡｐｐ．ｗａｒは、図中リクエストスイッチ５４４として機能する。以下の説明では、Ｗｅｂアプリケーション５５を現用系ＷｅｂアプリケーションＡｐｐ１．ｅａｒと代替系ＷｅｂアプリケーションＡｐｐ２．ｅａｒとして説明する。

【００２８】

リクエストスイッチ５４４は、ファイルシステム５６上でＡｐｐ．ｗａｒとして生成されるもので、Ｗｅｂサーバ４からの処理要求を現用系ＷｅｂアプリケーションＡｐｐ１．ｅａｒに転送する。そしてリクエストスイッチ５４４は、後述するリプレースマネージャ５４３から所定の指令があった場合は、Ｗｅｂサーバ４からの処理要求を代替系ＷｅｂアプリケーションＡｐｐ２．ｅａｒへ転送するように切り替える処理を実行する。

【００２９】

デプロイア５４８は、前記従来例と同様にＷｅｂアプリケーション（Ａｐｐ１、Ａｐｐ２）をアプリケーションサーバ５４に配備（デプロイ）したり、配備解除（アンデプロイ）する。配備解除の際には、配備済みアプリケーションが使用していたメモリ領域は未使用状態となり、その後発生するゴミ集め処理（ガーベージコレクション処理）により、回収されて、再利用可能となる。

【００３０】

Ｎデプロイア５４５は、前記デプロイア５４８とは異なり、現用系Ｗｅｂアプリケーションと代替系Ｗｅｂアプリケーションの間でセッション情報を共有するために、セッション共有モードで現用系や代替系のＷｅｂアプリケーションを配備する。

【００３１】

リプレースマネージャ５４３は、後述するように現用系ＷｅｂアプリケーションＡｐｐ１と代替系ＷｅｂアプリケーションＡｐｐ２の入れ替えを制御する。

【００３２】

デプロイツール５４７は、後述するように管理者が使用する管理端末７へのユーザインターフェースとＷｅｂアプリケーションの配備などの操作を実現する。

【００３３】

状態保持部（Ｓｔａｔｅ　Ｓｔｏｒｅ）５４６は、実行するアプリケーションのセッション情報等の状態を保持する。現用系ＷｅｂアプリケーションＡｐｐ１から代替系ＷｅｂアプリケーションＡｐｐ２に入れ替えられたとき、Ａｐｐ１で使用していた状態はＡｐｐ２でも使用できないと処理ができない。そこで、Ｓｔａｔｅ　Ｓｔｏｒｅは、後述するように、現用系と代替系Ｗｅｂアプリケーションの間でセッション情報の共有を実現している。

【００３４】

図３のメモリ上のファイルシステム５６は、ひとつのＷｅｂアプリケーション５５のオリジナルＡｐｐ．ｅａｒから現用系ＷｅｂアプリケーションＡｐｐ１と代替系ＷｅｂアプリケーションＡｐｐ２及びリクエストスイッチ５４４を生成した例を示す。なお、図３においては、代替系ＷｅｂアプリケーションＡｐｐ２がひとつの例を示すが、アプリケーションマネージャ５４１は複数の代替系ＷｅｂアプリケーションＡｐｐ２～ｎを生成することができ、これら複数の代替系ＷｅｂアプリケーションＡｐｐ２～ｎに対応するリクエストスイッチ５４４（Ａｐｐ．ｗａｒ）を生成することができる。

【００３５】

図４は、デプロイツール５４７の機能の一例を示すブロック図である。

【００３６】

デプロイツール５４７は、管理端末７にアプリケーションサーバ５４で実行するＷｅｂアプリケーション５５の入れ替えに関連する情報の提供と、入れ替えに関する指令の受付を行うユーザインターフェース５４７２と、ユーザインターフェース５４７２が受け付け

た指令を実行するコントローラ５４７１を備える。ユーザインターフェース５４７２は、後述の図５、図６のようにデプロイ操作画面とアプリケーションリスト画面を提供する。

【００３７】

図５は、ユーザインターフェース５４７２が管理端末７に提供するデプロイ操作画面１６０１の一例を示す説明図である。

【００３８】

デプロイ操作画面１６０１に示すファイルパス１６０２には、オリジナルのＷｅｂアプリケーションのファイル名が格納される。なお、このファイルの指定は管理者などが設定する。デプロイボタン１６０３は、ファイルパス１６０２で指定したオリジナルＷｅｂアプリケーション５５をデプロイ（配備）する。

【００３９】

リプレースチェックボックス１６０４は、ファイルパス１６０２で指定したオリジナルのアプリケーションＡｐｐ．ｅａｒから、リクエストスイッチＡｐｐ．ｗａｒや現用系ＷｅｂアプリケーションＡｐｐ１．ｅａｒや代替系ＷｅｂアプリケーションＡｐｐ２．ｅａｒを生成し、現用系ＷｅｂアプリケーションＡｐｐ１．ｅａｒから代替系ＷｅｂアプリケーションＡｐｐ２．ｅａｒへ入れ替え（リプレース）を行うか否かを設定する。図示のように、リプレースチェックボックス１６０４をチェックすることで、リプレースの実施を指令することができる。なお、この指令はリプレースマネージャ５４３に対して行われる。

【００４０】

リプレース条件１６０５は、現用系ＷｅｂアプリケーションＡｐｐ１．ｅａｒから代替系ＷｅｂアプリケーションＡｐｐ２．ｅａｒへの入れ替え条件の種類と、入れ替え条件の値を設定することができる。図示の例では、入力されたインターバル（秒）１６０６または空きヒープ（ＪＡＶＡ（登録商標）仮想マシンのヒープメモリ領域のうちの空き容量を指定）１６０７のいずれかのチェックボックスにチェックを入れ、入力欄に入れ替え条件の値を設定する。図示の例では、リプレースの条件の種類としてインターバル（時間間隔）が選択され、６００秒の値が設定された場合を示している。これら各設定値は、後述するリプレースマネージャ５４３へ格納され、Ｗｅｂアプリケーション５５は、現用系ＷｅｂアプリケーションＡｐｐ１．ｅａｒが実行を開始してから６００秒を経過すると、後述するように代替系ＷｅｂアプリケーションＡｐｐ２．ｅａｒへ入れ替えられる。上述のリプレース条件は、インターバルと空きヒープ以外の条件であっても良い。

【００４１】

上記デプロイ操作画面１６０１の各設定は、リプレースマネージャ５４３の入れ替え条件テーブル５４３０（図２７参照）へ格納される。

【００４２】

図６は、ユーザインターフェース５４７２が管理端末７へ提供するアプリケーションリスト画面６１１の一例を示す説明図である。管理端末７で表示するアプリケーションリスト画面６１１には、現在、アプリケーションサーバ５４で操作可能なＷｅｂアプリケーション５５の状態が表示され、これらのＷｅｂアプリケーション５５の開始、停止、アンデプロイ（配備解除）を制御することができる。

【００４３】

コンテキスト６１２には、ファイルパス１６０２で指定されたアプリケーションのコンテキスト名が表示される。ファイル名６１３には、ファイルパス１６０２で指定されたアプリケーションのファイル名が拡張子とともに表示される。

【００４４】

状態６１４は、アプリケーションの実行状態が表示され、「ｒｕｎ」は実行中を示し、「ｓｔｏｐ」は停止中を示している。

【００４５】

リプレース状態６１５は、「ｏｎ」であれば現用系Ｗｅｂアプリケーションから代替系Ｗｅｂアプリケーションへの入れ替えを行うことを意味し、「ｏｆｆ」の場合には、入れ

替えしない。開始ボタン６１６は、メモリ上にデプロイ（配備）したＷｅｂアプリケーション５５の開始を指令し、停止ボタン６１７は実行中のＷｅｂアプリケーション５５の停止を指令する。アンデプロイボタン６１８は、実行を停止したＷｅｂアプリケーション５５をメモリからアンデプロイ（配備解除）する。

【００４６】

コンテキスト６１２、ファイル名６１３、状態６１４、リプレース状態６１５は、コントローラ５４７１に記録された設定情報と実行情報をユーザインターフェース５４７２に提供したものである。

【００４７】

図７は、デプロイツール５４７のコントローラ５４７１で実行されるデプロイ処理の一例を示すフローチャートである。

【００４８】

この処理は、管理者などが管理端末７で図５のデプロイ操作画面１６０１を表示し、ファイルパス１６０２等を入力してから、デプロイボタン１６０３をクリックした場合に実行される。

【００４９】

Ｓ１では、リプレースチェックボックス１６０４がチェックされているかを判断する。リプレースチェックボックス１６０４がチェックされている場合にはＳ２に進み、チェックされていない場合にはＳ５へ進む。

【００５０】

Ｓ２では、コントローラ５４７１がアプリケーションマネージャ５４１を起動し、デプロイ操作画面１６０１のファイルパス１６０２に入力されたＷｅｂアプリケーション５５のオリジナルであるＷｅｂアプリケーションＡｐｐ．ｅａｒを読み込んで、現用系ＷｅｂアプリケーションＡｐｐ１．ｅａｒと代替系ＷｅｂアプリケーションＡｐｐ２．ｅａｒ及びリクエストスイッチＡｐｐ．ｗａｒを生成する。なお、これらアプリケーションやリクエストスイッチの生成については後述する。

【００５１】

Ｓ３では、コントローラ５４７１がＮデプロイア５４５を起動して、上記生成した現用系ＷｅｂアプリケーションＡｐｐ１．ｅａｒをメモリ上に配備する。このとき、Ｎデプロイア５４５は、現用系ＷｅｂアプリケーションＡｐｐ１．ｅａｒをセッション共有モードでメモリ上に配備する。なお、セッション共有モードは、現用系ＷｅｂアプリケーションＡｐｐ１．ｅａｒと代替系ＷｅｂアプリケーションＡｐｐ２．ｅａｒの間でセッション情報を共有するモードである。

【００５２】

Ｓ４では、コントローラ５４７１がデプロイア５４８に上記生成したリクエストスイッチＡｐｐ．ｗａｒをメモリ上に配備させ、処理を終了する。

【００５３】

上記Ｓ１の判定で、リプレースチェックボックス１６０４がチェックされていないＳ５では、Ｗｅｂアプリケーション５５を単体で実行させればよいので、コントローラ５４７１がデプロイア５４８を起動してＷｅｂアプリケーション５５のオリジナルである実行アプリケーションＡｐｐ．ｅａｒを配備させる。

【００５４】

図８は、デプロイツール５４７のコントローラ５４７１で実行されるアプリケーションの開始処理の一例を示すフローチャートである。

【００５５】

この処理は、管理者が管理端末７で図６のアプリケーションリスト画面６１１を表示し、選択したコンテキスト６１２に対応する開始ボタン６１６をクリックした場合に実行される。

【００５６】

Ｓ１１では、図５のデプロイ操作画面１６０１で入力されるリプレースチェックボック

ス１６０４がチェックされているかを判断する。リプレースチェックボックス１６０４がチェックされていればＳ１２へ進む一方、チェックされていなければＳ１５に進む。

【００５７】

Ｓ１２では、コントローラ５４７１が、図６のアプリケーションリスト画面６１１で操作された開始ボタン６１６に対応するＷｅｂアプリケーション５５（Ａｐｐ１．ｅａｒ）の設定情報に従い、リプレース条件１６０５をリプレースマネージャ５４３に設定する。リプレースマネージャ５４３は取得したリプレース条件１６０５をリプレース対象のＷｅｂアプリケーション５５（Ａｐｐ１．ｅａｒ）に対応付けて設定する。

【００５８】

Ｓ１３ではコントローラ５４７１が現用系ＷｅｂアプリケーションＡｐｐ１．ｅａｒの開始を指示し、現用系ＷｅｂアプリケーションＡｐｐ１．ｅａｒのサービスを開始させる。

【００５９】

Ｓ１４ではコントローラ５４７１がリクエストスイッチＡｐｐ．ｗａｒの開始を指示し、リクエストスイッチＡｐｐ．ｗａｒのサービスを開始する。

【００６０】

上記Ｓ１１の判定で、アプリケーションのデプロイ設定時に設定したデプロイ操作画面１６０１のリプレースチェックボックス１６０４がチェックされていない場合には、Ｓ１５に進んで、コントローラ５４７１はアプリケーションＡｐｐ．ｅａｒのサービスを開始させる。

【００６１】

図９は、デプロイツール５４７のコントローラ５４７１で実行されるアプリケーションの停止処理の一例を示すフローチャートである。

【００６２】

この処理は、管理者が管理端末７の図６のアプリケーションリスト画面６１１を表示し、選択したコンテキスト６１２に対応する停止ボタン６１７をクリックした場合に実行される。

【００６３】

Ｓ２１では、コントローラ５４７１がデプロイ操作画面１６０１のリプレースチェックボックス１６０４がチェックされているかを判定し、チェックされていればＳ２２へ進み、チェックされていない場合にはＳ２５へ進む。

【００６４】

Ｓ２２においてコントローラ５４７１は、停止ボタン６１７に対応するアプリケーション（ここでは現用系ＷｅｂアプリケーションＡｐｐ１．ｅａｒとする）のリクエストスイッチＡｐｐ．ｗａｒのサービスを停止させる。つまり、リクエストスイッチＡｐｐ．ｅａｒによるＷｅｂサーバ４からの処理要求の転送を停止させる。

【００６５】

Ｓ２３では、上記Ｓ２２で停止したリクエストスイッチＡｐｐ．ｗａｒに関連する現在実行中またはメモリに配備された現用系ＷｅｂアプリケーションＡｐｐ１．ｅａｒまたはＡｐｐ２．ｅａｒ（入れ替えが発生すると代替系Ａｐｐ２．ｅａｒは現用系となり、現用系Ａｐｐ１．ｅａｒは代替系となるため）のサービスを停止させる。

【００６６】

Ｓ２４では、コントローラ５４７１が、リプレースマネージャ５４３のリプレース条件をクリアする。

【００６７】

上記Ｓ２１の判定で、リプレースチェックボックス１６０４がチェックされていないＳ２５では、コントローラ５４７１が、実行アプリケーションＡｐｐ．ｅａｒを停止するように指令する。

【００６８】

以上の処理によって、コントローラ５４７１は、まず、リクエストスイッチＡｐｐ．ｗ

ａｒのサービスを停止させて処理要求の転送を止めてから、実行中のＷｅｂアプリケーション５５（Ａｐｐ１．ｅａｒまたは Ａｐｐ２．ｅａｒ）のサービスを停止するよう指令する。

【００６９】

図１０は、デプロイツール５４７のコントローラ５４７１で実行されるアプリケーションのアンデプロイ(配備解除)処理の一例を示すフローチャートである。

【００７０】

この処理は、管理者が管理端末７で図６のアプリケーションリスト画面６１１を表示し、選択したコンテキスト６１２に対応するアンデプロイボタン６１８をクリックした場合に実行される。

【００７１】

Ｓ３１では、Ｗｅｂアプリケーション５５に関するデプロイ設定をデプロイ操作画面１６０１にて設定する際に、リプレースチェックボックス１６０４がチェックされているかを判定する。リプレースチェックボックス１６０４がチェックされていればＳ３２へ進み、チェックされていない場合にはＳ３４へ進む。

【００７２】

Ｓ３２では、コントローラ５４７１が、デプロイア５４８にリクエストスイッチＡｐｐ．ｗａｒをアンデプロイ（配備解除、メモリ上からの解放）するよう指令する。

【００７３】

Ｓ３３では、上記Ｓ３２で解放したリクエストスイッチＡｐｐ．ｗａｒに関連する現在実行中または配備された現用系ＷｅｂアプリケーションＡｐｐ１またはＡｐｐ２をアンデプロイするよう、コントローラ５４７１はNデプロイアに指示する。

【００７４】

上記Ｓ３１の判定で、リプレースチェックボックス１６０４がチェックされていない場合には、Ｓ３４に進んで、コントローラ５４７１は実行アプリケーションＡｐｐ．ｅａｒのアンデプロイを指示してメモリ上から解放する。

【００７５】

図１１は、図７のＳ２で実行されるアプリケーションマネージャ５４１の処理の詳細を示すブロック図である。

【００７６】

アプリケーションマネージャ５４１は、デプロイ操作画面１６０１で管理者が指定したアプリケーションのファイル名１６０２で識別されるアプリケーションＡｐｐから現用系ＷｅｂアプリケーションＡｐｐ１．ｅａｒと代替系ＷｅｂアプリケーションＡｐｐ２．ｅａｒ及びリクエストスイッチＡｐｐ．ｗａｒを生成する変換処理部５４１２と、現用系Ｗｅｂアプリケーション、代替系Ｗｅｂアプリケーション及びリクエストスイッチのアプリケーション名を変換するためのＩＤ変数５４１１を備える。

【００７７】

アプリケーションマネージャ５４１は、変換処理部５４１２がデプロイツール５４７のコントローラ５４７１から起動の指示を受けると、リプレースを実施するアプリケーションＡｐｐ．ｅａｒをファイルシステム５６から取得して、予め設定したリクエストスイッチ雛形５４４１と識別子定義ファイル５４４２を読み込む。なお、ファイルシステム５６はアプリケーションサーバ計算機５のストレージ装置またはメモリに設定された記憶域である。

【００７８】

ＩＤ変数５４１１は、実行アプリケーションの名称（ファイル名）に代わって、現用系Ｗｅｂアプリケーション、代替系Ｗｅｂアプリケーション及びリクエストスイッチの名称とする所定のＩＤ変数を格納する。本実施形態では、名称として設定するＩＤ変数として実行アプリケーションＡｐｐ．ｅａｒを読み込んだ時点のタイムスタンプを所定のフォーマット（例えば、「ＹＹＭＭＤＤＨＨＭＭ」＝年月時分）で取得したものをＩＤ変数として設定する。

【００７９】
変換処理部５４１２は、読み込んだ識別子定義ファイル５４４２と上記ＩＤ変数に基づいて、現用系Ｗｅｂアプリケーション、代替系Ｗｅｂアプリケーション及びリクエストスイッチの名称を設定する。図示の例では、現用系Ｗｅｂアプリケーションに「－１」を付加し、代替系Ｗｅｂアプリケーションに「－２」を付加し、リクエストスイッチ５４４のコンテキスト名（またはファイル名）に「－ｒｓ」を付加する例を示す。
【００８０】
図示の例では、ＩＤ変数５４１１は、実行アプリケーションＡＰＰ．ｅａｒをファイルシステム５６から読み込んだ年月時分「ＹＹＭＭＤＤＨＨ」に設定され、変換処理部５４１２は取得したＩＤ変数に、識別子定義ファイル５４４２に基づく添え字をアプリケーションのファイル名「Ａｐｐ」に付加して「ＡｐｐＹＹＭＭＤＤＨＨ－Ｘ」に変換する。なお、変換処理部５４１２は、現用系Ｗｅｂアプリケーションと代替系Ｗｅｂアプリケーションの拡張子をそのまま（．ｅａｒ）とし、リクエストスイッチの拡張子を「．ｗａｒ」に設定する。
【００８１】
図１２は識別子定義ファイル５４４２の一例を示す。
【００８２】
図１２の例では、現用系ＷｅｂアプリケーションにはＩＤ変数に加えて第１の識別子「－１」を付加し、代替系ＷｅｂアプリケーションにはＩＤ変数に加えて第２の識別子「－２」を付加し、リクエストスイッチにはＩＤ変数に加えて所定の識別子「－ＲＳ」を付加することが定義されている。
【００８３】
これにより、アプリケーションマネージャ５４１は、図１１において、「Ａｐｐ．ｅａｒ」というオリジナルの実行アプリケーションが指定されると、そのときの時刻をＩＤ変数に設定し、例えばこの「ＩＤ」が「０６１２０５１７５０」のときには、現用系Ｗｅｂアプリケーションの名称を「Ａｐｐ０６１２０５１７５０－１．ｅａｒ」として設定し、代替系Ｗｅｂアプリケーションの名称を「Ａｐｐ０６１２０５１７５０－２．ｅａｒ」として設定し、リクエストスイッチの名称を「Ａｐｐ０６１２０５１７５０－ｒｓ．ｅａｒ」として設定する。
【００８４】
ここで、現用系Ｗｅｂアプリケーション Ａｐｐ０６１２０５１７５０－１．ｅａｒと代替系Ｗｅｂアプリケーション Ａｐｐ０６１２０５１７５０－２．ｅａｒは、同一機能のプログラムであるがファイル名とコンテキスト名が異なるため、アプリケーションサーバ５４上で並列的に実行できる。
【００８５】
また、リクエストスイッチＡｐｐ０６１２０５１７５０－ｒｓ．ｗａｒは、リクエストスイッチ雛形５４４１のプログラムをベースとし、現用系Ｗｅｂアプリケーション Ａｐｐ０６１２０５１７５０－１．ｅａｒから代替系Ｗｅｂアプリケーション Ａｐｐ０６１２０５１７５０－２．ｅａｒへＷｅｂサーバ４からの処理要求を切り替える記述を付加したものである。
【００８６】
アプリケーションマネージャ５４１は、現用系Ｗｅｂアプリケーション、代替系Ｗｅｂアプリケーション及びリクエストスイッチを生成すると、ファイルシステム５６上に現用系Ｗｅｂアプリケーションと代替系Ｗｅｂアプリケーションの間で共有するセッション情報に関する設定である共有情報５４４３と、アプリケーション「Ａｐｐ．ｅａｒ」から生成した現用系Ｗｅｂアプリケーション、代替系Ｗｅｂアプリケーション及びリクエストスイッチの構成を示す構成管理ファイル５４４４を生成する。
【００８７】
セッション共有情報５４４３は図１３で示すように正規表現を用いて例えばＸＭＬで記述され、「Ａｐｐ０６１２０５１７５０」の添え字「－１」と「－２」を同等に扱うこと

が記述される。

【００８８】

構成管理ファイル５４４４は、図１４で示すように例えばＸＭＬで記述され、実行するアプリケーションのリクエストスイッチのファイル名が「Ａｐｐ０６１２０５１７５０－ｒｓ．ｗａｒ」であり、コンテキストが「Ａｐｐ」であることを示し、現用系Ｗｅｂアプリケーションのファイル名が「Ａｐｐ０６１２０５１７５０－１．ｅａｒ」であり、コンテキスト名が「Ａｐｐ０６１２０５１７５０－１」であることを示し、代替系Ｗｅｂアプリケーションのファイル名が「Ａｐｐ０６１２０５１７５０－２．ｅａｒ」であり、コンテキスト名が「Ａｐｐ０６１２０５１７５０－２」であることを示す。

【００８９】

ここで、ファイルの拡張子が「ｅａｒ」であるアプリケーションファイルは、そのアプリケーションに関連した各種の設定情報を記述したデプロイメントデスクリプタ群と、所定の処理を記述したプログラム群をパッケージングしたものである。デプロイメントデスクリプタはＸＭＬ等で記述されており、例えば、図１５に示す「Ａｐｐｌｉｃａｔｉｏｎ．ｘｍｌ」である。図１５の「Ａｐｐｌｉｃａｔｉｏｎ．ｘｍｌ」５４４５の＜ｃｏｎｔｅｘｔ－ｒｏｏｔ＞タグの部分にはアプリケーションのコンテキスト名を指定する。本例のＡｐｐｌｉｃａｔｉｏｎ．ｘｍｌは、第１のアプリケーション（現用系Ｗｅｂアプリケーション）ファイルＡｐｐ０６１２０５１７５０－１．ｅａｒのデプロイメントデスクリプタの例であり、コンテキスト名が「Ａｐｐ０６１２０５１７５０－１」に設定されている。

【００９０】

アプリケーションマネージャ５４１は、オリジナルアプリケーションのファイルが指示されると、そのパッケージを展開し、Ａｐｐｌｉｃａｔｉｏｎ．ｘｍｌなどのデプロイメントデスクリプタ群と、プログラム群に分解し、Ａｐｐｌｉｃａｔｉｏｎ．ｘｍｌの＜ｃｏｎｔｅｘｔ－ｒｏｏｔ＞タグに第１のアプリケーションの識別子（例えば、Ａｐｐ０６１２０５１７５０－１）を書き込み、それをプログラム群と合わせてパッケージングして第１のアプリケーションのファイルＡｐｐ０６１２０５１７５０－１．ｅａｒを生成する。

【００９１】

アプリケーションマネージャ５４１は、同様にして、Ａｐｐｌｉｃａｔｉｏｎ．ｘｍｌの＜ｃｏｎｔｅｘｔ－ｒｏｏｔ＞に第２のアプリケーション（代替系Ｗｅｂアプリケーション）のコンテキスト名を書き込み、それをプログラム群と合わせてパッケージングし第２のアプリケーションを生成する。

【００９２】

図１６は、アプリケーションマネージャ５４１の変換処理部５４１２で実行されるアプリケーションの生成処理の一例を示すフローチャートである。

【００９３】

アプリケーションマネージャ５４１は、オリジナルアプリケーションＡＰＰ．ｅａｒをファイルシステム５６から読み込むと、まず、Ｓ４１にて現在時刻を取得し、その値をＩＤ変数に設定する。なお、本実施形態では、ＩＤ変数に日時を用いる例を示したが、他のアプリケーションの名称と名称が重複せずに名称を一意に特定できる値であればなんでもよい。

【００９４】

Ｓ４２では、実行アプリケーションＡＰＰ．ｅａｒの名称「Ａｐｐ」に、ＩＤ変数の値と、識別子定義ファイル５４４２から読み込んだ第１の識別子である「－１」を付加したものを第１のアプリケーション（現用系Ｗｅｂアプリケーション）として生成する。

【００９５】

Ｓ４３では、同様に、ＩＤ変数の値と、識別子定義ファイル５４４２から読み込んだ第２の識別子である「－２」をアプリケーションの名称に付加したものを第２のアプリケーション（代替系Ｗｅｂアプリケーション）として生成する。

【００９６】
Ｓ４４では、同様に、ＩＤ変数の値と、識別子定義ファイル５４４２から読み込んだ所定の識別子である「－ＲＳ」を付加したものを名称とし、リクエストスイッチ雛形５４４１に第１及び第２のアプリケーションの名称を加えたファイルをリクエストスイッチとして生成する。そして、リクエストスイッチの情報を構成管理ファイル５４４４に格納し、また、第１及び第２アプリケーションの識別子を設定する。
【００９７】
以上の処理により、現用系Ｗｅｂアプリケーションと代替系Ｗｅｂアプリケーション及びリクエストスイッチが生成される。
【００９８】
図１７は、上記図１６のＳ４２，Ｓ４３で行われる第１及び第２アプリケーションの生成処理の詳細を示すフローチャートで、変換処理部５４１２が実行する処理を示す。
【００９９】
変換処理部５４１２は、オリジナルアプリケーションのファイルＡＰＰ．ｅａｒを読み込むと、Ｓ５１で、パッケージを解凍し、デプロイメントデスクリプタ群とプログラム群を取り出す。
【０１００】
次に、変換処理部５４１２はＳ５２で、デプロイメントデスクリプタの一つであるＡｐｐｌｉｃａｔｉｏｎ．ｘｍｌの＜ｃｏｎｔｅｘｔ－ｒｏｏｔ＞に、オリジナルアプリケーションのコンテキスト名とＩＤ変数の値と第一のアプリケーションの識別子とを結合した文字列を設定し、Ｓ５３では、更新したデプロイメントデスクリプタ群とプログラム群をパッケージ化して第１のアプリケーションのファイルＡｐｐ０６１２０５１７５０－１．ｅａｒを生成する。
【０１０１】
なお、第２のアプリケーションも上記の処理を実行することで生成する。
【０１０２】
図１８は、図１６のＳ４４で行われるリクエストスイッチの生成処理の詳細を示すフローチャートである。変換処理部５４１２はＳ６１にて、構成管理ファイル５４４４におけるリクエストスイッチのコンテキスト名を参照し、その値をデプロイメントデスクリプタ雛形の＜ｃｏｎｔｅｘｔ－ｒｏｏｔ＞に設定してリクエストスイッチ用のデプロイメントデスクリプタを生成する。なお、warファイルのデプロイメントデスクリプタはｗｅｂ．ｘｍｌである。ｗｅｂ．ｘｍｌの雛形は図１９で示すように、<url－pattern>タグに「／＊」が設定されており、これは全てのリクエストをクライアントから受け付けることを意味している。
【０１０３】
Ｓ６２では、リクエストスイッチ雛形５４４１と生成したデプロイメントデスクリプタを合わせてパッケージングし、リクエストスイッチファイルをパッケージングして、上記図１６のＩＤ変数の値と、識別子定義ファイル５４４２の所定の識別子とを読み込んで生成されたファイル名のファイル、例えばＡｐｐ０６１２０５１７５０－ＲＳ．warを生成する。
【０１０４】
図２０は、図７のＳ３においてＮデプロイア５４５が実行するアプリケーションのデプロイ処理の一例を示すフローチャートである。
【０１０５】
Ｎデプロイア５４５は、Ｓ７１にて、Ｗｅｂアプリケーションのデプロイ時に、操作者がデプロイ操作画面１６０１でリプレース実施チェックボックス１６０４をチェックしたか否かを判定する。リプレース実施であればＳ７２へ進み、実施でなければＳ７７へ進む。
【０１０６】
Ｓ７２では、図１３に示したセッション共有情報５４４３に記述された正規表現のいず

れかと、指定された現用系または代替系のＷｅｂアプリケーションのコンテキスト情報が一致しているか否かを判定する。この判定では、一致していれば現用系Ｗｅｂアプリケーションと代替系Ｗｅｂアプリケーションでセッションの共有を行うためＳ７３へ進み、一致していなければセッションの共有は行わずに、Ｓ７７へ進む。

【０１０７】

次に、Ｓ７３では指定されたＷｅｂアプリケーションのコンテキスト情報が既に共有コンテキストマップ５４６１（図２１参照）に登録されているか否かを判定する。この判定は、Ｎデプロイア５４５が共有コンテキストマップ５４６１のコンテキスト正規表現列を参照し、指定されたコンテキスト情報と一致するエントリが存在するか否かにより判定を行う。共有コンテキストマップ５４６１にマッチするエントリが登録されていなければ、Ｓ７４に進む。

【０１０８】

Ｓ７４では、状態保持部５４６内に新しいセッションマップ５４６２を生成して、コンテキスト名と、生成したセッションマップ５４６２の組を共有コンテキストマップ５４６１に登録するすると共に、そのセッションマップを指定されたＷｅｂアプリケーションのセッションマップとして設定する。

【０１０９】

一方、Ｓ７３で既に指定されたＷｅｂアプリケーションのコンテキスト名が共有コンテキストマップ５４６１に登録されている場合には、Ｓ７５へ進み、登録されているセッションマップ５４６２を当該Ｗｅｂアプリケーションのセッションマップとして設定する。

【０１１０】

最後に、Ｓ７７でデプロイア５４８に、指定されたＷｅｂアプリケーションのファイル（例えば、Ａｐｐ０６１２０５１７５０－１．ｅａｒ）をメモリ上に配備させる。

【０１１１】

以上の処理により、Ｎデプロイア５４５は、Ｗｅｂアプリケーションをメモリ上に配備する。現用系のＷｅｂアプリケーションは、操作者が図５のデプロイ操作画面１６０１からＷｅｂアプリケーションのデプロイを指示した際に、Ｎデプロイアを呼び出して配備される。一方、代替系Ｗｅｂアプリケーションのデプロイは、後述するように、リプレースマネージャ５４３の入れ替え制御部５４３５からの指令があったときにＮデプロイア５４５がメモリ上に配備する。

【０１１２】

図２１は、状態保持部（Ｓｔａｔｅ　Ｓｔｏｒｅ）５４６の内容の一部を示すブロック図で、状態保持部５４６に格納される共有コンテキストマップ５４６１と、セッションマップ５４６２の関係を示す。

【０１１３】

共有コンテキストマップ５４６１には、Ｗｅｂアプリケーションのコンテキスト名の正規表現と、このコンテキスト名の正規表現に対応しセッション情報を蓄える５４６２のセッションマップへのポインタを組としたテーブルである。セッションマップ５４６２は、セッションＩＤと、このセッションＩＤに対応するセッションオブジェクトへのポインタを組としたマップである。

【０１１４】

図２２は、リクエストスイッチ５４４の機能要素を示すブロック図である。

【０１１５】

リクエストスイッチ５４４は、上述のようにＡｐｐ０６１２０５１７５０－ｒｓ．ｗａｒなどとしてメモリ上に配備され、Ｗｅｂサーバ４からの処理要求をＷｅｂアプリケーション５５に転送し、Ｗｅｂアプリケーション５５を現用系Ｗｅｂアプリケーションから代替系Ｗｅｂアプリケーションへ入れ替える際には、処理要求の転送先を現用系Ｗｅｂアプリケーションから代替系Ｗｅｂアプリケーションに切り替える。

【０１１６】

このため、リクエストスイッチ５４４には、処理要求の転送先Ｗｅｂアプリケーション

のコンテキスト名を格納する現コンテキスト６０１と、切り替え前に現用系Ｗｅｂアプリケーションとして使用していたＷｅｂアプリケーションのコンテキスト名を格納する旧コンテキスト６０２と、Ｗｅｂアプリケーションで現在処理中の処理要求の数を管理する処理中管理テーブル６０３と、現用系ＷｅｂアプリケーションにＷｅｂサーバ４からの処理要求を転送するリクエスト転送制御部６０４と、現コンテキスト６０１と旧コンテキスト６０２を入れ替えて処理要求の転送先を切り替える切り替え処理部６０５と、旧コンテキスト６０２のＷｅｂアプリケーションの処理が完了したか否かを判定する処理状態取得部６０６と、から構成される。なお、上記各部の詳細については後述する。

【０１１７】

図２３は、リクエストスイッチ５４４の処理中管理テーブル６０３の一例を示す説明図である。処理中管理テーブル６０３は、リクエストスイッチ５４４が処理要求の転送を行うＷｅｂアプリケーション５５の識別子を格納するｋｅｙと、処理中の処理要求の数を格納するｖａｌｕｅの２つのフィールドを備える。なお。ｋｅｙにはＷｅｂアプリケーション５５の識別子としてコンテキスト名などを設定することができる。

【０１１８】

処理中管理テーブル６０３は、主にリクエスト転送制御部６０４が管理するもので、Ｗｅｂサーバ４からの処理要求をＷｅｂアプリケーション５５へ転送すると、該当するＷｅｂアプリケーション５５の識別子のｖａｌｕｅの値を１だけインクリメントし、Ｗｅｂアプリケーション５５で処理要求の処理が完了するとｖａｌｕｅの値を１だけデクリメントする。すなわち、ｖａｌｕｅの値が正の整数となるｋｅｙに対応するＷｅｂアプリケーション５５が処理を実行していることを示す。

【０１１９】

図２４は、リクエストスイッチ５４４のリクエスト転送制御部６０４で行われる処理の一例を示すフローチャートである。この処理は、Ｗｅｂサーバ４から処理要求を受け付ける度に起動する。

【０１２０】

Ｓ８１では、リクエストスイッチ５４４は、Ｗｅｂサーバ４からの処理要求を受け付けると、リクエスト転送制御部６０４で処理を開始する。リクエスト転送制御部６０４は、Ｓ８２で現コンテキスト６０１から取得した現在使用中のＷｅｂアプリケーションのコンテキスト名を取得し、変数ｃｏｎｔｅｘｔへ代入する。

【０１２１】

次にＳ８３では、Ｗｅｂサーバ４からの処理要求に含まれるリクエストＵＲＬのコンテキスト部分を、Ｓ８２で取得したコンテキスト名＝変数ｃｏｎｔｅｘｔの値に書き換えて、リクエスト転送先のＵＲＬを生成する。

【０１２２】

次に、Ｓ８４でリクエスト転送制御部６０４は、処理中管理テーブル６０３のｋｅｙの値が、変数ｃｏｎｔｅｘｔと一致するエントリのｖａｌｕｅの値を１インクリメントする。Ｓ８５では、Ｓ８３で変更したＵＲＬに対し、Ｗｅｂサーバ４から受け付けた処理要求を転送する。すなわち、処理要求は現コンテキスト６０１で指定されたＷｅｂアプリケーション５５へ転送される。

【０１２３】

次に、Ｓ８６にてリクエスト転送制御部６０４は、処理要求を転送したＷｅｂアプリケーション５５から処理要求に対する結果を受け付ける。Ｓ８７では、処理中管理テーブル６０３のｋｅｙのうち、変数ｃｏｎｔｅｘｔに格納されたコンテキスト名と一致するエントリのｖａｌｕｅの値を１デクリメントする。

【０１２４】

Ｓ８８では、使用中のＷｅｂアプリケーション５５から得られた処理要求に対する結果をＷｅｂサーバ４を介してクライアントへ返送する。

【０１２５】

以上の処理により、現コンテキスト１に設定されたＷｅｂアプリケーション５５に対し

てＷｅｂサーバ４からの処理要求が転送され、処理中管理テーブル６０３が更新されていく。

【０１２６】

図２５は、リクエストスイッチ５４４の切り替え処理部６０５で行われる処理の一例を示すフローチャートである。リクエスト転送制御部６０４は、Ｗｅｂアプリケーション５５を現用系Ｗｅｂアプリケーションから代替系Ｗｅｂアプリケーションへ入れ替える処理の要求をリプレースマネージャ５４３から受けると、切り替え処理部６０５を起動する。

【０１２７】

Ｓ９１では、切り替え処理部６０５はリプレースマネージャ５４３から入れ替える処理の要求を受け付けると、その処理要求の引数に含まれる代替系Ｗｅｂアプリケーションのコンテキスト名を取得する。

【０１２８】

Ｓ９２では、切り替え処理部６０５が、現コンテキスト６０１に格納されていたコンテキスト名を旧コンテキスト６０２へコピーする。次に、Ｓ９３では、切り替え処理部６０５がリプレースマネージャ５４３から取得した代替系Ｗｅｂアプリケーションのコンテキスト名を現コンテキスト６０１へ格納する。

【０１２９】

Ｓ９４では、切り替え処理部６０５が処理中管理テーブル６０３のｋｅｙを検索し、現コンテキスト６０１と一致するｋｅｙのエントリが存在すればそのｖａｌｕｅの値を０に設定する。

【０１３０】

以上の処理により、Ｗｅｂサーバ４からの処理要求の転送先は現用系Ｗｅｂアプリケーションから代替系Ｗｅｂアプリケーションへ切り替えられて、代替系Ｗｅｂアプリケーションが処理の主体となる。

【０１３１】

例えば、現用系ＷｅｂアプリケーションＡｐｐ１．ｅａｒのコンテキスト名がＡｐｐ０６１２０５１７５０－１、代替系ＷｅｂアプリケーションＡｐｐ２．ｅａｒのコンテキスト名がＡｐｐ０６１２０５１７５０－２の場合、初期状態では現コンテキスト６０１にＡｐｐ０６１２０５１７５０－１が設定され、初期状態なので旧コンテキスト６０２には値は設定されていない。この状態で、リクエスト転送制御部６０４は、処理要求をコンテキスト名Ａｐｐ０６１２０５１７５０－１のＷｅｂアプリケーションに転送する。ここで、切り替え制御部６０５が代替系ＷｅｂアプリケーションＡｐｐ２．ｅａｒのコンテキスト名Ａｐｐ０６１２０５１７５０－２を引数として切り替え処理を要求すると、切り替え処理部６０５は現コンテキスト６０１の値Ａｐｐ０６１２０５１７５０－１を旧コンテキスト６０２にコピーし、現コンテキスト６０１にコンテキスト名Ａｐｐ０６１２０５１７５０－２を設定する。従って、リクエスト転送制御部６０４は以後Ｗｅｂサーバ４から送られてきた処理要求をコンテキスト名Ａｐｐ０６１２０５１７５０－２のＷｅｂアプリケーション、すなわち代替系に転送して処理することになる。

【０１３２】

図２６は、リクエストスイッチ５４４の処理状態取得部６０６で行われる処理の一例を示すフローチャートである。処理状態取得部６０６は、リプレースマネージャがＷｅｂアプリケーションのリクエスト処理の完了を調査する際に起動される。

【０１３３】

Ｓ１０１では、処理状態取得部６０６が、処理中管理テーブル６０３のｋｅｙが旧コンテキスト６０２と一致するｖａｌｕｅのエントリが存在しかつその値が０、つまり全ての処理要求が完了した状態、であるか否かを判定する。ｖａｌｕｅの値が０であれば旧コンテキスト６０２のＷｅｂアプリケーション５５は転送された処理要求を全て完了したので、処理状態取得部６０６は「処理完了」を示すステータスを返す。一方、上記ｖａｌｕｅの値が０でない場合には、旧コンテキスト６０２のＷｅｂアプリケーション５５にはまだ処理が完了していない処理要求があるため、処理状態取得部６０６は「処理未完了」のス

テータスを返す。

【０１３４】

図２７は、所定の条件となったときに、Ｗｅｂアプリケーションを現用系から代替系に入れ替える指示をするリプレースマネージャ５４３の構成を示すブロック図である。

【０１３５】

リプレースマネージャ５４３は、管理端末７が設定したＷｅｂアプリケーション５５の入れ替え条件を格納する入れ替え条件テーブル５４３０と、現用系Ｗｅｂアプリケーションから代替系Ｗｅｂアプリケーションへの入れ替えを指令する入れ替え制御部５４３５とを含んで構成される。

【０１３６】

入れ替え条件テーブル５４３０は、Ｗｅｂアプリケーション５５のコンテキスト名を格納するアプリケーションコンテキスト５４３１と、リプレースの条件を格納する入れ替え条件５４３２と、現用系ＷｅｂアプリケーションのＩＤを格納する現用系アプリケーションＩＤ５４３３と、代替系ＷｅｂアプリケーションのＩＤを格納する代替系アプリケーションＩＤ５４３４を有する。

【０１３７】

入れ替え条件５４３２には、図５のデプロイ操作画面で示した、インターバルや空きヒープなど条件の種類と、それに対応したインターバルの時間や空きヒープのバイト数が格納される。図示の例では、コンテキスト名が「ａｐｐ」の入れ替え条件５４３２は、空きヒープが１００ＭＢより小さくなったらＷｅｂアプリケーションの入れ替えを実施することを示す。現用系、及び代替系アプリケーションのＩＤとは、図１４の構成管理ファイルの＜ａｐｐｌｉｃａｔｉｏｎ－ｉｄ＞タグに指定された文字列を示す。本例では、現用系アプリケーションＩＤにａｐｐ．ａｐｐ１が指定されており、これは構成管理ファイル５４４４に記載されたＷｅｂアプリケーション、すなわち、Ｗｅｂアプリケーションのファイルがａｐｐ０６１２０５１７５０－１．ｅａｒ、コンテキスト名がａｐｐ０６１２０５１７５０－１であることを示す。また、代替系アプリケーションＩＤにａｐｐ．ａｐｐ２が指定されており、これは構成管理ファイル５４４４に記載されたＷｅｂアプリケーション、すなわちＷｅｂアプリケーションのファイルがａｐｐ０６１２０５１７５０－２．ｅａｒ、コンテキスト名がａｐｐ０６１２０５１７５０－２であることを示す。入れ替え制御部５４３５は、各アプリケーションコンテキスト５４３１毎に入れ替え条件５４３２を監視して、入れ替え条件５４３２が成立したときにはＷｅｂアプリケーション５５の入れ替えを実施する。

【０１３８】

図２８は、リプレースマネージャ５４３の入れ替え制御部５４３５で行われる処理の一例を示すフローチャートで、入れ替え条件５４３２がインターバルの場合を示す。なお、この処理は、入れ替え条件テーブル５４３０の各エントリ毎に起動される。

【０１３９】

Ｓ１１１では、入れ替え制御部５４３５が図６のアプリケーションリスト画面のリプレース実施欄６１５、すなわち当該Ｗｅｂアプリケーションにおいて入れ替えをするかどうかの設定を読み込み、入れ替えを実施する設定になっていればＳ１１２へ進み、実施する設定になっていなければ処理を終了する。

【０１４０】

Ｓ１１２では、入れ替え条件テーブル５４３０に設定されたインターバルの設定時間だけ休止（Ｓｌｅｅｐ）し、所定時間を経過するとＳ１１３へ進む。Ｓ１１３では、入れ替え制御部５４３５は、入れ替え条件テーブル５４３０の代替系WebアプリケーションＩＤ５４３４に設定されたWebアプリケーションをデプロイするようにＮデプロイア５４５へ指令する。

【０１４１】

次に、Ｓ１１４で、入れ替え制御部５４３５は、デプロイを指令した代替系Ｗｅｂアプリケーションに対してアクセスが可能になったか否かを判定し、Ｎデプロイア５４５によ

るデプロイが完了し、Ｗｅｂアプリケーションのサービスが利用可能になるまで待つ。代替系Ｗｅｂアプリケーションへアクセスが可能になるとＳ１１５に進む。

【０１４２】

Ｓ１１５では、入れ替え制御部５４３５は、リクエストスイッチ５４４に対してアプリケーションコンテキスト５４３１を引数として、クライアントの処理要求の転送先を現用系Ｗｅｂアプリケーションから代替系Ｗｅｂアプリケーションへ切り替えを指令する。

【０１４３】

Ｓ１１６では、入れ替え制御部５４３５はリクエストスイッチ５４４に対してデプロイ対象のアプリケーション（現用系アプリケーション）の処理が完了したか否かを判定する。すなわち、リクエストスイッチ図２２の処理状態取得部６０６を呼び出し、図２６の現用系Ｗｅｂアプリケーションで処理中のすべての処理要求が完了しているかチェックし、処理完了のステータスが返った場合は、Ｓ１１７へ進み、処理未完了のステータスが返った場合は、再びＳ１１６を実行する。これによって、現用系Ｗｅｂアプリケーションにおけるすべての処理要求の処理が完了するまで待ってから、Ｓ１１７が実行されることになる。Ｓ１１７では、入れ替え制御部５４３５は、入れ替え条件テーブル５４３０の現用系ＷｅｂアプリケーションＩＤ５４３３で識別されるアプリケーションをアンデプロイするようＮデプロイア５４５に指令する。

【０１４４】

そして、Ｓ１１８で入れ替え制御部５４３５は、現用系ＷｅｂアプリケーションＩＤ５４３３の欄に格納されている値と、代替系ＷｅｂアプリケーションＩＤ５４３４の欄に格納されている値を入れ替える。

【０１４５】

以上の処理により、入れ替え制御部５４３５は、所定のインターバルの設定時間毎に代替系Ｗｅｂアプリケーションをデプロイさせてからリクエストスイッチ５４４へ処理要求の転送先の切り替えを指令し、現用系アプリケーションの処理が完了するまで待ってから、現用系Ｗｅｂアプリケーションをアンデプロイさせる。

【０１４６】

また、リクエストスイッチ５４４の処理状態取得部は、現用系Ｗｅｂアプリケーションから代替系Ｗｅｂアプリケーションへ切り替えを指定した時刻から所定時間（例えば、数秒）経過したときに、処理完了ステータスを通知するようにしてもよい。この場合、現用系Ｗｅｂアプリケーションの処理が非常に長い場合でも、強制的に現用系Ｗｅｂアプリケーションを停止させることができ、従って、確実に現用系Ｗｅｂアプリケーションから代替系Ｗｅｂアプリケーションへの入れ替えを実施でき、リソースリークによる障害の発生を予防できる。

【０１４７】

図２９は、リプレースマネージャ５４３の入れ替え制御部５４３５で行われる処理の一例を示し、入れ替えの条件が空きヒープの場合のフローチャートである。なお、この処理は、図２８と同様に、入れ替え条件テーブル５４３０のアプリケーションコンテキスト５４３１毎に起動される。

【０１４８】

Ｓ１２１では、上記図２８と同様にリプレースを実施する設定になっている場合にはＳ１２１以降を実施し、そうでない場合には終了する。

【０１４９】

Ｓ１２２では、入れ替え制御部５４３５は、空きヒープサイズを確認し、空きヒープが、入れ替え条件テーブル５４３０に設定された値以下になるまで待機する。空きヒープが所定の値以下になると、Ｓ１２３へ進む。Ｓ１２３～Ｓ１２８では、上記図２８と同様である。

【０１５０】

以上の処理により、入れ替え制御部５４３５は、ヒープサイズが所定の価以下になる度に、代替系Ｗｅｂアプリケーションをデプロイさせてからリクエストスイッチ５４４へ処

理要求の転送先の切り替えを指令し、現用系アプリケーションの処理が完了するまで待ってから、現用系Ｗｅｂアプリケーションをアンデプロイさせる。

【０１５１】

以上のように、本実施形態では、ひとつのＷｅｂアプリケーション５５のオリジナル（Ａｐｐ．ｅａｒ）から、ひとつの現用系Ｗｅｂアプリケーション（Ａｐｐ１．ｅａｒ）と少なくともひとつの代替系ＷｅｂアプリケーションＡｐｐ２．ｅａｒを生成し、さらに、現用系Ｗｅｂアプリケーションや代替系Ｗｅｂアプリケーションへ Ｗｅｂサーバやクライアントからの処理要求を転送するためのリクエストスイッチ５４４（Ａｐｐ．ｗａｒ）を生成し、現用系Ｗｅｂアプリケーションを代替系Ｗｅｂアプリケーションへ入れ替える入れ替え条件をリプレースマネージャ５４３に設定しておく。

【０１５２】

リプレースマネージャ５４３は、各Ｗｅｂアプリケーション５５毎に入れ替え条件テーブル５４３０に設定された入れ替え条件５４３２を監視し、条件が成立するとＮデプロイア５４５に代替系Ｗｅｂアプリケーションをデプロイさせてから、リクエストスイッチ５４４に切り替えを指令する。リクエストスイッチ５４４は、Ｗｅｂサーバ４からの処理要求の転送先を現用系Ｗｅｂアプリケーションから代替系Ｗｅｂアプリケーションへ切り替える。リプレースマネージャ５４３は、旧現用系Ｗｅｂアプリケーションの処理が全て完了したことを確認してから、旧現用系Ｗｅｂアプリケーションをアンデプロイして、旧現用系Ｗｅｂアプリケーションが使用していたメモリを解放する。

【０１５３】

このように、Ｗｅｂアプリケーション５５を実行するＯＳ，ＪＶＭ，アプリケーションサーバ、Ｗｅｂアプリケーションというソフトウェア階層のうち、リソースのリークが最も発生する恐れのあるＷｅｂアプリケーションの部分のみのメモリを解放することで、障害の発生を抑制し、さらに、ＯＳやＪＶＭやアプリケーションサーバはメモリやキャッシュに残っていることによって、キャッシュのヒット率の減少を最小限に抑え、従って、性能の低下を最小限に抑えることが可能となる。

【０１５４】

さらに、代替系Ｗｅｂアプリケーションをデプロイしてからリクエストスイッチ５４４を機能させて、現用系Ｗｅｂアプリケーションへの処理要求を、代替系Ｗｅｂアプリケーションへ転送して処理させるため、Ｗｅｂサーバ４は処理要求の結果を遅滞なく受け取ってクライアントへ転送できる。これにより、リソースリークを防ぐためのＷｅｂアプリケーションの入れ替え時に、前記従来例のように処理要求の受け付け中断などを行うことなく円滑にＷｅｂアプリケーションの入れ替えを実現することができるのである。

【０１５５】

なお、上記第１実施形態では、リクエストスイッチ５４４とリプレースマネージャ５４３を別のモジュールとして構成した例を示したが、図３０で示すように、リクエストスイッチ５４４とリプレースマネージャ５４３をひとつのモジュールにまとめても良い。この場合入れ替え条件テーブルは、単一のＷｅｂアプリケーション５５、つまり自アプリケーションについてのみ管理すればよく、上記と同様の作用効果を得ることができる。

【０１５６】

＜第２実施形態＞

図３１、図３２は第２の実施形態を示し、前記第１実施形態ではリクエストスイッチ５４４をＡｐｐ－ｒｓ．ｗａｒというファイルで構成したのに対し、本第２実施形態ではリクエストスイッチ５４４をアプリケーションサーバ５４へ組み込んだ例を示し、その他の構成は前記第１実施形態と同様である。

【０１５７】

図３１において、アプリケーションマネージャ５４１は、サービスを提供するＷｅｂアプリケーション５５のオリジナルであるアプリケーションＡｐｐ．ｅａｒから、現用系ＷｅｂアプリケーションＡｐｐ１．ｅａｒと代替系ＷｅｂアプリケーションＡｐｐ２．ｅａｒをメモリ（または記憶装置）上のファイルシステム５６上に生成する。

【０１５８】
なお、リクエストスイッチ５４４Ａは予めアプリケーションサーバ５４に設定されたモジュールとして構成され、アプリケーションマネージャ５４１が生成した現用系ＷｅｂアプリケーションＡｐｐ１．ｅａｒと代替系ＷｅｂアプリケーションＡｐｐ２．ｅａｒへ選択的にＷｅｂサーバ４からの処理要求を転送するように設定される。リクエストスイッチ５４４Ａの機能は、前記第１実施形態と同様である。
【０１５９】
図３２は、アプリケーションマネージャ５４１が、現用系ＷｅｂアプリケーションＡｐｐ１と代替系ＷｅｂアプリケーションＡｐｐ２．ｅａｒを生成する様子を示す図である。前記第１実施形態の図１１と同様に、アプリケーションマネージャ５４１は与えられたＷｅｂアプリケーション５５であるＡｐｐ．ｅａｒからＩＤ変数の値と識別子定義ファイル５４４２を取得し、現用系ＷｅｂアプリケーションＡｐｐ１と代替系ＷｅｂアプリケーションＡｐｐ２のコンテキスト名を生成する。本第２実施形態では、アプリケーションマネージャ５４１は、現用系Ｗｅｂアプリケーションと代替系Ｗｅｂアプリケーションのみを生成する。
【０１６０】
そして、リクエストスイッチ５４４Ａはアプリケーションサーバ５４に組み込まれているので、Ｎデプロイア５４５ではリクエストスイッチ５４４Ａのデプロイを行わない点が前記第１実施形態と相違する。その他の構成は前記第１実施形態と同様である。
【０１６１】
この場合も、前記第１実施形態と同様にＷｅｂサーバ４からの処理要求の転送先を現用系Ｗｅｂアプリケーションから代替系Ｗｅｂアプリケーションへ切り替えて、作用、効果を得ることができる。
【０１６２】
＜第３実施形態＞
図３３～図３８は、第３の実施形態を示し、前記第第２実施形態に、Ｗｅｂアプリケーション５５（Ａｐｐ．ｅａｒ）のバージョン変更をオンラインで行うオンラインバージョン変更機能を加えたもので、その他の構成は前記第１実施形態または第２実施形態と同様である。
【０１６３】
Ｗｅｂアプリケーション５５などでは、プログラムの開発者がバグフィックスや新機能の追加などで、バージョン（またはリビジョン）を変更(新しいものに置き換える場合と、古いものに戻す場合がある)することが一般的に行われている。本実施形態では、使用中のＷｅｂアプリケーション５５のバージョンをオンラインにて更新する例を示す。
【０１６４】
図３３は、前記第１実施形態の図５に示したデプロイ操作画面１６０１を変更したもので、リプレースの実施に関する情報の入力欄をすべて省略したもので、その他の構成は、前記第１実施形態と同様である。
【０１６５】
図３４は、アプリケーションサーバ５４が管理端末７へ提供するＷｅｂアプリケーション５５のバージョン変更操作画面６２０を示す。バージョン変更操作画面６２０では、バージョン変更の際に、現在実行中のＷｅｂアプリケーションの代わりに実行するＷｅｂアプリケーションのファイル名を入力するファイル名６２１と、バージョン変更の実施を指示するバージョン変更ボタン６２２を備える。
【０１６６】
管理者などが管理端末７から、ファイル名６２１を指定し、バージョン変更ボタン６２２をクリックすると、アプリケーションサーバ５４のアプリケーションマネージャ５４１へバージョン変更の指令が送信される。
【０１６７】
図３５は、アプリケーションマネージャ５４１の動作を示すブロック図である。アプリ

ケーションマネージャ５４１は、前記第１実施形態の図１１に示した変換処理部５４１２とＩＤ変数取得部５４１１に加え、Ｗｅｂアプリケーション５５のバージョン番号を管理するバージョン番号管理部５４１３を備える。

【０１６８】

バージョン番号管理部５４１３は、Ｗｅｂアプリケーション５５（Ａｐｐ．ｅａｒ）毎にバージョン番号を管理しており、アプリケーションを生成する際には、コンテキスト名にバージョン（またはリビジョン）番号を加えたものをＷｅｂアプリケーション５５の識別子として生成する。例えば、図中アプリケーションＡｐｐ．ｅａｒのバージョン番号が０の場合、アプリケーションＡｐｐ－ｒｅｖ０．ｅａｒとして生成する。なお、図３５ではＩＤ変数を省略したが、前記第１実施形態の図１１に示したように、アプリケーションのコンテキスト名にＩＤ変数を付加するものとする。

【０１６９】

図３６は、アプリケーションサーバ５４がバージョン変更の指令を受けたときの処理の一例を示すフローチャートである。

【０１７０】

Ｓ１３１では、アプリケーションマネージャ５４１を起動して、上記図３５で示したように、指定されたアプリケーションのコンテキスト名にバージョン番号（ｒｅｖＸ）を付加したものを、実行するアプリケーションＡｐｐ－ｒｅｖＸ．ｅａｒとして生成する。

【０１７１】

Ｓ１３２では、アプリケーションサーバ５４がＮデプロイア５４５を起動して、アプリケーションマネージャ５４１が生成したアプリケーションＡｐｐ－ｒｅｖＸ．ｅａｒをメモリ上に配備する。そして、Ｓ１３３では、アプリケーションサーバ５４がリクエストスイッチ５４４ＡにＷｅｂサーバ４からの処理要求を転送するアプリケーションＡｐｐ－ｒｅｖＸ．ｅａｒを設定し、処理を終了する。

【０１７２】

なお、前記第１実施形態に適用する場合では、Ｗｅｂサーバ４がリクエストスイッチＡｐｐ．ｗａｒにアプリケーションＡｐｐ－ｒｅｖＸ．ｅａｒを設定すればよい。

【０１７３】

図３７は、上記図３４に示すバージョン変更操作画面６２０でバージョン変更の指令があったときに、アプリケーションサーバ５４で行われる処理の一例を示すフローチャートである。

【０１７４】

Ｓ１４１では、アプリケーションサーバ５４は、バージョン変更操作画面６２０から受け付けたバージョン変更指令に基づいて、アプリケーションマネージャ５４１を起動して、新しいバージョン番号の付いたアプリケーションのコンテクストを持つアプリケーションを上記図３６で示したように生成する。

【０１７５】

次に、Ｓ１４２ではＮデプロイア５４５を起動して、アプリケーションマネージャ５４１が生成した新しいバージョン番号のアプリケーションＡｐｐ－ｒｅｖＸ．ｅａｒをメモリ上に配備する。

【０１７６】

次に、アプリケーションサーバ５４は、現用系Ｗｅｂアプリケーションとしての Ａｐｐ．ｅａｒを、代替系Ｗｅｂアプリケーションとしての Ａｐｐ－ｒｅｖＸ．ｅａｒへ入れ替えるようリクエストスイッチ５４４Ａに対して指令する。リクエストスイッチ５４４Ａでは、前記第１実施形態の図２５で示した処理を実行し、現コンテキスト６０１に格納されていたアプリケーションＡｐｐ．ｅａｒを旧コンテキスト６０２へコピーし、現コンテキスト６０１に　Ｓ１４４では、前記第１実施形態の図２９に示したＳ１２６と同様にして、旧コンテキスト６０２へ移動したアプリケーションの処理が完了するのを待つ。

【０１７７】

Ｓ１４５では、旧コンテキスト６０２のアプリケーションＡｐｐ．ｅａｒの処理が全て

完了すると、アプリケーションサーバ５４は、Ｎデプロイア５４５に旧コンテキスト６０２のアプリケーションをアンデプロイするよう指令する。

【０１７８】

以上の処理により、実行中のＷｅｂアプリケーション５５を異なるバージョンのアプリケーションに入れ替えることが可能となり、Ｗｅｂアプリケーション５５を停止さえることなく、かつ、Ｗｅｂサーバ４からの処理要求を受け付けながらバージョン変更を実現することが可能となる。

【０１７９】

なお、上記第３実施形態では、第２実施形態に適用した例を示したが、リクエストスイッチ５４４をＷｅｂアプリケーション５５であるＡｐｐ．ｅａｒから生成する第１実施形態に適用しても同様である。

【０１８０】

以上のように、第１実施形態～第３実施形態をまとめると、図３８～図４１で示すようになる。まず、アプリケーションマネージャ５４１が第１のアプリケーションと第２のアプリケーションを作成しておく。そして、図３８で示すように、アプリケーションサーバ５４では第１のアプリケーション（Ａｐｐ１）を実行し、リクエストスイッチ５４４（５４４Ａ）はクライアント３からの処理要求をＷｅｂサーバ４から受け付け、カウンタ（ｖａｌｕｅ）値をインクリメントしてから第１のアプリケーションへ転送し（１）、（２）、第１のＷｅｂアプリケーションが処理要求を実行する。第１のＷｅｂアプリケーションは処理要求の実行結果をリクエストスイッチ５４４へ返し（３）、リクエストスイッチ５４４はカウンタ（ｖａｌｕｅ）の値をデクリメントしてからＷｅｂサーバ４を介してクライアント３へ結果を返す。

【０１８１】

次に、図３９で示すように、リプレースマネージャ５４３はＮデプロイア５４５に指示して第２のＷｅｂアプリケーションをメモリ上に配備して、第２のＷｅｂアプリケーションのサービスが利用可能になるまで待つ。そして、図４０で示すように、第２のＷｅｂアプリケーションが完全に配備されたら、リプレースマネージャ５４３はリクエストスイッチ５４４に指示して処理要求の転送先を第１のＷｅｂアプリケーションから第２のＷｅｂアプリケーションへ切り替えると共に、第１のＷｅｂアプリケーションの処理が全て完了するのを待つ。

【０１８２】

そして、第１のＷｅｂアプリケーションにおける処理要求の処理が完了した後には、図４１で示すように、リプレースマネージャ５４３はＮデプロイア５４５に指示し、第１のＷｅｂアプリケーションを配備解除（アンデプロイ）する。これにより、リークした不要メモリを開放して障害の発生を予防できる。また、Ｗｅｂアプリケーション５５の実行環境のうち、ＯＳ，ＪＶＭ，アプリケーションサーバ、リクエストスイッチ５４４やリプレースマネージャ５４３等の実行環境を残しておくことで、第２のＷｅｂアプリケーションへ入れ替えた直後のＣＰＵキャッシュのヒット率低下による性能低下を抑制できる。また、リクエストスイッチ５４４により、処理要求の受付を停止することなくシームレスに第１のＷｅｂアプリケーションから第２のＷｅｂアプリケーションへ移行することができる。

【０１８３】

＜第４実施形態＞

図４２、図４３は、第４の実施形態を示し、前記第１～第３実施形態のリクエストスイッチ５４４に代わってＷｅｂアプリケーション５５（図中Ａｐｐ１）が直接処理結果をＷｅｂサーバ４からクライアント３へ返すもので、その他の構成は前記第１～第３実施形態と同様である。

【０１８４】

すなわち、図４２において、ＷｅｂアプリケーションＡｐｐ１は、Ｗｅｂサーバ４からの処理要求をリクエストスイッチ５４４から受け取る（１）、（２）。Ｗｅｂアプリケー

ションＡｐｐ１は、処理要求の実行結果をＷｅｂサーバ４からクライアント３へ返す。ここで、ＷｅｂアプリケーションＡｐｐ１は、直接Ｗｅｂサーバ４へ処理結果を返すので、リクエストスイッチ５４４は、ＷｅｂアプリケーションＡｐｐ１の処理が完了したことを検知することができない。

【０１８５】

そこで、ＷｅｂアプリケーションＡｐｐ１には現在実行中の処理要求の数をカウントし、リクエストスイッチ５４４に全処理要求の処理が完了したことを通知する完了フィルタ（Ｃｏｍｐｌｅｔｉｏｎ　Ｆｉｌｔｅｒ）５４９を設ける。完了フィルタ５４９は、図４３で示すように、リクエストスイッチ５４４からの処理要求をＷｅｂアプリケーションＡｐｐ１へ転送するリクエスト転送制御部５４９１と、ＷｅｂアプリケーションＡｐｐ１へ投入した処理要求と、ＷｅｂアプリケーションＡｐｐ１が出力した処理結果とをそれぞれカウントして、ＷｅｂアプリケーションＡｐｐ１における処理の完了を判定する処理中管理部５４９２と、を備える。処理中管理部５４９２は、前記第１実施形態の処理状態取得部６０６と同様に構成されて、ＷｅｂアプリケーションＡｐｐ１へ投入した処理が完了したこと（ステータス）をリクエストスイッチ５４４やリプレースマネージャ５４３に通知することができる。

【０１８６】

このように、Ｗｅｂアプリケーション５５が、直接Ｗｅｂサーバ４へ処理結果を返すような場合では、Ｗｅｂアプリケーション５５に完了フィルタ５４９を付加することで、前記第１～第３実施形態と同様に、第１のアプリケーションでの処理が完全に完了してから、当該第１のアプリケーションをアンデプロイすることができる。

【０１８７】

Ｗｅｂアプリケーション５５に付加するフィルタは、例えば、Ｓｅｒｖｌｅｔ　Ｆｉｌｔｅｒ等として実現できる。

【０１８８】

Ｓｅｒｖｌｅｔ　Ｆｉｌｔｅｒはアプリケーションコードに手をいれることなく、アプリケーションの出入り口に処理を付加するための仕組みであり、この機能を利用して、リクエストの処理完了を検知する。上記完了フィルタ５４９は特定のアプリケーションに依存しないため、アプリケーションマネージャ５４１は、指定されたＷｅｂアプリケーションから、現用系（Ａｐｐ１．ｅａｒ）と代替系（Ａｐｐ２．ｅａｒ）の２つのアプリケーションを生成する際に、現用系ＷｅｂアプリケーションＡｐｐ１．ｅａｒと代替系ＷｅｂアプリケーションＡｐｐ２．ｅａｒにそれぞれ完了フィルタ５４９を挿入する。

【０１８９】

また、図４２に示したリクエストスイッチ５４４は、前記第１実施形態の図２２に示したリクエストスイッチ５４４は、処理状態取得部６０６を実行せず、各Ｗｅｂアプリケーションの完了フィルタ５４９と連携して処理中の処理要求の数を管理する。

【０１９０】

なお、上記各実施形態においては、アプリケーションマネージャ５４１がオリジナルのＷｅｂアプリケーション５５から第１のアプリケーション（現用系Ｗｅｂアプリケーション）と第２のアプリケーション（代替系Ｗｅｂアプリケーションまたは現用系Ｗｅｂアプリケーションとはバージョンの異なるＷｅｂアプリケーション）及びリクエストスイッチＡｐｐ．ｗａｒを生成する例を示したが、第１及び第２のアプリケーション及びリクエストスイッチ５４４は、予めファイルシステム５６上に作成しておいても良い。本形態を採ることによる効果は、第１～第3実施形態の効果に加えて、処理結果がリクエストスイッチを経由しないため、応答性能が向上するという特徴がある。

【産業上の利用可能性】

【０１９１】

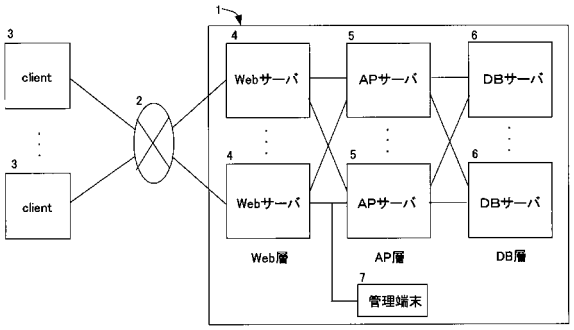
以上のように、本発明は、Ｗｅｂアプリケーションでサービスを提供する計算機システムやＷｅｂアプリケーションを制御するプログラムに適用することがえきる。

【図面の簡単な説明】

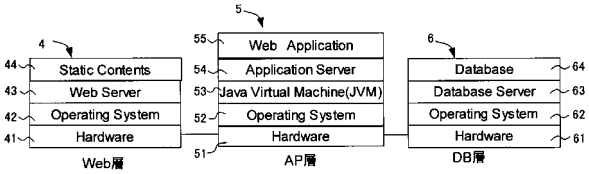
【０１９２】
【図１】第１実施形態を示し、本発明を適用する計算機システムのブロック図である。
【図２】第１実施形態を示し、Ｗｅｂ３階層アプリケーション（業務システム）の各層のソフトウェア構成を示すブロック図である。
【図３】第１実施形態を示し、アプリケーションサーバの機能を示すブロック図である。
【図４】第１実施形態を示し、デプロイツールの機能の一例を示すブロック図である。
【図５】第１実施形態を示し、ユーザインターフェースが管理端末へ提供するデプロイ操作画面の一例を示す説明図である。
【図６】第１実施形態を示し、ユーザインターフェースが管理端末へ提供するアプリケーションリスト画面６１１の一例を示す説明図である。
【図７】第１実施形態を示し、デプロイツールのコントローラで実行されるデプロイ処理の一例を示すフローチャートである。
【図８】第１実施形態を示し、デプロイツールのコントローラで実行されるアプリケーションの開始処理の一例を示すフローチャートである。
【図９】第１実施形態を示し、デプロイツールのコントローラで実行されるアプリケーションの停止処理の一例を示すフローチャートである。
【図１０】第１実施形態を示し、デプロイツールのコントローラで実行されるアプリケーションの解放（アンデプロイ）処理の一例を示すフローチャートである。
【図１１】第１実施形態を示し、図７のＳ２で実行されるアプリケーションマネージャの処理の詳細を示すブロック図である。
【図１２】第１実施形態を示し、識別子定義ファイルの一例を示す説明図。
【図１３】第１実施形態を示し、セッション共有情報の一例を示す説明図。
【図１４】第１実施形態を示し、構成管理ファイルの一例を示す説明図。
【図１５】第１実施形態を示し、ａｐｐｌｉｃａｔｉｏｎ．ｘｍｌの一例を示す説明図。
【図１６】第１実施形態を示し、アプリケーションマネージャの変換処理部で実行されるアプリケーションの生成処理の一例を示すフローチャートである。
【図１７】第１実施形態を示し、上記図１６のＳ４２，Ｓ４３で行われる第１及び第２アプリケーションの生成処理の詳細を示すフローチャートで、変換処理部が実行する処理を示す。
【図１８】第１実施形態を示し、図１６のＳ４４で行われるリクエストスイッチの生成処理の詳細を示す。
【図１９】第１実施形態を示し、デプロイメント記述子の一例を示す説明図。
【図２０】第１実施形態を示し、図８のＳ１３においてＮデプロイアが実行するアプリケーションのデプロイ処理の一例を示すフローチャートである。
【図２１】第１実施形態を示し、状態保持部（Ｓｔａｔｅ　Ｓｔｏｒｅ）の一部を示すブロック図で、共有コンテキストマップと、セッションマップの関係を示す。
【図２２】第１実施形態を示し、リクエストスイッチの機能要素を示すブロック図である。
【図２３】第１実施形態を示し、リクエストスイッチの処理中管理テーブルの一例を示す説明図である。
【図２４】第１実施形態を示し、リクエストスイッチのリクエスト転送制御部で行われる処理の一例を示すフローチャートである。
【図２５】第１実施形態を示し、リクエストスイッチの切り替え処理部で行われる処理の一例を示すフローチャート。
【図２６】第１実施形態を示し、リクエストスイッチの処理状態取得部で行われる処理の一例を示すフローチャート。
【図２７】第１実施形態を示し、リプレースマネージャの構成を示すブロック図である。
【図２８】第１実施形態を示し、リプレースマネージャの切り替え制御部で行われる処理の一例を示すフローチャートで、入れ替え条件がインターバルの場合を示す。
【図２９】第１実施形態を示し、リプレースマネージャの切り替え制御部で行われる処理

の一例を示し、切り替えの条件がシステムヒープの場合のフローチャートである。
【図３０】第１実施形態を示し、リクエストスイッチの他の構成を示すブロック図である。
【図３１】第２実施形態を示し、アプリケーションサーバの機能を示すブロック図である。
【図３２】第２実施形態を示し、アプリケーションマネージャの処理の詳細を示すブロック図である。
【図３３】第３実施形態を示し、ユーザインターフェースが管理端末へ提供するデプロイ操作画面の一例を示す説明図である。
【図３４】第３実施形態を示し、ユーザインターフェースが管理端末へ提供するバージョン変更操作画面の一例を示す説明図である。
【図３５】第３実施形態を示し、アプリケーションマネージャの動作を示すブロック図である。
【図３６】第３実施形態を示し、アプリケーションサーバがバージョン変更の指令を受けたときの処理の一例を示すフローチャートである。
【図３７】第３実施形態を示し、アプリケーションサーバで行われるバージョン変更処理の一例を示すフローチャートである。
【図３８】第１～第３実施形態を示し、アプリケーションサーバの処理の流れを示すブロック図で第１のアプリケーションの実行中を示す。
【図３９】第１～第３実施形態を示し、アプリケーションサーバの処理の流れを示すブロック図で第２のアプリケーションのデプロイ中を示す。
【図４０】第１～第３実施形態を示し、アプリケーションサーバの処理の流れを示すブロック図で第２のアプリケーションへの切り替え完了状態を示す。
【図４１】第１～第３実施形態を示し、アプリケーションサーバの処理の流れを示すブロック図で第１のアプリケーションのアンデプロイ後の状態を示す。
【図４２】第４実施形態を示し、アプリケーションサーバの処理の流れを示すブロック図。
【図４３】第４実施形態を示し、完了フィルタのブロック図。
【符号の説明】
【０１９３】
３　クライアント
４　Ｗｅｂサーバ
５４　アプリケーションサーバ
５５　Ｗｅｂアプリケーション
５４３　リプレースマネージャ
５４４　リクエストスイッチ
５４１　アプリケーションマネージャ
５４５　Ｎデプロイア
５４８　デプロイア
５４６　状態保持部
５４７　デプロイツール
Ａｐｐ１　現用系Ｗｅｂアプリケーション
Ａｐｐ２　代替系Ｗｅｂアプリケーション

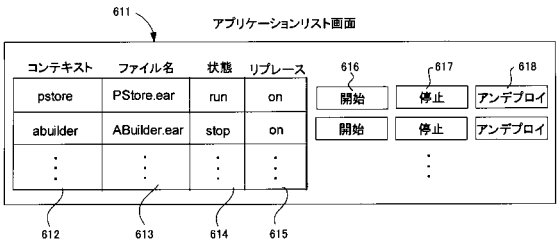
【図１】


1
2
3
client
3
client
4
Webサーバ
4
Webサーバ
5
APサーバ
5
APサーバ
6
DBサーバ
6
DBサーバ
Web層
AP層
DB層
7
管理端末

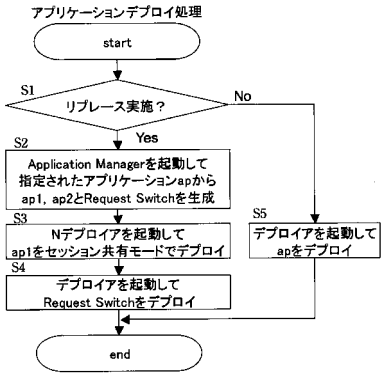
【図２】


4
Static Contents
Web Server
Operating System
Hardware
Web層
5
Web Application
Application Server
Java Virtual Machine(JVM)
Operating System
Hardware
AP層
6
Database
Database Server
Operating System
Hardware
DB層

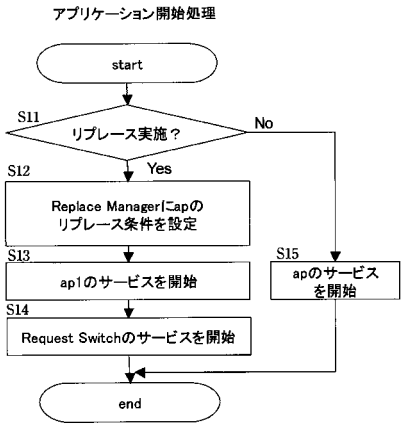
【図６】


611
アプリケーションリスト画面
コンテキスト
ファイル名
状態
リプレース
pstore
PStore.ear
run
on
abuilder
ABuilder.ear
stop
on
616
617
618
開始
停止
アンデプロイ
開始
停止
アンデプロイ
612
613
614
615

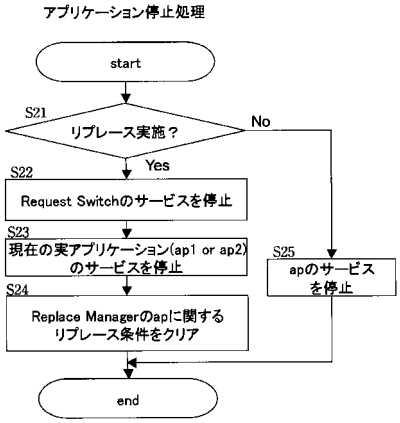
【図７】


アプリケーションデプロイ処理
start
S1
リプレース実施？
No
Yes
S2
Application Managerを起動して
指定されたアプリケーションapから
ap1，ap2とRequest Switchを生成
S3
Nデプロイアを起動して
ap1をセッション共有モードでデプロイ
S4
デプロイアを起動して
Request Switchをデプロイ
S5
デプロイアを起動して
apをデプロイ
end

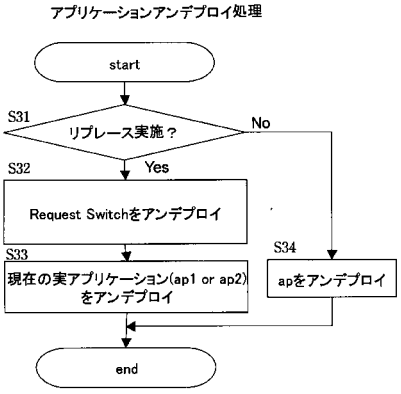
【図８】



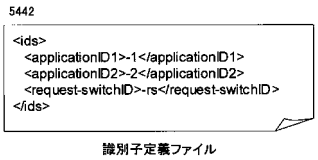

【図９】




【図１０】




【図１２】




【図１３】

【図１４】

5444

```
<application-set>
  <set-id>app</set-id>
  <request-switch>
    <request-switch-id>app.rs</request-switch-id>
    <file>app0612051750-rs.war</file>
    <context>app</context>
  </request-switch>
  <applications>
    <application>
      <application-id>app.app1</application-id>
      <file>app0612051750-1.ear</file>
      <context>app0612051750-1</context>
    </application>
    <application>
      <application-id>app.app2</application-id>
      <file>app0612051750-2.ear</file>
      <context>app0612051750-2</context>
    </application>
  </applications>
</application-set>
```

構成管理ファイル

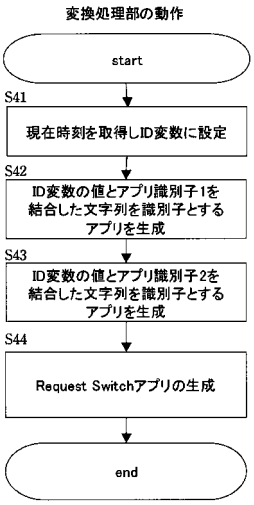
【図１５】

5445

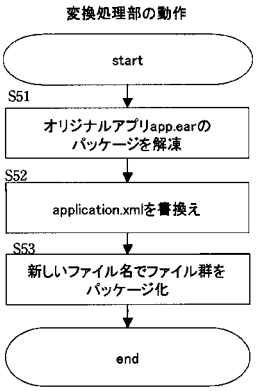
```
<application>
  ・・・
  <module>
    <web>
      ・・・
      <context-root>App0612051750-1</context-root>
    </web>
  </module>
</application>
```

application.xml

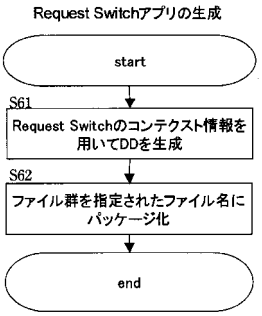
【図１６】



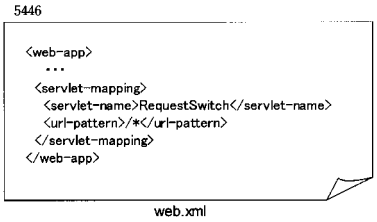

【図１７】



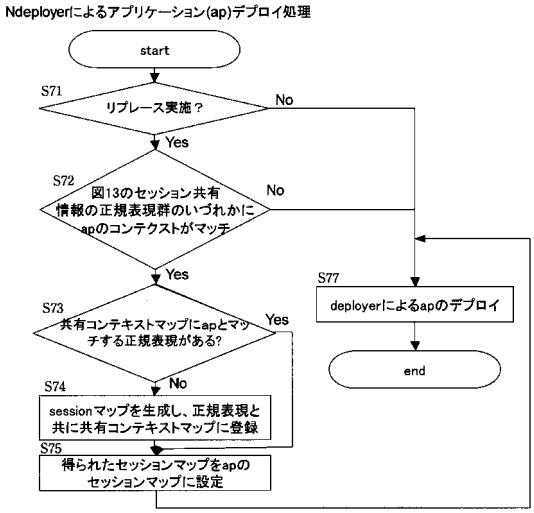

【図１８】



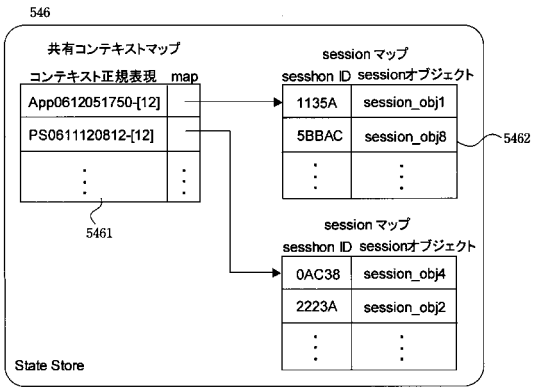
【図１９】



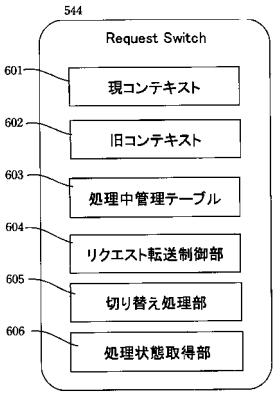

【図２０】



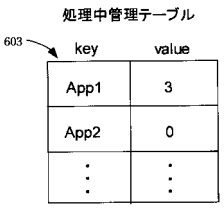

【図２１】



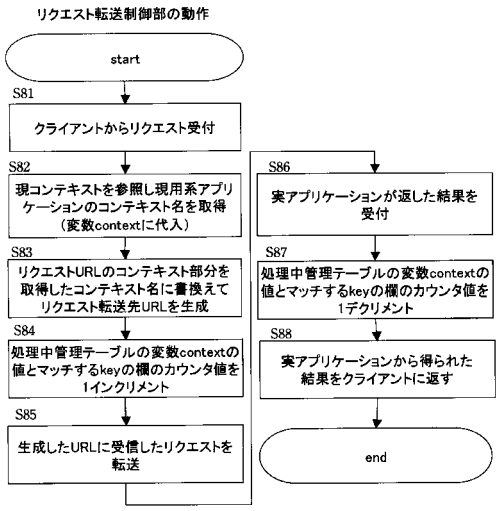

【図２２】



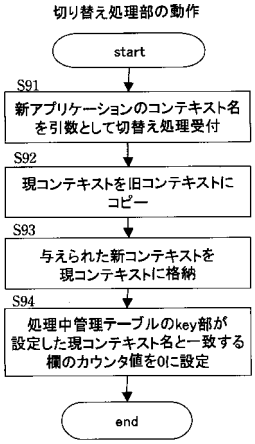

【図２３】



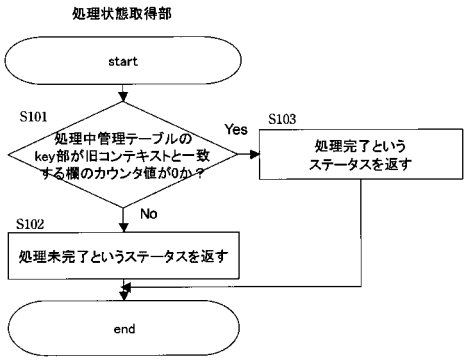
【図２４】



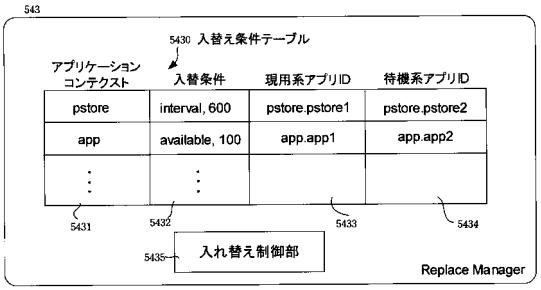

【図２５】



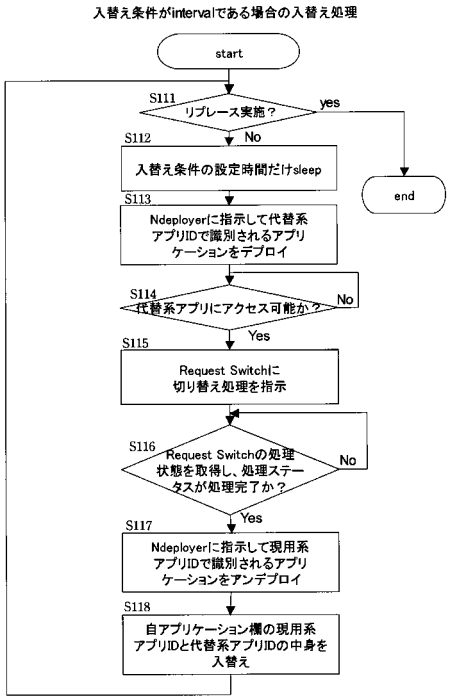

【図２６】



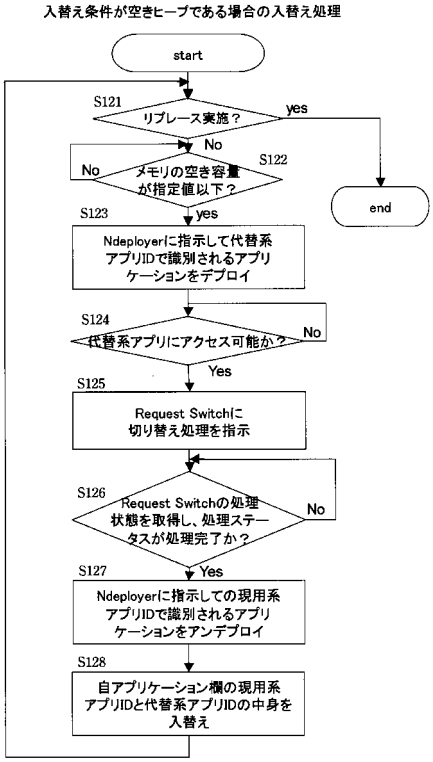

【図２７】



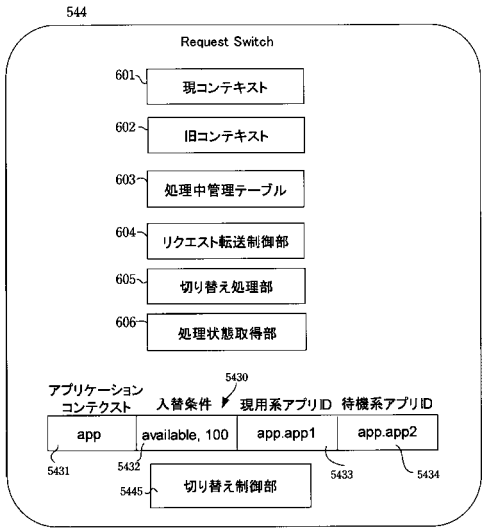
【図２８】



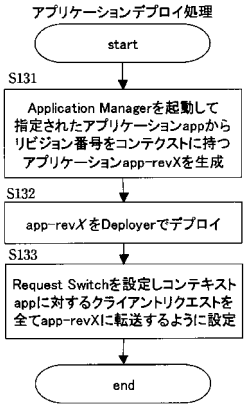

【図２９】



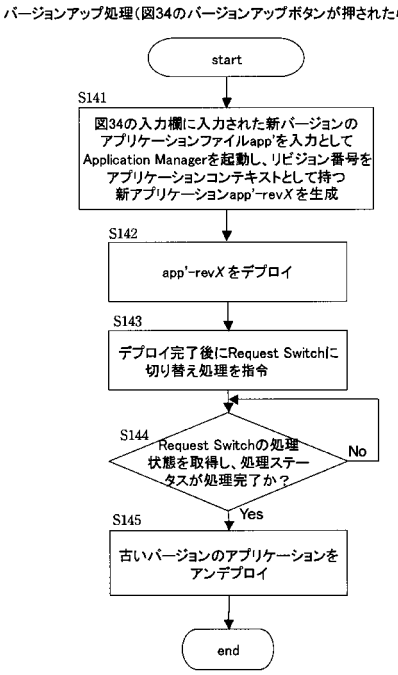

【図３０】



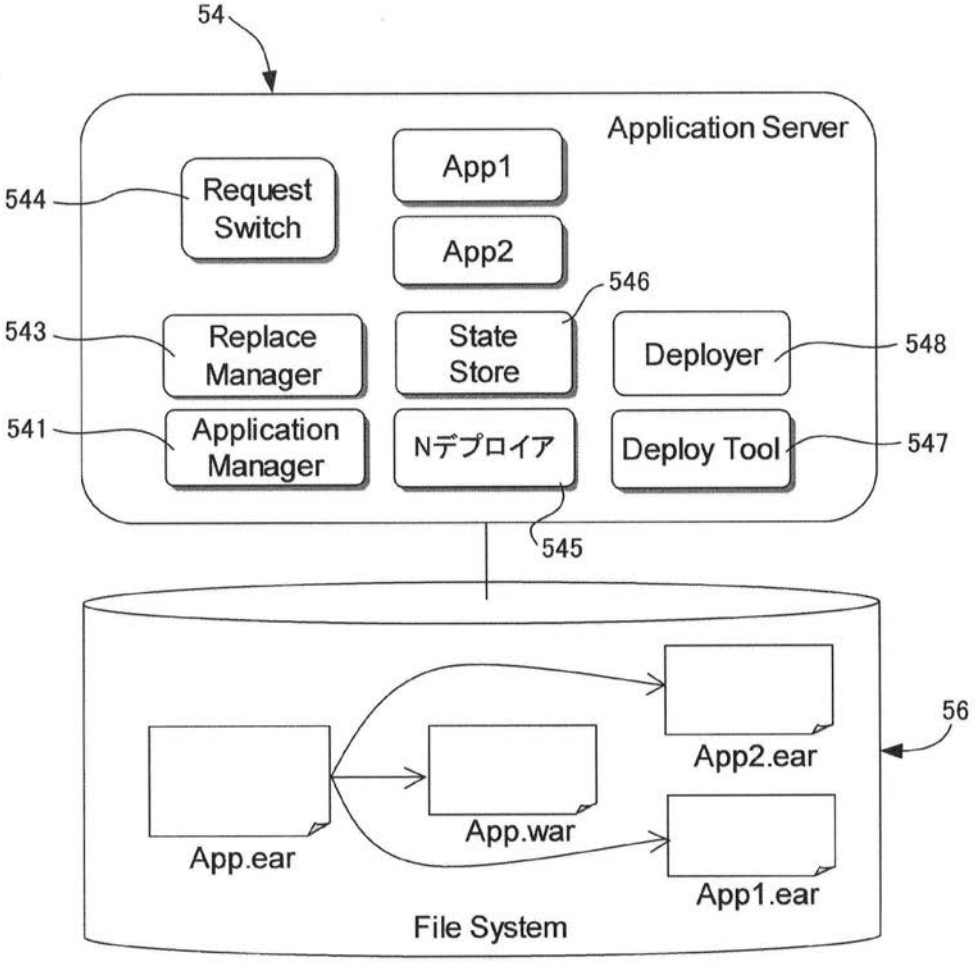

【図３６】



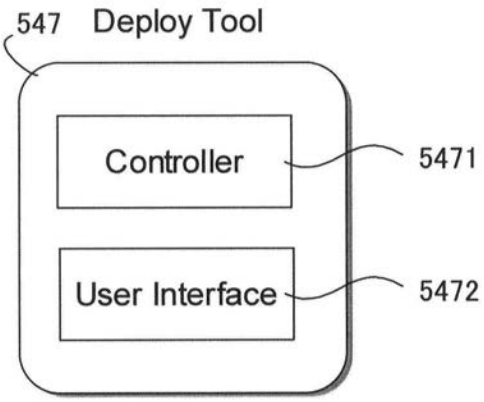
【図３７】


バージョンアップ処理（図34のバージョンアップボタンが押されたら開始）
start
S141
図34の入力欄に入力された新バージョンのアプリケーションファイルapp'を入力としてApplication Managerを起動し、リビジョン番号をアプリケーションコンテキストとして持つ新アプリケーションapp'-revXを生成
S142
app'-revXをデプロイ
S143
デプロイ完了後にRequest Switchに切り替え処理を指令
S144
Request Switchの処理状態を取得し、処理ステータスが処理完了か？
No
Yes
S145
古いバージョンのアプリケーションをアンデプロイ
end

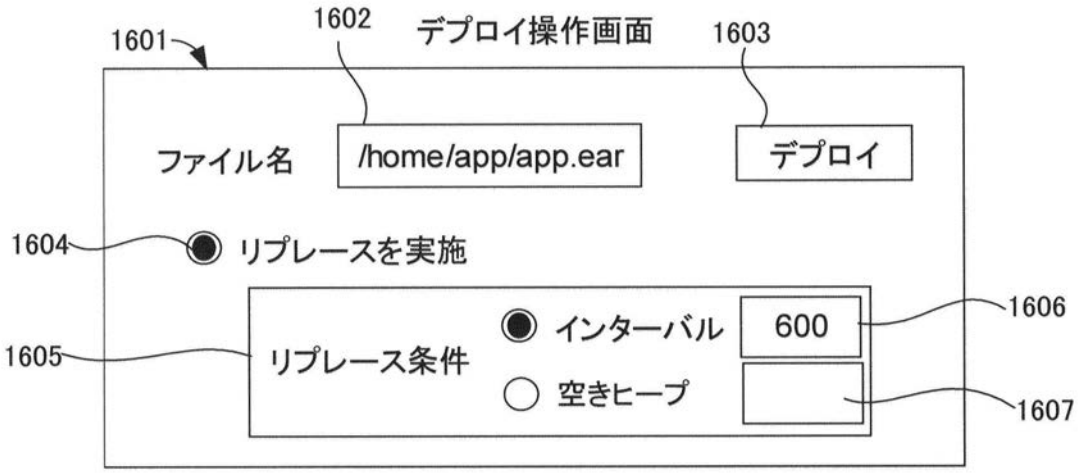
【図３】


54
Application Server
App1
App2
544
Request Switch
546
543
Replace Manager
State Store
Deployer
548
541
Application Manager
Nデプロイア
Deploy Tool
547
545
56
App2.ear
App.war
App.ear
App1.ear
File System

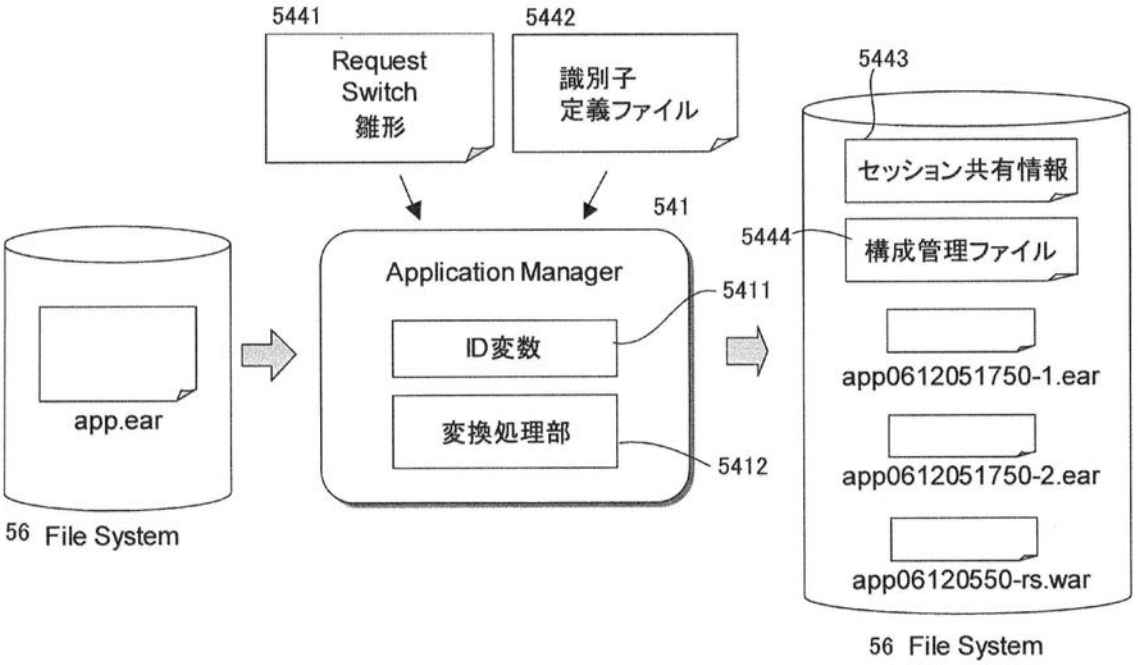
【図４】


547
Deploy Tool
Controller
5471
User Interface
5472

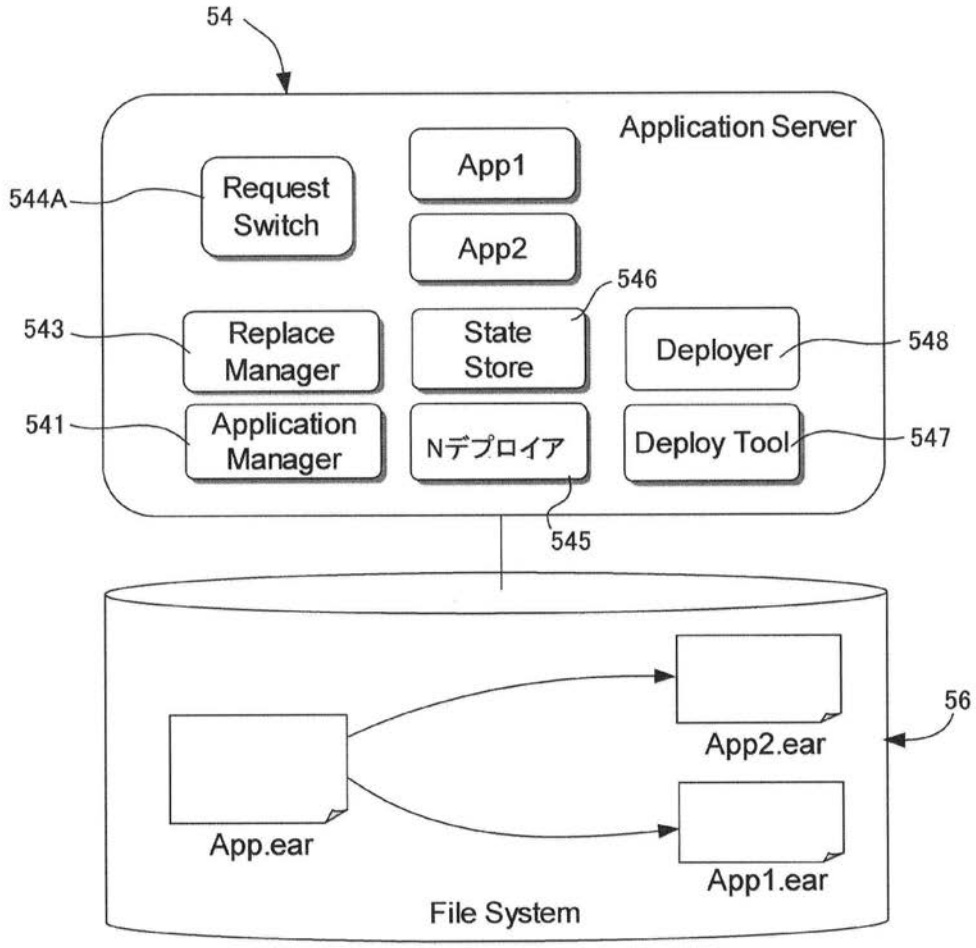
【図５】


1601
1602
デプロイ操作画面
1603
ファイル名
/home/app/app.ear
デプロイ
1604
リプレースを実施
1605
リプレース条件
インターバル
600
1606
空きヒープ
1607

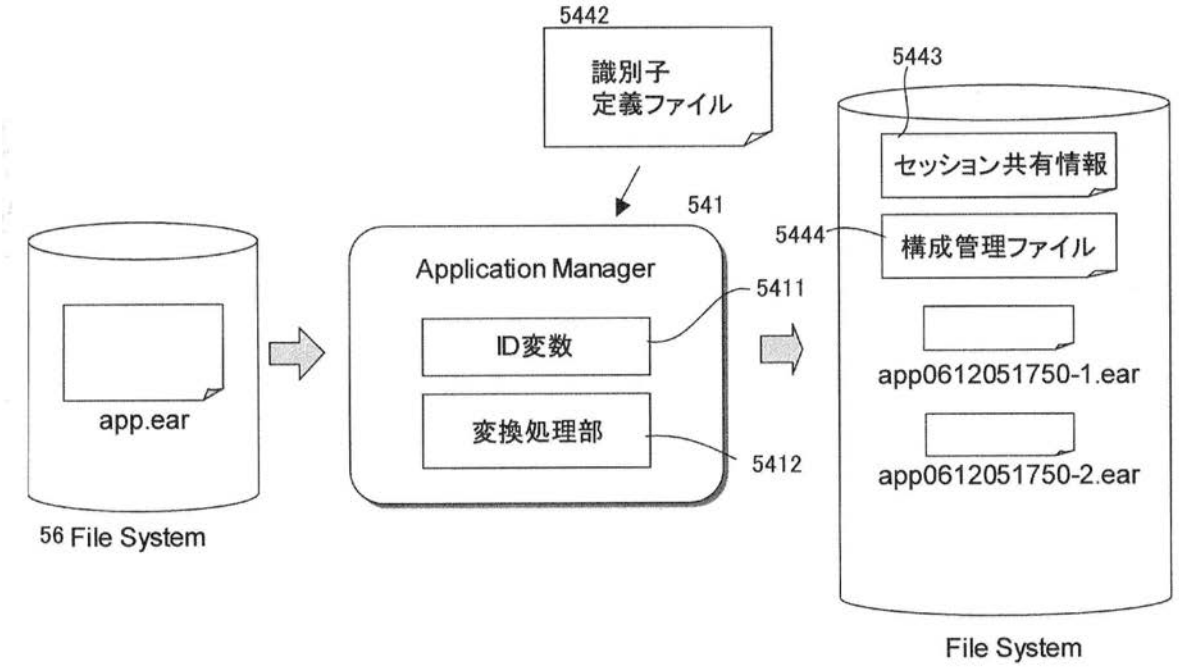
【図１１】


5441
Request
Switch
雛形
5442
識別子
定義ファイル
5443
セッション共有情報
5444
構成管理ファイル
541
Application Manager
5411
ID変数
変換処理部
5412
app.ear
56 File System
app0612051750-1.ear
app0612051750-2.ear
app06120550-rs.war
56 File System

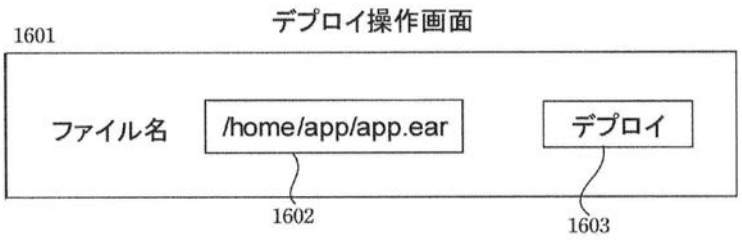
【図３１】


54
Application Server
App1
App2
544A
Request Switch
546
543
Replace Manager
State Store
Deployer
548
541
Application Manager
Nデプロイア
Deploy Tool
547
545
56
App2.ear
App.ear
App1.ear
File System

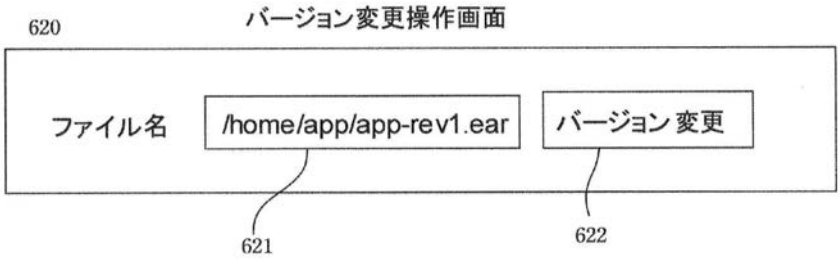
【図３２】


5442
識別子
定義ファイル
5443
セッション共有情報
541
5444
構成管理ファイル
Application Manager
5411
ID変数
app0612051750-1.ear
変換処理部
5412
app0612051750-2.ear
app.ear
56 File System
File System

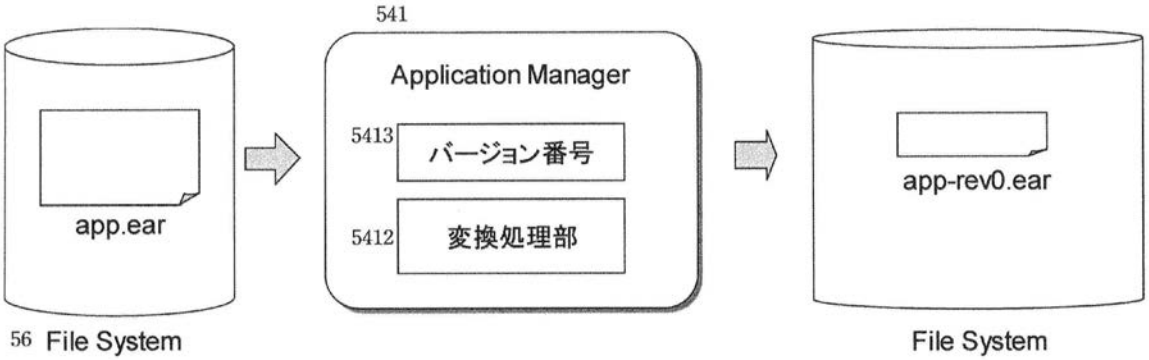
【図３３】


デプロイ操作画面
1601
ファイル名
/home/app/app.ear
デプロイ
1602
1603

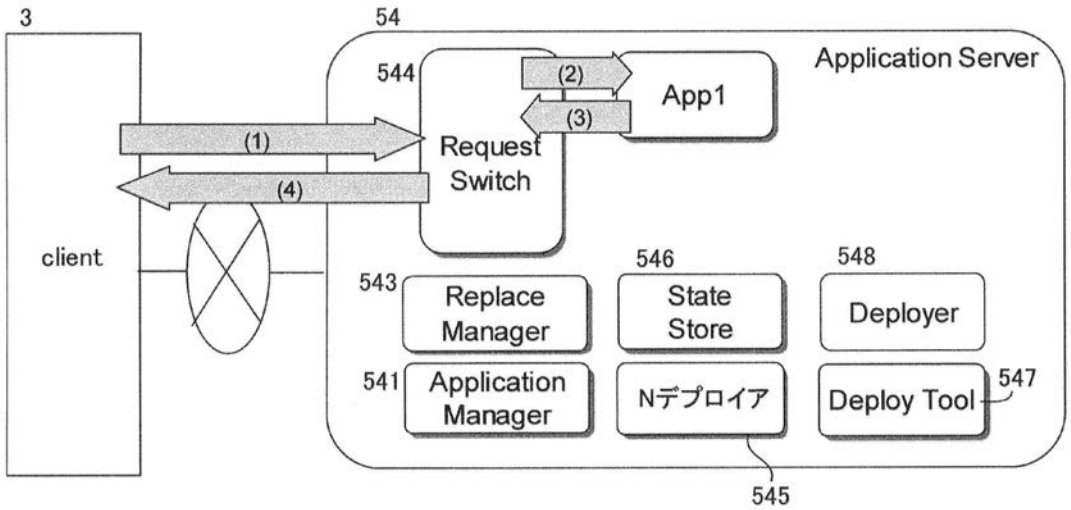
【図３４】


620
バージョン変更操作画面
ファイル名
/home/app/app-rev1.ear
バージョン変更
621
622

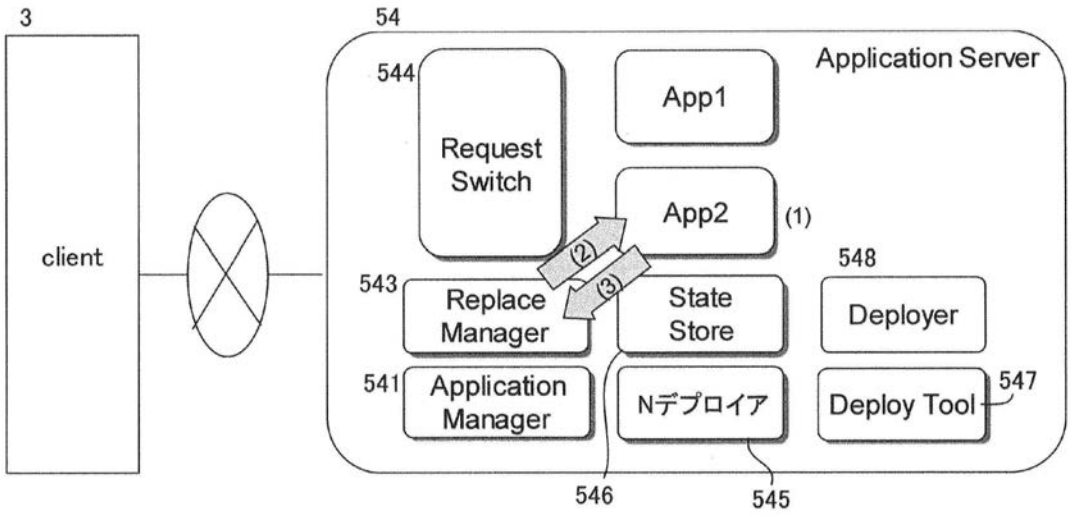
【図３５】


541
Application Manager
5413
バージョン番号
5412
変換処理部
app.ear
app-rev0.ear
56 File System
File System

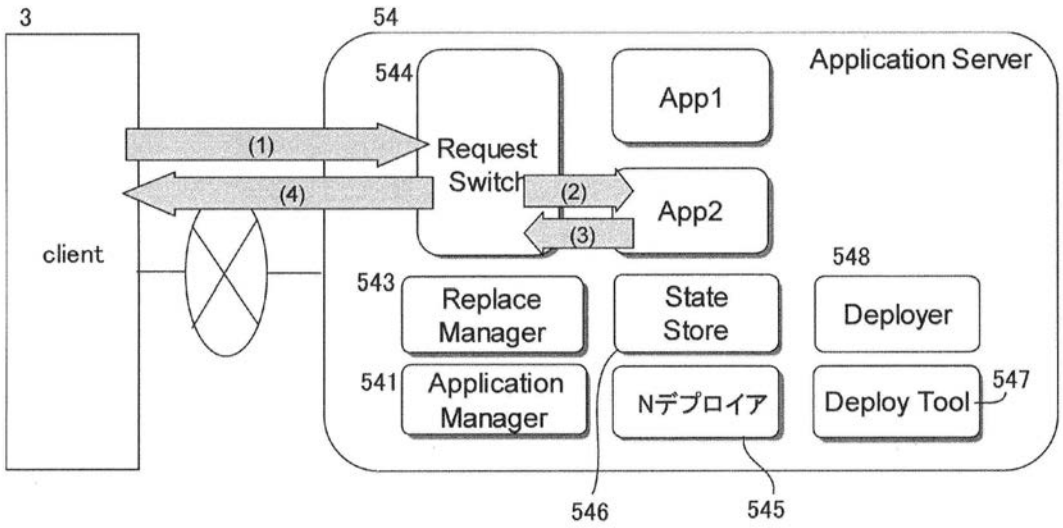
【図３８】


3
54
Application Server
544
(2)
App1
(3)
(1)
Request
Switch
(4)
client
543
Replace
Manager
546
State
Store
548
Deployer
541
Application
Manager
Nデプロイア
Deploy Tool
547
545

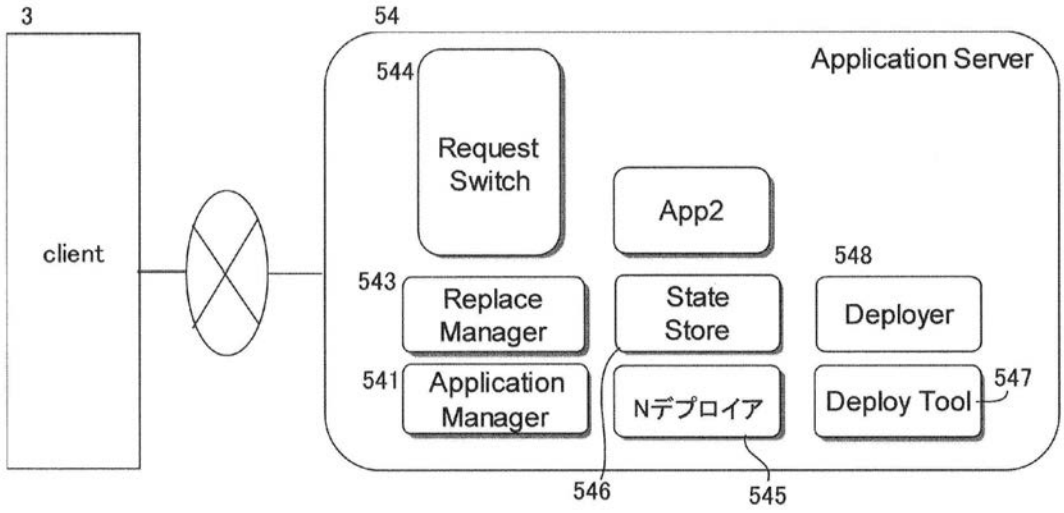
【図３９】


3
client
54
Application Server
544
Request Switch
App1
App2
(1)
(2)
(3)
543
Replace Manager
State Store
548
Deployer
541
Application Manager
Nデプロイア
Deploy Tool
547
546
545

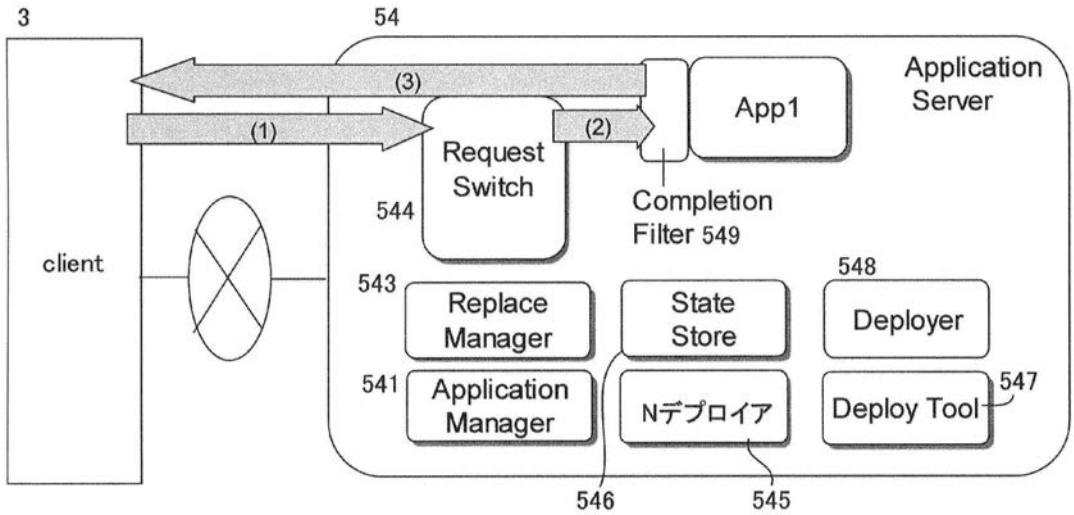
【図４０】


3
client
54
Application Server
544
Request Switch
App1
App2
(1)
(2)
(3)
(4)
543
Replace Manager
State Store
548
Deployer
541
Application Manager
Nデプロイア
Deploy Tool
547
546
545

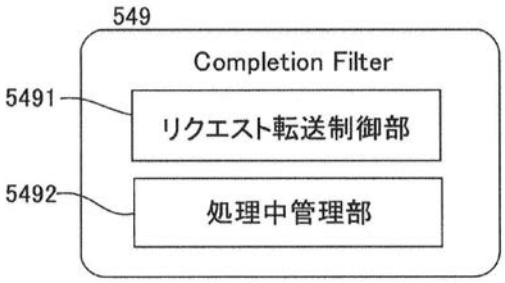
【図４１】


3
54
Application Server
544
Request Switch
App2
client
548
543
Replace Manager
State Store
Deployer
541
Application Manager
Nデプロイア
Deploy Tool
547
546
545

【図４２】


3
54
Application Server
(3)
App1
(1)
(2)
Request Switch
544
Completion Filter 549
client
548
543
Replace Manager
State Store
Deployer
541
Application Manager
Nデプロイア
Deploy Tool
547
546
545

【図４３】



フロントページの続き

(72)発明者　馬場　恒彦
東京都国分寺市東恋ヶ窪一丁目２８０番地　株式会社日立製作所　中央研究所内

審査官　井上　宏一

(56)参考文献　特開２００５－２６７３１２（ＪＰ，Ａ）
特開２００６－３３８０６９（ＪＰ，Ａ）
特開２００６－２７７０４７（ＪＰ，Ａ）
特開２００５－２０２５７３（ＪＰ，Ａ）
特開２０００－１３７６２０（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　　９／４６　－９／５４
Ｇ０６Ｆ　１１／００