(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号
特許第7311710号
(P7311710)

(45)発行日　令和5年7月19日(2023.7.19)

(24)登録日　令和5年7月10日(2023.7.10)

(51)国際特許分類

		FI	
Ｇ０６Ｆ	*8/60 (2018.01)*	Ｇ０６Ｆ	8/60
Ｇ０６Ｆ	*9/4401(2018.01)*	Ｇ０６Ｆ	9/4401

請求項の数　21　（全26頁）

(21)出願番号　特願2022-526203(P2022-526203)
(86)(22)出願日　令和2年11月6日(2020.11.6)
(65)公表番号　特表2022-553860(P2022-553860A)
(43)公表日　令和4年12月26日(2022.12.26)
(86)国際出願番号　PCT/US2020/059555
(87)国際公開番号　WO2021/092502
(87)国際公開日　令和3年5月14日(2021.5.14)
審査請求日　令和4年8月9日(2022.8.9)
(31)優先権主張番号　16/677,805
(32)優先日　令和1年11月8日(2019.11.8)
(33)優先権主張国・地域又は機関
米国(US)

(73)特許権者　502208397
グーグル エルエルシー
Ｇｏｏｇｌｅ ＬＬＣ
アメリカ合衆国 カリフォルニア州 ９４０４３ マウンテン ビュー アンフィシアター パークウェイ １６００
１６００ Ａｍｐｈｉｔｈｅａｔｒｅ Ｐａｒｋｗａｙ ９４０４３ Ｍｏｕｎｔａｉｎ Ｖｉｅｗ， ＣＡ Ｕ．Ｓ．Ａ．
(74)代理人　110001195
弁理士法人深見特許事務所
(72)発明者　クレーマー，ドミニク
アメリカ合衆国、９４０４３ カリフォルニア州、マウンテン・ビュー、アンフィシアター・パークウェイ、１６００

最終頁に続く

(54)【発明の名称】　実行時コンテナ

(57)【特許請求の範囲】

【請求項１】

方法（４００）であって、

データ処理ハードウェア（１４４）において、データストア（１５０）からブートストラップ実行環境（３１０）を受信することを含み、前記ブートストラップ実行環境（３１０）はソフトウェアアプリケーション（２００）を含み、前記ソフトウェアアプリケーション（２００）は依存関係のマニフェスト（２１２）を含み、前記方法はさらに、

前記データ処理ハードウェア（１４４）が、前記ブートストラップ実行環境（３１０）を実行し、前記データ処理ハードウェア（１４４）に、動作を実行させることを含み、前記動作は、

拡張された実行環境（３２０）が前記データストア（１５０）から利用可能であるかどうかを判断することと、

前記拡張された実行環境（３２０）が前記データストア（１５０）から利用可能である場合、

前記拡張された実行環境（３２０）を前記データストア（１５０）から受信することと、

前記拡張された実行環境（３２０）に基づいて前記ブートストラップ実行環境（３１０）を拡張することと、

前記ソフトウェアアプリケーション（２００）を実行することと、

前記拡張された実行環境（３２０）が前記データストア（１５０）から利用可能ではない場合、

前記依存関係のマニフェスト（２１２）に基づいて前記ブートストラップ実行環境（３１０）を拡張して、前記拡張された実行環境（３２０）を作成することと、
前記拡張された実行環境（３２０）を前記データストア（１５０）に記憶することと、
前記ソフトウェアアプリケーション（２００）を実行することとを含む、方法。
【請求項２】
前記データ処理ハードウェア（１４４）が、前記データストア（１５０）内の前記拡張された実行環境（３２０）に注釈を付けて、前記ソフトウェアアプリケーション（２００）のバージョン、前記拡張された実行環境（３２０）に関連付けられるオペレーティングシステムバージョン、または前記拡張された実行環境（３２０）に関連付けられるプロセッサアーキテクチャのうちの少なくとも１つを示すことをさらに含む、請求項１に記載の方法（４００）。
【請求項３】
前記拡張された実行環境（３２０）が前記データストア（１５０）から利用可能であるかどうかを判断することは、前記拡張された実行環境（３２０）が、前記ソフトウェアアプリケーション（２００）、前記ブートストラップ実行環境（３１０）に関連付けられるオペレーティングシステムバージョン、または前記ブートストラップ実行環境（３１０）に関連付けられるプロセッサアーキテクチャのうちの少なくとも１つと互換性があるかどうかを判断することを含む、請求項１～２のいずれか１項に記載の方法（４００）。
【請求項４】
さらに、
前記データ処理ハードウェア（１４４）において、前記ソフトウェアアプリケーション（２００）のための構築要求（１８０）を受信することと、
前記データ処理ハードウェア（１４４）が、前記ソフトウェアアプリケーション（２００）に基づいて前記ブートストラップ実行環境（３１０）を構成することと、
前記データ処理ハードウェア（１４４）が、前記ブートストラップ実行環境（３１０）を前記データストア（１５０）に記憶することとを含む、請求項１～３のいずれか１項に記載の方法（４００）。
【請求項５】
前記依存関係のマニフェスト（２１２）に基づいて前記ブートストラップ実行環境（３１０）を拡張して、前記拡張された実行環境（３２０）を作成することの後に、更新されたファイルのセットを識別することをさらに含み、前記更新されたファイルのセットは、前記依存関係のマニフェスト（２１２）に基づいて前記ブートストラップ実行環境（３１０）を拡張することによって追加または修正されたファイルを含む、請求項１～４のいずれか１項に記載の方法（４００）。
【請求項６】
前記拡張された実行環境（３２０）を前記データストア（１５０）に記憶することは、前記更新されたファイルのセットを前記データストア（１５０）に記憶することを含む、請求項５に記載の方法（４００）。
【請求項７】
前記ブートストラップ実行環境（３１０）はコンテナイメージを含み、前記データストア（１５０）はコンテナレジストリを含む、請求項１～６のいずれか１項に記載の方法（４００）。
【請求項８】
前記依存関係のマニフェスト（２１２）に基づいて前記ブートストラップ実行環境（３１０）を拡張することは、アプリケーション依存関係（２１２）をインストールすることを含む、請求項１～７のいずれか１項に記載の方法（４００）。
【請求項９】
前記アプリケーション依存関係（２１２）は、サポートライブラリ、アーキテクチャ固有のバイナリモジュール、または実行時にコンパイルされるモジュールのうちの少なくとも１つを含む、請求項８に記載の方法（４００）。

【請求項１０】
前記ソフトウェアアプリケーション（２００）は、JavaScriptアプリケーション（２００）、Pythonアプリケーション（２００）、またはRubyアプリケーション（２００）のうちの１つである、請求項１～９のいずれか１項に記載の方法（４００）。
【請求項１１】
システム（１００）であって、
データ処理ハードウェア（１４４）と、
前記データ処理ハードウェア（１４４）と通信するメモリハードウェア（１４６）とを備え、前記メモリハードウェア（１４６）は、前記データ処理ハードウェア（１４４）上で実行されると前記データ処理ハードウェア（１４４）に動作を実行させる命令を記憶し、前記動作は、
データストア（１５０）からブートストラップ実行環境（３１０）を受信することを含み、前記ブートストラップ実行環境（３１０）はソフトウェアアプリケーション（２００）を含み、前記ソフトウェアアプリケーション（２００）は依存関係のマニフェスト（２１２）を含み、前記動作はさらに、
前記ブートストラップ実行環境（３１０）を実行し、前記データ処理ハードウェア（１４４）にさらなる動作を実行させることを含み、前記さらなる動作は、
拡張された実行環境（３２０）が前記データストア（１５０）から利用可能であるかどうかを判断することと、
前記拡張された実行環境（３２０）が前記データストア（１５０）から利用可能である場合、
前記拡張された実行環境（３２０）を前記データストア（１５０）から受信することと、
前記拡張された実行環境（３２０）に基づいて前記ブートストラップ実行環境（３１０）を拡張することと、
前記ソフトウェアアプリケーション（２００）を実行することと、
前記拡張された実行環境（３２０）が前記データストア（１５０）から利用可能ではない場合、
前記依存関係のマニフェスト（２１２）に基づいて前記ブートストラップ実行環境（３１０）を拡張して、前記拡張された実行環境（３２０）を作成することと、
前記拡張された実行環境（３２０）を前記データストア（１５０）に記憶することと、
前記ソフトウェアアプリケーション（２００）を実行することとを含む、システム。
【請求項１２】
前記動作は、さらに、前記データストア（１５０）内の前記拡張された実行環境（３２０）に注釈を付けて、前記ソフトウェアアプリケーション（２００）のバージョン、前記拡張された実行環境（３２０）に関連付けられるオペレーティングシステム（１００）バージョン、または前記拡張された実行環境（３２０）に関連付けられるプロセッサ（５１０）アーキテクチャのうちの少なくとも１つを示すことを含む、請求項１１に記載のシステム（１００）。
【請求項１３】
前記拡張された実行環境（３２０）が前記データストア（１５０）から利用可能であるかどうかを判断することは、前記拡張された実行環境（３２０）が、前記ソフトウェアアプリケーション（２００）、前記ブートストラップ実行環境（３１０）に関連付けられるオペレーティングシステム（１００）バージョン、または前記ブートストラップ実行環境（３１０）に関連付けられるプロセッサ（５１０）アーキテクチャのうちの少なくとも１つと互換性があるかどうかを判断することを含む、請求項１１～１２のいずれか１項に記載のシステム（１００）。
【請求項１４】
前記動作はさらに、
前記ソフトウェアアプリケーション（２００）のための構築要求（１８０）を受信することと、

前記ソフトウェアアプリケーション（２００）に基づいて前記ブートストラップ実行環境（３１０）を構成することと、
前記ブートストラップ実行環境（３１０）を前記データストア（１５０）に記憶することとを含む、請求項１１～１３のいずれか１項に記載のシステム（１００）。
【請求項１５】
前記動作はさらに、前記依存関係のマニフェスト（２１２）に基づいて前記ブートストラップ実行環境（３１０）を拡張して、前記拡張された実行環境（３２０）を作成することの後に、更新されたファイルのセットを識別することを含み、前記更新されたファイルのセットは、前記依存関係のマニフェスト（２１２）に基づいて前記ブートストラップ実行環境（３１０）を拡張することによって追加または修正されたファイルを含む、請求項１１～１４のいずれか１項に記載のシステム（１００）。
【請求項１６】
前記拡張された実行環境（３２０）を前記データストア（１５０）に記憶することは、前記更新されたファイルのセットを前記データストア（１５０）に記憶することを含む、請求項１５に記載のシステム（１００）。
【請求項１７】
前記ブートストラップ実行環境（３１０）はコンテナイメージを含み、前記データストア（１５０）はコンテナレジストリを含む、請求項１１～１６のいずれか１項に記載のシステム（１００）。
【請求項１８】
前記依存関係のマニフェスト（２１２）に基づいて前記ブートストラップ実行環境（３１０）を拡張することは、アプリケーション依存関係（２１２）をインストールすることを含む、請求項１１～１７のいずれか１項に記載のシステム（１００）。
【請求項１９】
前記アプリケーション依存関係（２１２）は、サポートライブラリ、アーキテクチャ固有のバイナリモジュール、または実行時にコンパイルされるモジュールのうちの少なくとも１つを含む、請求項１８に記載のシステム（１００）。
【請求項２０】
前記ソフトウェアアプリケーション（２００）は、JavaScriptアプリケーション（２００）、Pythonアプリケーション（２００）、またはRubyアプリケーション（２００）のうちの１つである、請求項１１～１９のいずれか１項に記載のシステム（１００）。
【請求項２１】
データ処理ハードウェア（１４４）に請求項１～１０のいずれか１項に記載の方法を実行させるプログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
技術分野
本開示は、ソフトウェアアプリケーションのための実行環境を効率的に構成および展開することに関する。
【背景技術】
【０００２】
背景
コンテナ技術は、従来の仮想化環境に関連付けられる大きなオーバーヘッドを招くことなく、アプリケーションおよびサービスを迅速にスケーリングする見込みを提供する。ソフトウェアアプリケーションのためのソースコードは、コンテナ内での動作展開に先立って、ソフトウェア開発環境内で開発および試験されてもよい。コンテナビルダは、入力されるソースコードから出力されるコンテナを構築するために公然利用可能である。これらのコンテナビルダは、概して、ソースコードを展開のために対応するコンテナにどのようにパッケージ化するかを記述する命令および構築ツールを含む。コンテナビルダは、しば

しば、ソフトウェア開発キット、コンパイラ、および／またはデバッガなどの構築時ツール、ならびに出力されるコンテナを実行するためのランタイム環境の両方を含む、重いコンテナイメージを生成する。これらの重いコンテナイメージはより大きく、展開および顧客への配信時に不要なコンテンツ／コンポーネントを含む。例えば、展開されたコンテナにコンパイラを含めることは、コンテナに重量を追加するだけでなく、パッケージ化された展開に攻撃ベクトルおよびセキュリティ脆弱性を導入するので、不必要である。より軽量な実行環境は、アプリケーションまたはサービスをサポートするために必要なコンポーネント／コンテンツのみを含む。しかしながら、実行環境のこの合理化は、構成負担の増加という犠牲を払って成り得る。ソフトウェア開発プラットフォームは、アプリケーション実行環境を構成および展開することに対しては最適化されていない場合がある。さらに、ソフトウェア開発者は、軽量実行環境を構成および展開するのに熟練もしくは知識がないことがあり、または実行環境を構成もしくは展開する権限がないことがある。その結果、軽量実行環境においてアプリケーションを展開することを望むソフトウェア開発者は、非効率に直面する。

【発明の概要】

【課題を解決するための手段】

【０００３】

概要

本開示の態様は、ソフトウェアアプリケーションのための実行環境を構成および展開することに関する。一態様は、データ処理ハードウェアにおいて、データストアからブートストラップ実行環境を受信することを含む方法を提供し、ブートストラップ実行環境は、依存関係のマニフェストを有するソフトウェアアプリケーションを含む。本方法は、データ処理ハードウェアがブートストラップ実行環境を実行し、拡張された実行環境がデータストアから利用可能であるかどうかを判断することを含む動作をデータ処理ハードウェアに実行させることをさらに含む。拡張された実行環境がデータストアから利用可能である場合、動作は、データストアから拡張された実行環境を受信することと、拡張された実行環境に基づいてブートストラップ実行環境を拡張することと、ソフトウェアアプリケーションを実行することとをさらに含む。拡張された実行環境がデータストアから利用可能ではない場合、動作は、依存関係のマニフェストに基づいてブートストラップ実行環境を拡張して、拡張された実行環境を作成することと、拡張された実行環境をデータストアに記憶することと、ソフトウェアアプリケーションを実行することとを含む。

【０００４】

本開示の実現例は、以下の任意選択の特徴のうちの１つ以上を含んでもよい。いくつかの実現例では、本方法は、データ処理ハードウェアが、データストア内の拡張された実行環境に注釈を付けて、ソフトウェアアプリケーションのバージョン、拡張された実行環境に関連付けられるオペレーティングシステムバージョン、または拡張された実行環境に関連付けられるプロセッサアーキテクチャのうちの少なくとも１つを示すことを含む。拡張された実行環境がデータストアから利用可能であるかどうかを判断することは、拡張された実行環境が、ソフトウェアアプリケーション、ブートストラップ実行環境に関連付けられるオペレーティングシステムバージョン、またはブートストラップ実行環境に関連付けられるプロセッサアーキテクチャのうちの少なくとも１つと互換性があるかどうかを判断することを含んでもよい。

【０００５】

本方法は、データ処理ハードウェアにおいて、ソフトウェアアプリケーションのための構築要求を受信することと、データ処理ハードウェアが、ソフトウェアアプリケーションに基づいてブートストラップ実行環境を構成することと、データ処理ハードウェアが、ブートストラップ実行環境をデータストアに記憶することとをさらに含んでもよい。依存関係のマニフェストに基づいてブートストラップ実行環境を拡張することは、アプリケーション依存関係をインストールすることを含んでもよい。いくつかの例では、ソフトウェアアプリケーションは、JavaScriptアプリケーション、Pythonアプリケーション、またはR

ubyアプリケーションのうちの１つである。アプリケーション依存関係は、サポートライブラリ、アーキテクチャ固有のバイナリモジュール、または実行時にコンパイルされるモジュールのうちの少なくとも１つを含んでもよい。いくつかの実現例では、本方法は、依存関係のマニフェストに基づいてブートストラップ実行環境を拡張して、拡張された実行環境を作成した後、依存関係のマニフェストに基づいてブートストラップ実行環境を拡張することによって追加または修正されたファイルを含む更新されたファイルのセットを識別することを含む。拡張された実行環境をデータストアに記憶することは、更新されたファイルのセットをデータストアに記憶することを含んでもよい。いくつかの例では、ブートストラップ実行環境はコンテナイメージを含み、データストアはコンテナレジストリを含む。

【０００６】

本開示の別の態様は、ソフトウェアアプリケーションのための実行環境を構成し、展開するためのシステムを提供する。本システムは、データ処理ハードウェアと、データ処理ハードウェアと通信するメモリハードウェアとを備える。メモリハードウェアは、データ処理ハードウェア上で実行されるとデータ処理ハードウェアに動作を実行させる命令を記憶する。動作は、データストアからブートストラップ実行環境を受信することを含み、ブートストラップ実行環境は、依存関係のマニフェストを有するソフトウェアアプリケーションを含む。動作は、ブートストラップ実行環境を実行し、拡張された実行環境がデータストアから利用可能であるかどうかを判断することを含むさらなる動作をデータ処理ハードウェアに実行させることをさらに含む。拡張された実行環境がデータストアから利用可能である場合、動作は、データストアから拡張された実行環境を受信することと、拡張された実行環境に基づいてブートストラップ実行環境を拡張することと、ソフトウェアアプリケーションを実行することとをさらに含む。拡張された実行環境がデータストアから利用可能ではない場合、動作は、依存関係のマニフェストに基づいてブートストラップ実行環境を拡張して、拡張された実行環境を作成することと、拡張された実行環境をデータストアに記憶することと、ソフトウェアアプリケーションを実行することとを含む。

【０００７】

本開示の実現例は、以下の任意選択の特徴のうちの１つ以上を含んでもよい。いくつかの実現例では、動作は、データストア内の拡張された実行環境に注釈を付けて、ソフトウェアアプリケーションのバージョン、拡張された実行環境に関連付けられるオペレーティングシステムバージョン、または拡張された実行環境に関連付けられるプロセッサアーキテクチャのうちの少なくとも１つを示すことをさらに含む。拡張された実行環境がデータストアから利用可能であるかどうかを判断することは、拡張された実行環境が、ソフトウェアアプリケーション、ブートストラップ実行環境に関連付けられるオペレーティングシステムバージョン、またはブートストラップ実行環境に関連付けられるプロセッサアーキテクチャのうちの少なくとも１つと互換性があるかどうかを判断することを含んでもよい。

【０００８】

動作は、ソフトウェアアプリケーションのための構築要求を受信することと、ソフトウェアアプリケーションに基づいてブートストラップ実行環境を構成することと、ブートストラップ実行環境をデータストアに記憶することとをさらに含んでもよい。依存関係のマニフェストに基づいてブートストラップ実行環境を拡張することは、アプリケーション依存関係をインストールすることを含んでもよい。いくつかの例では、ソフトウェアアプリケーションは、JavaScriptアプリケーション、Pythonアプリケーション、またはRubyアプリケーションのうちの１つである。アプリケーション依存関係は、サポートライブラリ、アーキテクチャ固有のバイナリモジュール、または実行時にコンパイルされるモジュールのうちの少なくとも１つを含んでもよい。いくつかの実現例では、動作は、依存関係のマニフェストに基づいてブートストラップ実行環境を拡張して、拡張された実行環境を作成した後、依存関係のマニフェストに基づいてブートストラップ実行環境を拡張することによって追加または修正されたファイルを含む更新されたファイルのセットを識別することを含む。拡張された実行環境をデータストアに記憶することは、更新されたファイルの

セットをデータストアに記憶することを含んでもよい。いくつかの例では、ブートストラップ実行環境はコンテナイメージを含み、データストアはコンテナレジストリを含む。

【０００９】

本開示の１つ以上の実現例の詳細を、添付の図面および以下の説明に記載する。他の態様、特徴、および利点は、説明および図面、ならびに特許請求の範囲から明らかになるであろう。

【図面の簡単な説明】

【００１０】

【図１】アプリケーション実行環境を構成し展開するための例示的な環境の概略図である。

【図２Ａ】ソフトウェアアプリケーションおよび対応の依存関係のマニフェストの概略図である。

【図２Ｂ】ソフトウェアアプリケーションおよび対応の依存関係のマニフェストの概略図である。

【図２Ｃ】ソフトウェアアプリケーションおよび対応の依存関係のマニフェストの概略図である。

【図３Ａ】ソフトウェアアプリケーションを実行するための実行環境を構築する概略図である。

【図３Ｂ】ソフトウェアアプリケーションを実行するための実行環境を構築する概略図である。

【図３Ｃ】ソフトウェアアプリケーションを実行するための実行環境を構築する概略図である。

【図３Ｄ】ソフトウェアアプリケーションを実行するための実行環境を構築する概略図である。

【図３Ｅ】ソフトウェアアプリケーションを実行するための実行環境を構築する概略図である。

【図４】ソフトウェアアプリケーションのための実行環境を構成および展開する方法のための動作の例示的な構成のフローチャートである。

【図５】例示的なコンピューティングデバイスの概略図である。

【発明を実施するための形態】

【００１１】

様々な図面における同様の参照符号は、同様の要素を示す。

詳細な説明

ソフトウェアアプリケーションは、ソフトウェア開発環境において開発された後に、アプリケーション実行環境において動作可能に展開される場合がある。コンテナは、ソフトウェアアプリケーションを展開するための好ましいアプリケーション実行環境として現れつつある。コンテナ技術は、従来の仮想化環境に関連付けられる大きなオーバーヘッドを招くことなく、アプリケーションおよびサービスを迅速にスケーリングする見込みを提供する。複数のコンテナ化されたアプリケーションは、単一のホストコンピュータ上で実行され、同じ基底のオペレーティングシステムにアクセスしてもよい。さらに、コンテナ化されたアプリケーションは、様々な基底のオペレーティングシステムおよび／またはコンピュータもしくはプロセッサアーキテクチャ上で実行されてもよい。コンテナビルダは、入力ソースコードからコンテナイメージを構築するために公然利用可能である。これらのコンテナビルダは、典型的には、ソフトウェア開発キット、コンパイラ、および／またはデバッガ等の構築時ツール、ならびにソフトウェアアプリケーションを実行するために必要とされる実行時環境を含む、重いコンテナイメージを生成する。これらの重いコンテナイメージは大きく、展開／配信されるときに不要なコンテンツ／コンポーネントを含むことが多い。例えば、展開されたコンテナにコンパイラを含めることは、コンテナに重量を追加し、パッケージ化された展開に攻撃ベクトルおよびセキュリティ脆弱性を導入するので、不必要である。コンテナイメージを構成および展開することは、時間、労力、および技術を必要とする。コンテナイメージは、ソフトウェアアプリケーションが更新されるた

びに再構成される必要がある場合がある。頻繁な再構成は、ソフトウェア開発の費用および労力を増大させる。

【００１２】

より軽量な実行環境は、例えばソフトウェア開発コンテンツ／コンポーネントなしで、アプリケーションまたはサービスをサポートするために必要なコンポーネント／コンテンツのみを含む。しかしながら、より軽量な実行環境を構成および展開するためには、特殊な技術または知識が必要とされ得る。構成は、省略されるべきソフトウェア開発コンテンツ／コンポーネントを識別すること、および実行環境に含まれるべき必要な実行時コンテンツ／コンポーネントを識別することを必要とし得る。要するに、アプリケーション実行環境は、ソフトウェア開発環境とは実質的に異なり得る。例えば、ソフトウェア開発環境は、ウェブブラウザ、電子メールクライアント、修正制御システム、および／またはソフトウェア開発を容易にするために最適化されたグラフィカルディスプレイを含み得る。しかしながら、アプリケーション実行環境は、展開されたアプリケーションの、信頼できる効率的な実行のために、最適化され得る。ソフトウェア開発者は、軽量実行環境を構成および展開するのに熟練もしくは知識がないことがあり、または実行環境を構成もしくは展開する権限がないことがある。

【００１３】

コンテナビルダは、コンテナ構成最良実践の深い習熟を必要とすることなく汎用コンテナイメージを構築するために公然利用可能である。しかしながら、これらのコンテナビルダは、静的イメージからコンテナを構築する。すなわち、静的コンテナイメージは、アプリケーションの初期または「ブートストラップ」実行のための環境を含む。この手法は、アーキテクチャに依存しないソフトウェアアプリケーションまたはアーキテクチャついて互換性のあるソフトウェアアプリケーションに対しては充分であるかもしれないが、ソフトウェアアプリケーションは、それらの実行環境をカスタマイズまたは拡張した後、より効率的に実行される場合がある。例えば、ソフトウェアアプリケーションは、アーキテクチャ特有のバイナリモジュール、サポートライブラリ、もしくは他のファイル等のコンテンツを動的にインストールしてもよく、または、アプリケーションランタイム環境は、ソースコードまたはバイトコードをアーキテクチャ特有の機械コードにコンパイルして、アプリケーション性能を向上させてもよい。したがって、静的イメージからコンテナを構築する汎用コンテナ構築ツールは、これらの動的実行時拡張を捕らえるかまたはコンテナイメージを更新して実行時アーキテクチャ特有拡張を反映する能力を欠いている。アプリケーションが再開されるたびに、これらの実行時拡張は、再度実行される。コンテナ内で重い重量のコンテナ構成ツールを実行することは、コンテナを構成および／または展開するのに、重量を追加すること、ならびに時間、労力、および技術を必要とすることという犠牲を払って、これらの動的実行時拡張を捕らえることが可能であってもよい。本明細書の実現例は、ソフトウェアアプリケーションの後続の実行において用いるためにソフトウェアアプリケーションによって実行環境に合わせて調整された動的および／またはアーキテクチャ固有の拡張を捕らえることを含む、ソフトウェアアプリケーションのためのセキュアな実行環境（たとえば、コンテナ）を効率的に構成および展開する方法ならびにシステムを対象とする。

【００１４】

図１を参照すると、いくつかの実現例では、システム１００は、例えば、ネットワーク１３０を介して、遠隔システム１４０と通信し得る、ソフトウェア開発者１０と関連付けられる開発者デバイス１１０、１１０ａ－ｎを含む。遠隔システム１４０は、スケーラブル／弾性リソース１４２を有する分散型システム（例えば、クラウド環境）であってもよい。リソース１４２は、コンピューティングリソース（例えば、データ処理ハードウェア）１４４および／または記憶リソース（例えば、メモリハードウェア）１４６を含む。ソフトウェア開発者１０は、開発者デバイス１１０を用いて、遠隔システム１４０上で実行するためのソフトウェアアプリケーション２００を開発することができる。遠隔システム１４０はまた、ソフトウェアアプリケーション２００が開発者１０によってどのように開

発／作成されたかとは無関係に、遠隔システム１４０上でソフトウェアアプリケーション２００を実行するための１つ以上のセキュアな実行環境３１０、３２０（たとえば、コンテナ）を構築するためのコンテナビルダ３００を実行する。ソフトウェアアプリケーション２００を実行するためにコンテナを構築するための命令に関連付けられる構成を開発者１０が提供することを要求する汎用コンテナ構築ツール（例えば、ドッカー）とは対照的に、コンテナビルダ３００は、構成段階３０１中に開発者デバイス１１０から受信されたソフトウェアアプリケーション２００に基づいてブートストラップ実行環境３１０（たとえば、ブートストラップコンテナ３１０）を自動的に構成するように構成される。ここで、ブートストラップ実行環境３１０は、静的コンテンツを有する静的コンテナイメージに関連付けられる。特に、汎用コンテナ構築ツール（例えば、ドッカー）で使用されるワークフローとは対照的に、構成段階３０１は、動的コンテンツを作成するためのコマンドを実行することなく、ブートストラップ実行環境３１０に関連付けられる静的イメージを構成する。ブートストラップ実行環境３１０は、実行環境３１０、３２０を構築または構成するためのツーリング３１６を含んでもよい。いくつかの実現例では、ツーリング３１６は、コンテナを作成、構成、および管理するためのコマンドラインツールのセットである。遠隔および／またはウェブベースのサービスを含む、他の形態のツーリング３１６が可能である。いくつかの実現例では、コンテナビルダ３００は、起動命令のセット３１８を含むようにブートストラップ実行環境３１０を構成する。起動命令３１８は、ブートストラップ実行環境３１０を実行するときにコンテナビルダ３００が取るステップを定義する。ツーリング３１６および起動命令３１８の使用は、図３Ａ～図３Ｅを参照して以下でより詳細に説明される。

【００１５】

その後、実行段階３０２中に、コンテナビルダ３００は、ブートストラップ実行環境３１０を走らせて／実行して、拡張された実行環境３２０を動的コンテンツ（例えば、アプリケーション依存関係３２２）とともに構築／作成するように構成される。具体的には、コンテナビルダ３００は、動的コンテンツ（例えば、アプリケーション依存関係３２２）でブートストラップ実行環境３１０を拡張することによって、ソフトウェアアプリケーション２００のための拡張された実行環境３２０を構築する。すなわち、実行段階３０２中に、コンテナビルダ３００は、ブートストラップ実行環境３１０内でソフトウェアアプリケーション２００を実行する。コンテナビルダ３００および／またはソフトウェアアプリケーション２００は、ソフトウェアアプリケーション２００の依存関係のマニフェスト２１２に基づいて動的コンテンツ（たとえば、アプリケーション依存関係３２２）を作成し、それによって、ブートストラップ実行環境３１０を動的コンテンツ３２２で拡張して、拡張された実行環境３２０を作成してもよい。その結果、コンテナビルダ３００は、コンテナを構築するために静的および動的コンテンツがイメージにどのように記述されるべきかに関する命令を提供する構成（例えば、ドッカーファイル）を提供することを開発者１０に要求することなく、遠隔システム１４０上でソフトウェアアプリケーション２００を実行するための、動的コンテンツ３２２を含む、拡張された実行環境３２０を、開発者１０が構築し、および任意選択肢的に展開することを可能にする。

【００１６】

いくつかの実現例では、遠隔システム１４０は、メモリハードウェア１４６、５２０（図５）内に１つ以上のブートストラップ実行環境３１０、３１０ａ～ｎと１つ以上の拡張された実行環境３２０、３２０ａ～ｎとを記憶するデータストア１５０を含む。いくつかの例では、データストア１５０はコンテナレジストリを含み、各実行環境３１０、３２０は、それぞれのコンテナ化されたソフトウェアアプリケーション２００に関連付けられる。これらの例において、ブートストラップ実行環境３１０は、静的コンテンツ（たとえば、依存関係のマニフェスト２１２）を有する静的イメージに関連付けられる「ブートストラップコンテナ」と称されてもよく、拡張された実行環境３２０は、ブートストラップ実行環境３１０の実行中に（例えば実行段階３０２において）動的に作成されるコンテンツ３２２（たとえば、アプリケーション依存関係）を有するコンテナイメージに関連付けら

れる「拡張されたコンテナ」と称されてもよい。各実行環境３１０、３２０は、実行環境３１０、３２０に関連付けられるソフトウェアアプリケーション２００を識別する識別子３１４（例えば、タグ）を含んでもよい。拡張された実行環境３２０はまた、拡張された実行環境３２０を構築した遠隔システム１４０のオペレーティングシステムバージョンおよび／またはプロセッサアーキテクチャなどの、拡張された実行環境３２０の態様を示すメタデータ３２４を含んでもよい。

【００１７】

図示の例では、コンテナビルダ３００は、開発者デバイス１１０から、開発者１０によってローカルに開発されたソフトウェアアプリケーション２００のための実行環境３１０、３２０を構築するよう、構築要求１８０を受信する。構築要求１８０は、単に、依存関係のマニフェスト２１２を含むソフトウェアアプリケーション２００を含んでもよい。構築要求１８０はまた、ソフトウェアアプリケーション２００に加えて、またはその代わりに、ソフトウェアアプリケーション２００を識別する識別子３１４（例えば、タグ）を含んでもよく、コンテナビルダ３００は、識別子３１４を用いて、データストア１５０からソフトウェアアプリケーション２００に関連付けられるブートストラップ実行環境３１０を取り出してもよい。ここで、ブートストラップ実行環境３１０は、コンテンツに関する情報のみを含む静的イメージに関連付けられている。

【００１８】

以下でより詳細に説明されるように、ブートストラップ実行環境３１０を受信した後、コンテナビルダ３００は、ブートストラップ実行環境３１０を実行し、拡張された実行環境３２０がデータストア１５０から利用可能であるかどうか、例えば、それが以前に作成され、データストア１５０に記憶されたかどうかを判断してもよい。拡張された実行環境３２０が利用可能である場合、コンテナビルダ３００は、少なくとも、拡張された実行環境３２０に関連付けられるアプリケーション依存関係３２２（たとえば、動的コンテンツ）をデータストア１５０から取り出し、ブートストラップ実行環境３１０の以前の初期実行中に動的に作成されたアプリケーション依存関係３２２を用いてブートストラップ実行環境３１０を拡張してもよい。ブートストラップ実行環境３１０は、実行環境３１０、３２０を構築または構成するためのツーリング３１６を含んでもよい。いくつかの例では、ブートストラップ実行環境３１０は、ツーリング３１６を用いて、拡張された実行環境３２０を構成し、拡張された実行環境３２０をデータストア１５０に記憶する。この筋書では、ブートストラップ実行環境３１０は、拡張された実行環境３２０を作成するためにすでに１回実行されているので、ブートストラップ実行環境３１０を実行する後続のインスタンスは、動的コンテンツ（例えば、アプリケーション依存関係）を作成するための起動コマンドの実行をしないで済ますことになり、なぜならば、動的コンテンツは、データストア１５０から単純に、そしてより効率的に、取り出されることができるからである。

【００１９】

ソフトウェアアプリケーション２００（すなわち、ソフトウェアリソース）という用語は、コンピューティングデバイス（例えば、データ処理ハードウェア１４４、５００（図５））にタスクを実行させるコンピュータソフトウェアを指し得る。したがって、ソフトウェアアプリケーション２００は、分散型システム（例えば、遠隔システム１４０）上にインストール、実行、展開、および／もしくは別様に実現され得る、任意のタイプまたは形態のソフトウェア、ファイル、ならびに／または実行可能コードに対応してもよい。いくつかの例では、ソフトウェアアプリケーション２００は、「アプリケーション」、「アプリ」、または「プログラム」と呼ばれることがある。例示的なアプリケーションは、システム診断アプリケーション、システム管理アプリケーション、システム保守アプリケーション、ワードプロセッシングアプリケーション、スプレッドシートアプリケーション、メッセージングアプリケーション、ウェブブラウザアプリケーション、メディアストリーミングアプリケーション、ソーシャルネットワーキングアプリケーション、セキュリティアプリケーション、およびゲームアプリケーションを含むが、これらに限定されない。ソフトウェアアプリケーション２００は、C/C++、Java（登録商標）、Python、Ruby、P

erl、またはJavaScript（登録商標）などの高レベルおよび構造化プログラミング言語を含む１つ以上のプログラミング言語で表現されてもよく、C/C++アプリケーション、Java（登録商標）アプリケーション、Pythonアプリケーション、Rubyアプリケーション、Perlアプリケーション、またはJavaScript（登録商標）アプリケーションと呼ばれることがある。ソフトウェアアプリケーション２００は、他の形態、フレームワーク、および／またはプログラミング言語でも表現され得る。たとえば、Node.jsフレームワークにおいて開発されたJavaScriptアプリケーションは、Node.jsアプリケーションまたはは、JavaScriptアプリケーションと呼ばれることがある。

【００２０】

いくつかの例では、ソフトウェアアプリケーション２００は、依存関係のマニフェスト２１２またはリストを含む。依存関係のマニフェスト（「依存関係マニフェスト」と相互交換可能に称される）２１２は、遠隔システム１４０上での実行中にソフトウェアアプリケーション２００によって使用またはアクセスされるファイルおよび／もしくはソフトウェアライブラリまたはモジュールを列挙することができる。依存関係マニフェスト２１２は、ソフトウェアアプリケーション２００によって使用されるアーキテクチャ固有および／またはプログラミング言語固有のサポートライブラリを含んでもよい。例えば、依存関係マニフェスト２１２は、PythonプログラミングのためのNumPy科学コンピューティングパッケージを含んでもよい。依存関係のマニフェスト２１２は、遠隔システム１４０上で実行環境３１０、３２０を構成するために、ソフトウェア開発者１０によって明示的に構成されてもよい。いくつかの例では、依存関係のマニフェスト２１２は、ソフトウェアアプリケーション２００によって暗示される。例えば、ソフトウェアアプリケーション２００は、サポートライブラリを要求またはインポートするステートメントを含んでもよい。ソフトウェアアプリケーション２００は、「ＵＲＬ解析支援ライブラリへの依存関係を示すよう要求（「urlib」）」というステートメントを含んでもよい。集合的に、そのようなステートメントは、依存関係のマニフェスト２１２の一部と見なされてもよい。

【００２１】

ブートストラップ実行環境３１０は、ソフトウェアアプリケーション２００が動作環境にかかわらず一貫して実行できるように、ソフトウェアアプリケーション２００を最初に実行するために必要とされるすべて、例えば、コード、ランタイム、システムツール、およびライブラリを含む完全なファイルシステムに、ソフトウェアアプリケーション２００を含んでもよい。いくつかの例では、開発者デバイス１１０ａ～ｎは、ソフトウェアアプリケーション２００に基づいてブートストラップ実行環境３１０を構築する。この場合、遠隔システム１４０は、開発者デバイス１１０ａ～ｎからブートストラップ実行環境３１０を受信し、ブートストラップ実行環境３１０をデータストア１５０に記憶してもよい。これらの例では、開発者デバイス１１０から受信された構築要求１８０は、ブートストラップ実行環境３１０を含んでもよい。データストア１５０に記憶された実行環境３１０、３２０は、後でデータストア１５０から取り出すために、インデックス付けおよび／またはカタログ化および／または識別子３１４を割り当てられることができる。データストア１５０は、「タグ」または他の注釈を用いて、各実行環境３１０、３２０に割り当てられた識別子３１４を表して、その後の検索および取得を容易にしてもよい。識別子３１４は、実行環境に関連付けられるソフトウェアアプリケーション２００を識別してもよく、および／または実行環境３１０、３２０を具体的に識別してもよい。したがって、開発者デバイス１１０ａ～ｎからの構築要求１８０は、ソフトウェアアプリケーション２００および／またはブートストラップ実行環境３１０を識別する１つ以上の識別子３１４を含んでもよい。データストア１５０は、実行環境３１０、３２０またはソフトウェアアプリケーション２００への不正アクセスから保護するための認証などのセキュリティ機能を含んでもよい。

【００２２】

図２Ａ～図２Ｃは、ソフトウェアアプリケーション２００、２００ａ～ｃおよび依存関係のそれぞれのマニフェスト２１２、２１２ａ～ｃの概略図である。図２Ａは、アプリケ

ーション依存関係３２２を動的に作成するためのランタイムコマンドを含む依存関係２１２ａのマニフェストを列挙する「.gemspec」ファイルを含む例示的なRubyソフトウェアアプリケーション２００ａを示す。ここで、Rubyソフトウェアアプリケーション２００ａのアプリケーション依存関係３２２は、ブートストラップ実行環境３１０において依存関係２１２ａのマニフェストを実行すると動的に作成され、URL処理（curl）およびJSONデータ交換フォーマット（json）サポートライブラリを含む。いくつかの例では、「.gemspec」ファイルは、Rubyソフトウェアアプリケーション２００ａがブートストラップ実行環境３１０において実行されるときに動的に作成されるアプリケーション依存関係３２２の特定のバージョンまたはバージョンの好適な範囲を示すステートメントを含む。ここで、「.gemspec」ファイルは、バージョン０．０．９およびバージョン１．７．３がRubyソフトウェアアプリケーション２００ａのアプリケーション依存関係３２２であることを示す。いくつかの例では、「.gemspec」ファイルは、２．１より大きいが３．０より小さいjsonサポートライブラリがRubyソフトウェアアプリケーション２００ａのアプリケーション依存関係３２２であることを示すよう、ステートメント「s.add_dependency ('json', '~〉2.1')」を含む。コンテナビルダ３００は、Rubyソフトウェアアプリケーション２００ａがブートストラップ実行環境３１０内で実行されるときに、「.gemspec」ファイル内に列挙された依存関係のマニフェスト２１２に基づいてアプリケーション依存関係３２２を作成することによって、ブートストラップ実行環境３１０を拡張することができる。Rubyソフトウェアアプリケーション２００ａのためにブートストラップ実行環境３１０を拡張するための他の技法も可能である。コンテナビルダ３００は、拡張された実行環境３２０をデータストア１５０に記憶してもよい。

【００２３】

図２Ｂを参照すると、例示的なNode.jsソフトウェアアプリケーション（例えば、JavaScriptアプリケーション２００ｂ）が示される。Node.jsソフトウェアアプリケーション２００ｂは、アプリケーション依存関係３２２を動的に作成するためのランタイムコマンドを含む、依存関係のマニフェスト２１２ｂを列挙するパッケージマネージャ（npm）コマンドを含む。ここで、パッケージマネージャコマンドは、データベース（mysql）パッケージが、Node.jsソフトウェアアプリケーション２００ｂがブートストラップ実行環境３１０において実行されるときに動的に作成されることになるアプリケーション依存関係３２２であることを示す。いくつかの例では、コンテナビルダ３００は、Node.jsソフトウェアアプリケーション２００ｂがブートストラップ実行環境３１０内で実行されるとき、パッケージマネージャコマンドを実行してアプリケーション依存関係３２２を動的に作成することによってブートストラップ実行環境３１０を拡張する。Node.jsソフトウェアアプリケーション２００ｂは、依存関係のマニフェスト２１２をさらに列挙するJavaScriptステートメントを含む。ここで、JavaScriptステートメントは、暗号（「crypto」）およびデータベース（「mysql」）サポートライブラリも、Node.jsソフトウェアアプリケーション２００ｂがブートストラップ実行環境３１０において実行されるときに動的に作成されることになるアプリケーション依存関係３２２であることを示す。いくつかの例では、コンテナビルダ３００は、実行時（ＪＩＴ）コンパイラを含むJavaScriptエンジンにおいてNode.jsソフトウェアアプリケーション２００ｂを実行する。コンテナビルダ３００は、ブートストラップ実行環境３１０を拡張するために、ＪＩＴコンパイラに、Node.jsソフトウェアアプリケーション２００ｂを、より効率的な機械コードに、例えば実行時にコンパイルされるモジュールに、コンパイルさせてもよい。いくつかの実現例では、コンテナビルダ３００は、パッケージマネージャコマンドを実行し、Node.jsソフトウェアアプリケーション２００ｂを実行して、Node.jsソフトウェアアプリケーション２００ｂのアプリケーション依存関係３２２を動的に作成してもよい。Node.jsソフトウェアアプリケーション２００ｂのためのブートストラップ実行環境３１０を拡張するために他の技法が可能である。コンテナビルダ３００は、拡張された実行環境３２０をデータストア１５０に記憶してもよい。

【００２４】

図２Ｂを参照すると、例示的なPythonソフトウェアアプリケーション２００ｃが示される。Node.jsソフトウェアアプリケーション２００ｂと同様に、Pythonソフトウェアアプリケーション２００ｃは、アプリケーション依存関係３２２を動的に作成するためのランタイムコマンドを含む、依存関係のマニフェスト２１２を列挙するパッケージマネージャ（pip）コマンドを含む。ここで、パッケージマネージャコマンドは、ウェブフレームワーク（django）パッケージが、Pythonソフトウェアアプリケーション２００ｃがブートストラップ実行環境３１０において実行されるときに動的に作成されることになるアプリケーション依存関係３２２であることを示す。Pythonソフトウェアアプリケーション２００ｃは、アプリケーション依存関係のマニフェスト２１２をさらに列挙するPythonステートメントを含む。ここで、Pythonステートメントは、djangoの特定のバージョン（django==1.11.0, django-braces==0.2.1, django-model-utils==1.1.0, django-oauth2-provider==0.2.4, djangorestframework==2.3.1）サポートライブラリも、Pythonソフトウェアアプリケーション２００ｃがブートストラップ実行環境３１０において実行されるときに動的に作成されるアプリケーション依存関係３２２であることを示す。いくつかの実現例では、コンテナビルダ３００は、パッケージマネージャコマンドを実行し、Pythonソフトウェアアプリケーション２００ｃを実行して、Pythonソフトウェアアプリケーション２００ｃのためのアプリケーション依存関係３２２を動的に作成する。Pythonソフトウェアアプリケーション２００ｃのためのブートストラップ実行環境３１０を拡張するための他の技法が可能である。コンテナビルダ３００は、拡張された実行環境３２０をデータストア１５０に記憶してもよい。

【００２５】

図３Ａ～図３Ｅは、ソフトウェアアプリケーション２００のために拡張された実行環境３２０を構築するコンテナビルダ３００の概略図を示す。図３Ａを参照すると、コンテナビルダ３００の構成段階３０１（図１）が示される。いくつかの例では、コンテナビルダ３００は、構築要求１８０を受信する。ここで、構築要求１８０は、単に、依存関係のマニフェスト２１２を含むソフトウェアアプリケーション２００を含む。構成段階３０１の間、コンテナビルダ３００は、ソフトウェアアプリケーション２００のためにブートストラップ実行環境３１０を構成し、構築する。ブートストラップ実行環境３１０は、ソフトウェアアプリケーション２００が動作環境にかかわらず一貫して実行できるように、ソフトウェアアプリケーション２００を最初に実行するために必要とされるすべて、たとえばコード、ランタイム、システムツール、およびライブラリを含む完全なファイルシステムを含んで、ソフトウェアアプリケーション２００を静的イメージに含んでもよい。いくつかの例では、コンテナビルダ３００は、動的コンテンツを作成することなく、すなわち、実行時アーキテクチャ固有および／または動的アプリケーション依存関係３２２を伴わずに、静的イメージ（たとえば、ブートストラップ実行環境３１０）を構成する。

【００２６】

コンテナビルダ３００は、実行環境３１０、３２０を構築または構成するためのツーリング３１６を含むようにブートストラップ実行環境３１０を構成することができる。いくつかの実現例では、ツーリング３１６は、コンテナを作成、構成、および管理するためのコマンドラインツールのセットである。ツーリング３１６はまた、拡張された実行環境３２０をデータストア１５０に記憶することが可能であってもよい。例えば、図３Ａは、コンテナビルダ３００がブートストラップ実行環境３１０をデータストア１５０に記憶するのを示す。実行段階３０２に関して以下でより詳細に説明されるように、ツーリング３１６はまた、たとえば拡張された実行環境３２０が利用可能であるかどうかを判断するために、データストア１５０と対話することが可能であってもよい。いくつかの例では、ツーリング３１６は、実行環境内観が可能である。すなわち、ツーリング３１６は、実行環境、たとえばブートストラップ実行環境３１０を拡張して、拡張された実行環境３２０を作成するプロセス中に、ブートストラップ実行環境３１０内のどのファイルが追加または修正されたかを判断することが可能であってもよい。いくつかの実現例では、コンテナビルダ３００は、起動命令のセット３１８を含むようにブートストラップ実行環境３１０を構

成する。起動命令３１８は、実行段階３０２中にブートストラップ実行環境３１０を実行するときにコンテナビルダ３００が実行するための動作（例えば、ステップ）を定義する。コンテナビルダ３００は、起動命令３１８のステップを実行する際に、ブートストラップ実行環境３１０に含まれるツーリング３１６を用いてもよい。コンテナビルダ３００は、ツーリング３１６を用いて、ブートストラップ実行環境３１０を含むコンテナイメージをコンテナレジストリ（例えば、データストア１５０）にアップロードすることができる。コンテナビルダ３００は、ブートストラップ実行環境３１０に関連付けられるソフトウェアアプリケーション２００を識別する識別子３１４（たとえば、タグ）をブートストラップ実行環境３１０に追加することができる。識別子３１４は、その後の検索および取得を容易にすることができる。

【００２７】

図３Ｂを参照すると、コンテナビルダ３００は、実行段階３０２中にブートストラップ実行環境３１０を実行する。コンテナビルダ３００は、構成段階３０１（図３Ａ）中にブートストラップ実行環境３１０を構築／作成／構成した後、データストア１５０からブートストラップ実行環境３１０を（たとえば、識別子３１４を用いて）受信してもよい。いくつかの例では、コンテナビルダ３００は、構成段階３０１中にブートストラップ実行環境３１０を作成／構成すると、実行段階３０２中にブートストラップ実行環境３１０を直ちに実行する。ブートストラップ実行環境３１０は、依存関係のマニフェスト２１２を含むソフトウェアアプリケーション２００を含んでもよい。図示の例では、コンテナビルダ３００はブートストラップ実行環境３１０を実行し、コンテナビルダ３００に起動命令３１８を実行させる。起動命令３１８は、関連付けられる拡張された実行環境３２０がデータストア１５０から利用可能であるかどうかを判断することを含んでもよい。図３Ｂに示す例では、データストア１５０は、最初は、拡張された実行環境３２０を含まない。したがって、コンテナビルダ３００は、起動命令３１８を実行している間、関連付けられる拡張された実行環境３２０がデータストア１５０から利用可能ではないと判断する。例えば、コンテナビルダ３００は、識別子３１４を用いてデータストア１５０に問い合わせて、データストア１５０が関連付けられる拡張された実行環境３２０を含むかどうかを判断してもよい。識別子３１４は、ソフトウェアアプリケーション２００のバージョンに関連付けられてもよい。いくつかの例では、コンテナビルダ３００は、ブートストラップ実行環境３１０に関連付けられ（かつ構築要求１８０に含まれる）ソフトウェアアプリケーション２００のバージョン番号が、記憶された拡張された実行環境３２０に関連付けられるソフトウェアアプリケーション２００のバージョンと同じでない場合、拡張された実行環境３２０は利用可能ではないと判断する。

【００２８】

コンテナビルダ３００が、関連付けられる拡張された実行環境３２０がデータストア１５０から利用可能ではないと判断すると、起動命令３１８は、依存関係のマニフェスト２１２に基づいてブートストラップ実行環境３１０を拡張するための動作／ステップを含む。コンテナビルダ３００は、ブートストラップ実行環境３１０を拡張して、拡張された実行環境３２０を構築／作成／構成してもよい。コンテナビルダ３００は、アプリケーション依存関係３２２をインストールする（例えば、ダウンロードする、コピーする、コンパイルする）ことによってブートストラップ実行環境３１０を拡張してもよい。いくつかの例では、コンテナビルダ３００は、ソフトウェアアプリケーション２００を実行する前にアプリケーション依存関係３２２をインストールする。前述のように、コンテナビルダ３００は、依存関係のマニフェスト２１２に含まれるパッケージマネージャコマンドなどのコマンドを実行して、アプリケーション依存関係３２２を作成（例えばインストール）してもよい。いくつかの例では、コンテナビルダ３００は、ソフトウェアアプリケーション２００を実行して、アプリケーション依存関係３２２を作成する。コンテナビルダ３００は、ソフトウェアアプリケーション２００を実行して／走らせて、ソフトウェアアプリケーション２００に、サポートライブラリ、ファイル、もしくは他のアプリケーション依存関係３２２を必要とするかまたはインポートするステートメントを実行させてもよい。し

たがって、コンテナビルダ３００は、インポートされたアプリケーション依存関係３２２を用いてブートストラップ実行環境３１０を拡張して、拡張された実行環境３２０を作成する。コンテナビルダ３００は、ソフトウェアアプリケーション２００を実行して、アプリケーションソースコードまたはバイトコードなどの共通中間言語（ＣＩＬ）をアーキテクチャ固有の機械コードまたはバイナリコードにコンパイルすることができる。例えば、コンテナビルダ３００は、ＪＩＴ実行エンジンにおいてソフトウェアアプリケーション２００を実行してもよい。ＪＩＴエンジンは、ソフトウェアアプリケーション２００のモジュールが最初にロードまたはアクセスされるときに、ソフトウェアアプリケーション２００のモジュールを、効率的な機械コードを含むＪＩＴコンパイルされたモジュールにコンパイルしてもよい。いくつかの実現例では、コンテナビルダ３００は、ソフトウェアアプリケーション２００を実行する前にいくつかのアプリケーション依存関係３２２を作成し、ソフトウェアアプリケーション２００を実行／実行することによって他のアプリケーション依存関係３２２を作成する。コンテナビルダ３００は、アプリケーション依存関係３２２を動的に作成するために他の技法を用いてもよい。

【００２９】

動的コンテンツ（例えば、アプリケーション依存関係３２２）は、アーキテクチャ固有であってもよい。すなわち、アプリケーション依存関係３２２は、アプリケーション依存関係３２２を作成するためにコンテナビルダ３００によって実行されるブートストラップ実行環境３１０に関連付けられる特定のプロセッサアーキテクチャおよび／または特定のオペレーティングシステムバージョンと互換性があってもよい。例えば、アプリケーション依存関係３２２は、効率的なアーキテクチャ固有の機械コードもしくはバイナリコード、および／もしくはＪＩＴ実行エンジンによって作成される、実行時にコンパイルされるモジュールを含むファイルまたはモジュールを含んでもよい。いくつかの実現例では、コンテナビルダ３００が、動的に作成されたアプリケーション依存関係３２２を含むようにブートストラップ実行環境３１０を拡張した後、拡張された実行環境３２０は、もはや、アプリケーション依存関係３２２を作成することに関連付けられる起動命令３１８のステップを実行する必要はない。したがって、コンテナビルダ３００は、拡張された実行環境３２０から起動命令３１８を省くことができる。コンテナビルダ３００は、拡張された実行環境３２０を作成したブートストラップ実行環境３１０に関連付けられるオペレーティングシステムバージョンおよび／またはブートストラップ実行環境３１０に関連付けられるプロセッサアーキテクチャを示すメタデータ３２４を、拡張された実行環境３２０に追加してもよい。いくつかの例では、アーキテクチャ固有コンテンツ（たとえば、アプリケーション依存関係３２２）は、ブートストラップ実行環境３１０に関連付けられるオペレーティングシステムバージョンおよび／またはプロセッサアーキテクチャを示すメタデータを含む。コンテナ構築は、拡張された実行環境３２０を作成／構成するために他の技法を用いてもよい。

【００３０】

図３Ｂはまた、コンテナビルダ３００が、拡張された実行環境３２０をデータストア１５０に記憶することを含む起動命令３１８のさらなるステップを実行するのを示す。いくつかの実現例では、コンテナビルダ３００は、ツーリング３１６を用いて、拡張された実行環境３２０を記憶する。例えば、データストア１５０は、コンテナレジストリを含んでもよい。コンテナビルダ３００は、ツーリング３１６を用いて、拡張された実行環境３２０を含むコンテナイメージを作成することができる。拡張された実行環境３２０は、ブートストラップ実行環境３１０のコンテンツ（たとえば、静的コンテンツ）と、コンテナビルダ３００によって作成されたアプリケーション依存関係３２２（動的コンテンツ）とを含んでもよい。コンテナビルダ３００は、さらに、ツーリング３１６を用いて、拡張された実行環境３２０を含むコンテナイメージをコンテナレジストリ（例えば、データストア１５０）にアップロードしてもよい。起動命令３１８は、検索および取得を容易にするために、拡張された実行環境３２０をタグ付けすること、注釈付けすること、インデックス付けすること、または別様に識別することをさらに含んでもよい。例えば、起動命令３１

８は、拡張された実行環境３２０に識別子３１４を追加することを含んでもよい。例えば、識別子３１４は、データストア１５０が、ソフトウェアアプリケーション２００のバージョンに関連付けられる拡張された実行環境３２０を含むことを示してもよい。コンテナビルダ３００は、拡張された実行環境３２０を実行する遠隔システム１４０のオペレーティングシステムバージョンおよび／またはプロセッサアーキテクチャなどの、拡張された実行環境３２０の態様を示すために、メタデータ３２４を拡張された実行環境３２０に追加することができる。例えば、コンテナビルダ３００は、メタデータ３２４を用いて、拡張された実行環境３２０に「タグ」または注釈を付けることができる。一例では、コンテナビルダ３００は、拡張された実行環境３２０が「x86_64」アーキテクチャ上で実行されるUbuntu LINUXシステム上で作成されたことを示すメタデータ３２４を、拡張された実行環境３２０にタグ付けする。他のメタデータ３２４の規約またはコンテンツも同様に使用されてもよい。

【００３１】

図３Ｃを参照すると、コンテナビルダ３００は、ブートストラップ実行環境３１０と拡張された実行環境３２０との間の差分をデータストア１５０に記憶してもよい。すなわち、コンテナビルダ３００は、例えば、データストア１５０内の空間を節約するために、動的に作成されたアプリケーション依存関係３２２のみをデータストア１５０内に記憶してもよい。ここで、拡張された実行環境のための動的に作成されたアプリケーション依存関係３２２は、データストア１５０に記憶されてもよく、関連付けられる識別子３１４（例えば、アプリケーション依存関係の、後の検索および取得のため）および／または関連付けられるメタデータ３２４（例えば、動的に作成されたアプリケーション依存関係に関連付けられる拡張された実行環境３２０を実行するための遠隔システム１４０のオペレーティングシステムバージョンおよび／またはプロセッサアーキテクチャなどの態様を示すため）を含んでもよい。たとえば、起動命令３１８は、実行段階３０２の間に更新または動的に作成されたアプリケーション依存関係３２２を識別することによってブートストラップ実行環境３１０と拡張された実行環境３２０との間の差を判断するための内観を含んでもよい。すなわち、コンテナビルダ３００は、ブートストラップ実行環境３１０を拡張することによって追加または修正されたファイルに対応するアプリケーション依存関係３２２を判断／識別してもよい。コンテナビルダ３００は、ブートストラップ実行環境３１０に含まれるツーリング３１６を用いて、更新されたファイルのセット、すなわち、コンテナビルダ３００が動的コンテンツ（たとえば、アプリケーション依存関係３２２）を追加することによってブートストラップ環境３１０を拡張したときに追加または修正されたファイルを識別してもよい。いくつかの実現例では、コンテナビルダ３００は、標準的なオペレーティングシステムコマンドまたはプログラミングインターフェイスを用いて、依存関係のマニフェスト２１２に基づいてブートストラップ実行環境３１０を拡張することによって追加または修正されたファイルを判断する。例えば、コンテナビルダ３００は、UNIX（登録商標）「find」コマンドを用いて、変更されたファイルを再帰的に検索することができる。コンテナビルダ３００は、ブートストラップ実行環境３１０に対する拡張を判断するために他の技法を用いることができる。いくつかの実現例では、コンテナビルダ３００は、最近追加または修正されたファイルに基づいてUNIX（登録商標）「tar」ファイルを作成する。コンテナビルダ３００は、「tar」ファイルを拡張された実行環境３２０としてデータストア１５０に記憶し、「tar」ファイルをタグ付けして、拡張された実行環境３２０が、完全に実行可能な拡張された実行環境３２０ではなく、アプリケーション依存関係３２２のみを含む差分を含むことを示してもよい。その後ブートストラップ実行環境３１０が実行されると、コンテナビルダ３００は、データストア１５０から受信される「tar」ファイル内のアプリケーション依存関係３２２に基づいてブートストラップ実行環境３１０を拡張してもよい。

【００３２】

関連付けられる拡張された実行環境３２０がデータストア１５０から利用可能ではないとコンテナビルダが判断した図３Ｂに示される例とは対照的に、図３Ｄは、コンテナビル

ダ３００が、起動命令３１８を実行している間に、関連付けられる拡張された実行環境３２０がデータストア１５０から利用可能である（例えば、データストア１５０は、関連付けられる拡張された実行環境３２０を含む）、と判断するのを示す。いくつかの例においては、コンテナビルダ３００は、拡張された実行環境３２０を以前に作成しており、データストア１５０に記憶していた。以前に作成された実行環境３２０は、拡張された実行環境３２０をソフトウェアアプリケーション２００のバージョンと関連付ける識別子３１４を含んでもよい。コンテナビルダ３００は、拡張された実行環境３２０がソフトウェアアプリケーション２００の好適なバージョン（例えば、バージョン番号）を含むことを示す「タグ」（例えば、識別子３１４）を、拡張された実行環境３２０が含む場合に、拡張された実行環境３２０は利用可能である、と判断してもよい。いくつかの例では、バージョン番号は、メジャーバージョン番号およびマイナーバージョン番号を含む。これらの例では、コンテナビルダ３００は、ブートストラップ実行環境３１０に関連付けられるソフトウェアアプリケーション２００のメジャーソフトウェアバージョン番号が、記憶される拡張された実行環境３２０に関連付けられるソフトウェアアプリケーション２００のメジャーバージョンと同じである場合、関連付けられる拡張された実行環境３２０はデータストア１５０から利用可能である、と判断してもよい。

【００３３】

関連付けられる拡張された実行環境３２０は、拡張された実行環境３２０を構築するために使用される遠隔システム１４０のオペレーティングシステムバージョンおよび／またはプロセッサアーキテクチャなどの、拡張された実行環境３２０の態様を示すメタデータ３２４を含んでもよい。拡張された実行環境３２０は、メタデータ３２４を含む注釈またはタグを含んでもよい。コンテナビルダ３００は、拡張された実行環境３２０を構築した遠隔システム１４０のオペレーティングシステムバージョンおよび／またはプロセッサアーキテクチャなど、拡張された実行環境３２０の態様が好適であることを示すメタデータ３２４を拡張された実行環境３２０が含む場合、拡張された実行環境３２０は利用可能である、と判断してもよい。たとえば、コンテナビルダ３００は、メタデータ３２４がブートストラップ実行環境３１０と互換性がない（例えば、ブートストラップ実行環境３１０を実行する遠隔システム１４０とは互換性がない）プロセッサアーキテクチャを示すとき、拡張された実行環境３２０はデータストア１５０から利用可能ではない、と判断してもよい。この例では、コンテナビルダ３００は、図３Ｂを参照して上述したように、依存関係のマニフェスト２１２に基づいてブートストラップ実行環境３１０を拡張することによって起動命令３１８を実行することに進むことになる。代替として、コンテナビルダ３００は、拡張された実行環境３２０に関連付けられるメタデータ３２４が、ブートストラップ実行環境３１０と互換性があるオペレーティングシステムバージョンおよびプロセッサアーキテクチャを示すとき、拡張された実行環境３２０はデータストア１５０から利用可能である、と判断してもよい。コンテナビルダ３００は、他の技法を用いて、拡張された実行環境３２０がデータストア１５０から利用可能であるかどうかを判断してもよい。コンテナビルダ３００は、ブートストラップ実行環境３１０に含まれるツーリング３１６を用いて、関連付けられる拡張された実行環境３２０が利用可能であるかどうかを判断してもよい。例えば、データストア１５０は、コンテナレジストリを含んでもよい。この例では、ツーリング３１６は、コンテナレジストリ（例えば、データストア１５０）内で好適な拡張された実行環境３２０を検索することが可能であってもよい。ツールは、ソフトウェアバージョン識別子３１４および／またはメタデータ３２４を用いて、または好適な拡張された実行環境３２０がデータストア１５０から利用可能であることを判断するために他の技法を用いて、拡張された実行環境３２０を検索してもよい。

【００３４】

図示の例では、コンテナビルダ３００が、関連付けられる拡張された実行環境３２０がデータストア１５０から利用可能である、と判断すると、コンテナビルダ３００は、拡張された実行環境３２０をデータストア１５０から受信し、拡張された実行環境３２０に基づいてブートストラップ実行環境３１０を拡張する。コンテナビルダ３００は、データス

トア１５０からソフトウェアアプリケーション２００を含む拡張された実行環境３２０をダウンロードすることによって、拡張された実行環境３２０を受信し、静的コンテンツのみを有するブートストラップ実行環境３１０の代わりに、動的コンテンツ（たとえば、アプリケーション依存関係３２２）を有するダウンロードされた拡張された実行環境３２０を単に実行することによって、ブートストラップ実行環境３１０を拡張してもよい。別の例では、ダウンロードされた拡張された実行環境３２０を実行する代わりに、コンテナビルダ３００は、ダウンロードされた拡張された実行環境３２０に含まれるアプリケーション依存関係３２２に基づいてブートストラップ実行環境３１０を拡張する。例えば、コンテナビルダ３００は、ツーリング３１６を用いて、アプリケーション依存関係３２２をデータストア１５０内の関連付けられる拡張された実行環境３２０からブートストラップ実行環境３１０にダウンロードおよびインストールしてもよい。拡張された実行環境３２０に基づいてブートストラップ実行環境３１０を拡張することは、依存関係のマニフェスト２１２に基づいてブートストラップ実行環境３１０を拡張することよりも効率的であり得る。

【００３５】

図３Ｃに関して上述したように、コンテナビルダ３００は、ブートストラップ実行環境３１０と拡張された実行環境３２０との間の差分（すなわち、動的に作成されたアプリケーション依存関係３２２）をデータストア１５０に記憶してもよい。図３Ｅを参照すると、コンテナビルダ３００が、関連付けられる拡張された実行環境３２０がデータストア１５０から利用可能であるかどうかについて起動命令３１８を実行するとき、コンテナビルダ３００は、好適な差分がデータストア１５０から利用可能である、と判断してもよい。ここで、差分は、例えばブートストラップ実行環境３１０におけるソフトウェアアプリケーション２００の以前の実行中に動的に作成され、コンパイルされ、ダウンロードされ、またはそうでなければマーシャリングされたアプリケーション依存関係３２２を含むコンテナイメージを含んでもよい。前述のように、コンテナビルダ３００は、受信したブートストラップ実行環境３１０の識別子３１４を用いて、データストア１５０内のアプリケーション依存関係３２２を含む関連付けられる差分を検索してもよい。例えば、コンテナビルダ３００は、差分が遠隔システム１４０と同じオペレーティングシステムバージョンおよび／または同じプロセッサアーキテクチャを用いて生成されたことを示す「タグ」（例えば、識別子３１４）を、差分が含むとき、差分はデータストア１５０から利用可能である、と判断してもよい。いくつかの例では、コンテナビルダ３００は、アプリケーション依存関係３２２を含むUNIX（登録商標）「tar」ファイルを含むコンテナイメージ（たとえば、拡張された実行環境３２０）をデータストア１５０から受信する。コンテナビルダ３００は、データストア１５０から受信されたコンテナイメージに含まれるアプリケーション依存関係３２２に基づいてブートストラップ実行環境３１０を拡張してもよい。コンテナビルダ３００は、ツーリング３１６を用いて、データストア１５０からコンテナイメージをダウンロードし、アプリケーション依存関係３２２をインストールすることによってブートストラップ実行環境３１０を拡張してもよい。例えば、コンテナビルダ３００は、「tar」ファイルからアプリケーション依存関係３２２を抽出してもよい。コンテナビルダ３００は、拡張された実行環境３２０に基づいてブートストラップ実行環境３１０を拡張するために他の技法を用いてもよい。

【００３６】

いくつかの例では、実行段階３０２中にブートストラップ実行環境３１０を拡張した後、コンテナビルダ３００はソフトウェアアプリケーション２００を実行する。あるいは、コンテナビルダ３００は、ブートストラップ実行環境３１０内でソフトウェアアプリケーション２００を実行して、アプリケーション依存関係３２２を作成（すなわち、拡張された実行環境３２０を作成）してもよい。この例では、ソフトウェアアプリケーション２００は、拡張された実行環境３２０において続行してもよい。言い換えれば、起動命令３１８のステップ／動作の順序、例えば、拡張された実行環境３２０を作成するステップ、およびソフトウェアアプリケーション２００を実行するステップは、示される例から変更さ

れてもよい。
【００３７】
図４は、アプリケーション実行環境においてソフトウェアアプリケーション２００を実行する方法４００のための動作の例示的な構成のフローチャートを提供する。ソフトウェアアプリケーションは、JavaScriptアプリケーション、Pythonアプリケーション、またはRubyアプリケーションのうちの１つであってもよい。動作４０２において、方法４００は、データ処理ハードウェア１４４（例えば、コンテナビルダ３００）において、データストア１５０からブートストラップ実行環境３１０を受信することを含む。ブートストラップ実行環境３１０はソフトウェアアプリケーション２００を含み、ソフトウェアアプリケーション２００は依存関係のマニフェスト２１２を含む。いくつかの実現例では、データストア１５０は、ブートストラップ実行環境３１０を含むコンテナイメージを記憶するコンテナレジストリを含む。
【００３８】
動作４０４において、方法４００は、データ処理ハードウェア１４４が、ブートストラップ実行環境３１０を実行して、データ処理ハードウェア１４４に（たとえば、ブートストラップ実行環境３１０に関連付けられる起動命令３１８を実行することによって）動作を実行させることを含む。動作４０６において、方法４００は、拡張された実行環境３２０がデータストア１５０から利用可能であるかどうかを判断することを含む。たとえば、拡張された実行環境３２０がデータストア１５０から利用可能であるかどうかを判断することは、拡張された実行環境３２０が、ソフトウェアアプリケーション２００、ブートストラップ実行環境３２０に関連付けられるオペレーティングシステムバージョン、またはブートストラップ実行環境３２０に関連付けられるプロセッサアーキテクチャのうちの少なくとも１つと互換性があるかどうかを判断することを含んでもよい。
【００３９】
拡張された実行環境３２０がデータストア１５０から利用可能である場合、方法４００は、動作４０８において、拡張された実行環境３２０をデータストア１５０から受信することを含む。データストア１５０から拡張された実行環境３２０を受信した後、方法４００は、動作４１０において、拡張された実行環境３２０に基づいてブートストラップ実行環境３１０を拡張することを含む。たとえば、ブートストラップ実行環境３１０を拡張することは、拡張された実行環境３２０に基づいてアプリケーション依存関係３２２をインストールすることを含んでもよい。アプリケーション依存関係３２２は、サポートライブラリ、アーキテクチャ固有のバイナリモジュール、または実行時にコンパイルされるモジュールのうちの少なくとも１つを含んでもよい。拡張された実行環境３２０に基づいてブートストラップ実行環境３１０を拡張した後、方法４００は、動作４１６において、ソフトウェアアプリケーション２００を実行することを含む。
【００４０】
拡張された実行環境３２０がデータストア１５０から利用可能ではない場合、方法４００は、動作４１２において、拡張された実行環境３２０を作成するために、依存関係のマニフェスト２１２に基づいてブートストラップ実行環境３１０を拡張することを含む。拡張された実行環境３２０を作成するために依存関係のマニフェスト２１２に基づいてブートストラップ実行環境３１０を拡張した後、方法４００は、動作４１４において、拡張された実行環境３２０をデータストア１５０に記憶することを含む。いくつかの実現例では、方法４００は、データ処理ハードウェア１４４が、データストア１５０内の拡張された実行環境３２０に注釈を付けて、ソフトウェアアプリケーション２００のバージョン、拡張された実行環境３２０に関連付けられるオペレーティングシステムバージョン、または拡張された実行環境３２０に関連付けられるプロセッサアーキテクチャのうちの少なくとも１つを示すことも含む。拡張された実行環境３２０をデータストア１５０に記憶した後、方法４００は、動作４１６において、ソフトウェアアプリケーション２００を実行することを含む。
【００４１】

図５は、本明細書で説明するシステムおよび方法（たとえば、方法４００）を実施するために使用され得る例示的なコンピューティングデバイス５００の概略図である。コンピューティングデバイス５００は、ラップトップ、デスクトップ、ワークステーション、携帯情報端末、サーバ、ブレードサーバ、メインフレーム、および他の適切なコンピュータなど、様々な形態のデジタルコンピュータを表すことが意図されている。本明細書に示された構成要素、それらの接続および関係、ならびにそれらの機能は、例示的なものにすぎず、本文書に記載および／または特許請求される本発明の実現例を限定するものではない。

【００４２】

コンピューティングデバイス５００は、プロセッサ５１０（データ処理ハードウェアとも呼ばれる）と、メモリ５２０（メモリハードウェアとも呼ばれる）と、ストレージデバイス５３０と、メモリ５２０および高速拡張ポート５５０に接続する高速インターフェイス／コントローラ５４０と、低速バス５７０およびストレージデバイス５３０に接続する低速インターフェイス／コントローラ５６０とを含む。構成要素５１０，５２０，５３０，５４０，５５０，および５６０の各々は、様々なバスを用いて相互接続され、共通のマザーボード上に、または必要に応じて他の方法で実装されてもよい。プロセッサ５１０は、高速インターフェイス５４０に結合されたディスプレイ５８０などの外部入力／出力装置上にグラフィカルユーザインターフェイス（ＧＵＩ）のためのグラフィカル情報を表示するために、メモリ５２０またはストレージデバイス５３０に記憶された命令を含む、コンピューティングデバイス５００内で実行するための命令を処理することができる。他の実現例では、複数のプロセッサおよび／または複数のバスが、必要に応じて、複数のメモリおよびメモリのタイプとともに用いられてもよい。また、複数のコンピューティングデバイス５００が接続されてもよく、各デバイスは、（たとえば、サーババンクとして、ブレードサーバのグループとして、またはマルチプロセッサシステムとして）必要な動作の部分を提供する。

【００４３】

メモリ５２０は、コンピューティングデバイス５００内で情報を非一時的に記憶する。メモリ５２０は、コンピュータ可読媒体、揮発性メモリユニット、または不揮発性メモリユニットであってもよい。非一時的メモリ５２０は、コンピューティングデバイス５００による使用のためにプログラム（例えば、命令のシーケンス）またはデータ（例えば、プログラム状態情報）を一時的または永続的に記憶するために用いられる物理デバイスであってもよい。不揮発性メモリの例は、フラッシュメモリおよび読み出し専用メモリ（ＲＯＭ）／プログラマブル読み出し専用メモリ（ＰＲＯＭ）／消去可能プログラマブル読み出し専用メモリ（ＥＰＲＯＭ）／電子的消去可能プログラマブル読み出し専用メモリ（ＥＥＰＲＯＭ）（たとえば、ブートプログラムなどのファームウェアに典型的に用いられる）を含むが、これらに限定されない。揮発性メモリの例には、ランダムアクセスメモリ（ＲＡＭ）、ダイナミックランダムアクセスメモリ（ＤＲＡＭ）、スタティックランダムアクセスメモリ（ＳＲＡＭ）、相変化メモリ（ＰＣＭ）、およびディスクまたはテープが含まれるが、これらに限定されない。

【００４４】

ストレージデバイス５３０は、コンピューティングデバイス５００のための大容量ストレージを提供することができる。いくつかの実現例では、ストレージデバイス５３０はコンピュータ可読媒体である。様々な異なる実現例では、ストレージデバイス５３０は、フロッピー（登録商標）ディスクデバイス、ハードディスクデバイス、光ディスクデバイス、もしくはテープデバイス、フラッシュメモリもしくは他の同様のソリッドステートメモリデバイス、またはストレージエリアネットワークもしくは他の構成におけるデバイスを含むデバイスのアレイであってもよい。さらなる実現例では、コンピュータプログラム製品は、情報担体において有形に具現化される。コンピュータプログラム製品は、実行されると上述の方法などの１つ以上の方法を実行する命令を含む。情報担体は、メモリ５２０、ストレージデバイス５３０、またはプロセッサ５１０上のメモリなどのコンピュータ可読媒体または機械可読媒体である。

【００４５】
高速コントローラ５４０は、コンピューティングデバイス５００のための帯域幅集約型動作を管理し、低速コントローラ５６０は、より低い帯域幅集約型動作を管理する。そのような役割の割り当ては、例示的なものにすぎない。いくつかの実現例では、高速コントローラ５４０は、メモリ５２０、ディスプレイ５８０（たとえば、グラフィックスプロセッサまたはアクセラレータを通して）、および様々な拡張カード（図示せず）を受け入れ得る高速拡張ポート５５０に結合される。いくつかの実現例では、低速コントローラ５６０は、ストレージデバイス５３０および低速拡張ポート５９０に結合される。低速拡張ポート５９０は、様々な通信ポート（たとえば、ＵＳＢ、Bluetooth（登録商標）、イーサネット（登録商標）、無線イーサネット（登録商標））を含んでもよく、キーボード、ポインティングデバイス、スキャナ、またはスイッチもしくはルータなどのネットワーキングデバイスなどの１つ以上の入力／出力装置に、たとえばネットワークアダプタを介して結合されてもよい。

【００４６】
コンピューティングデバイス５００は、図に示されるように、いくつかの異なる形態で実現されてもよい。例えば、標準サーバ５００ａとして、もしくはそのようなサーバ５００ａのグループ内で複数回、ラップトップコンピュータ５００ｂとして、またはラックサーバシステム５００ｃの一部として実装されてもよい。

【００４７】
本明細書に記載のシステムおよび技術のさまざまな実現例は、デジタル電子および／もしくは光学回路系、集積回路系、特別に設計されたＡＳＩＣ（特定用途向け集積回路）、コンピュータハードウェア、ファームウェア、ソフトウェア、ならびに／またはそれらの組合せで実現されてもよい。これらのさまざまな実現例は、少なくとも１つのプログラマブルプロセッサを含むプログラマブルシステム上で実行可能および／または解釈可能な１つ以上のコンピュータプログラムにおける実現例を含んでいてもよく、当該プロセッサは専用であっても汎用であってもよく、ストレージシステム、少なくとも１つの入力装置、および少なくとも１つの出力装置からデータおよび命令を受信するように、かつこれらにデータおよび命令を送信するように結合されている。

【００４８】
これらのコンピュータプログラム（プログラム、ソフトウェア、ソフトウェアアプリケーションまたはコードとしても知られる）は、プログラム可能なプロセッサのための機械命令を含み、高水準手続き型および／もしくはオブジェクト指向型プログラミング言語で、ならびに／またはアセンブリ／機械言語で実現することができる。本明細書で使用されるとき、用語「機械可読媒体」および「コンピュータ可読媒体」は、機械命令を機械可読信号として受信する機械可読媒体を含む、機械命令および／またはデータをプログラマブルプロセッサに提供するよう使用される任意のコンピュータプログラム製品、非一時的コンピュータ可読媒体、装置および／またはデバイス（例えば、磁気ディスク、光ディスク、メモリ、プログラマブルロジックデバイス（ＰＬＤ））を指す。「機械可読信号」という用語は、機械命令および／またはデータをプログラマブルプロセッサに提供するために使用される任意の信号を指す。

【００４９】
本明細書に記載されるプロセスおよび論理フローは、入力データを操作し出力を生成することにより機能を実行するよう１つ以上のプログラマブルプロセッサが１つ以上のコンピュータプログラムを実行することによって実行され得る。プロセスおよび論理フローはまた、専用論理回路、たとえば、ＦＰＧＡ（フィールドプログラマブルゲートアレイ）またはＡＳＩＣ（特定用途向け集積回路）によって実行され得る。コンピュータプログラムの実行に好適であるプロセッサは、例として、汎用マイクロプロセッサおよび特殊目的マイクロプロセッサの両方、ならびに任意の種類のデジタルコンピュータの任意の１つ以上のプロセッサを含む。概して、プロセッサは、読み取り専用メモリもしくはランダムアクセスメモリまたは両方から命令およびデータを受信することになる。コンピュータの必須

要素は、命令を実行するためのプロセッサ、ならびに命令およびデータを記憶するための１つ以上のメモリデバイスである。一般に、コンピュータはさらに、たとえば磁気ディスク、光磁気ディスクまたは光ディスクといった、データを格納するための１つ以上の大容量記憶装置を含むか、当該１つ以上の大容量記憶装置からデータを受取るかもしくは当該１つ以上の大容量記憶装置にデータを転送するよう作動的に結合されるか、またはその両方を行うことにもなる。しかしながら、コンピュータは、そのようなデバイスを有する必要はない。コンピュータプログラム命令およびデータを記憶するのに好適なコンピュータ可読媒体は、例として、半導体メモリデバイス、たとえば、ＥＰＲＯＭ、ＥＥＰＲＯＭ、およびフラッシュメモリデバイス；磁気ディスク、たとえば内蔵ハードディスクまたはリムーバブルディスク；光磁気ディスク；およびＣＤ　ＲＯＭおよびＤＶＤ－ＲＯＭディスクを含む、あらゆる形態の不揮発性メモリ、媒体、ならびにメモリデバイスを含む。プロセッサおよびメモリは、特殊目的論理回路によって補足され得るか、または特殊目的論理回路に組み込まれ得る。

【００５０】

ユーザとの対話を提供するために、本開示の１つ以上の局面は、たとえばＣＲＴ（陰極線管）、ＬＣＤ（液晶ディスプレイ）モニタまたはタッチスクリーンといったユーザに対して情報を表示するための表示装置と、選択肢的にキーボードおよびたとえばマウス、トラックボールといったユーザがコンピュータに入力を提供可能であるポインティングデバイスとを有するコンピュータ上で実現され得る。他の種類のデバイスを用いて、ユーザとの対話を提供することもでき、たとえば、ユーザに提供されるフィードバックは、任意の形態の感覚フィードバック、たとえば、視覚フィードバック、聴覚フィードバック、または触覚フィードバックであり得、ユーザからの入力は、音響入力、音声入力、または触覚入力を含む、任意の形態で受信することができる。加えて、コンピュータは、ユーザが用いるデバイスにドキュメントを送信し、ユーザが用いるデバイスからドキュメントを受信することによって、たとえば、ユーザのクライアントデバイス上のウェブブラウザから受信された要求に応答してそのウェブブラウザにウェブページを送信することによって、ユーザと対話し得る。

【００５１】

いくつかの実現例について説明した。それにもかかわらず、本開示の精神および範囲から逸脱することなく、様々な修正がなされ得ることが理解されるであろう。したがって、他の実現例は特許請求の範囲内にある。

【図面】

【図１】

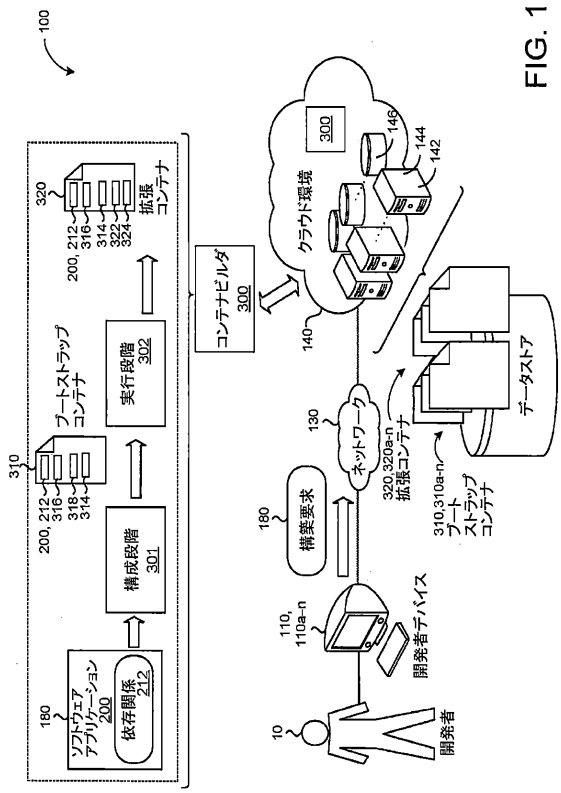

FIG. 1

【図２Ａ】

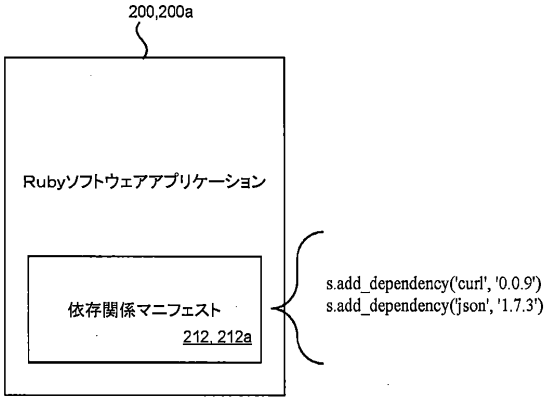

FIG. 2A

【図２Ｂ】

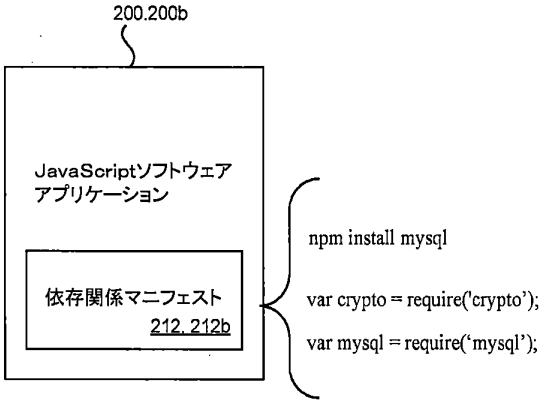

FIG. 2B

【図２Ｃ】

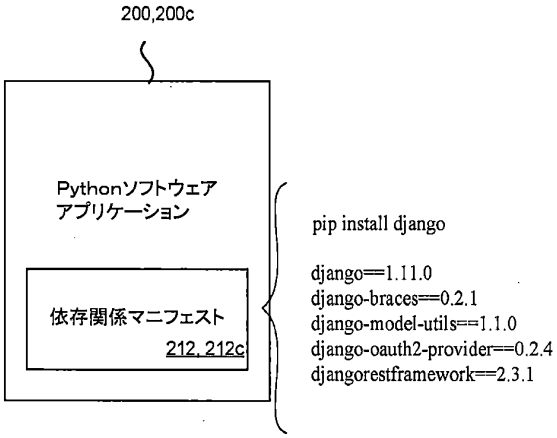

FIG. 2C

【図３Ａ】

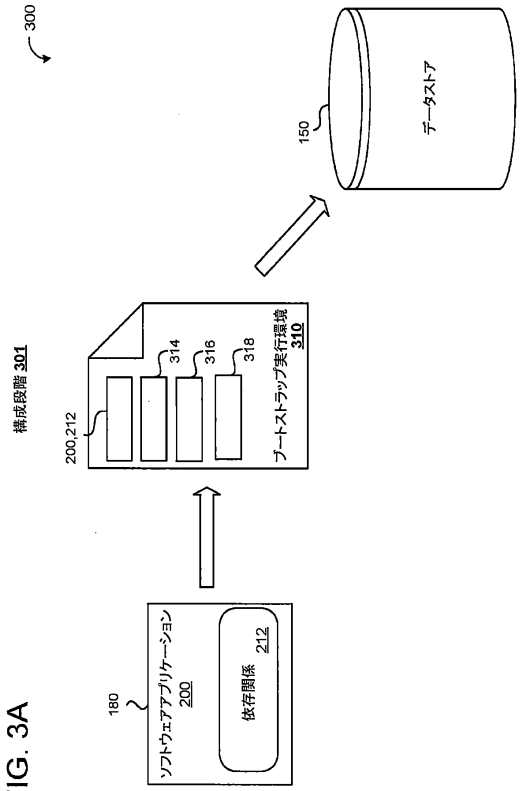

FIG. 3A

【図３Ｂ】

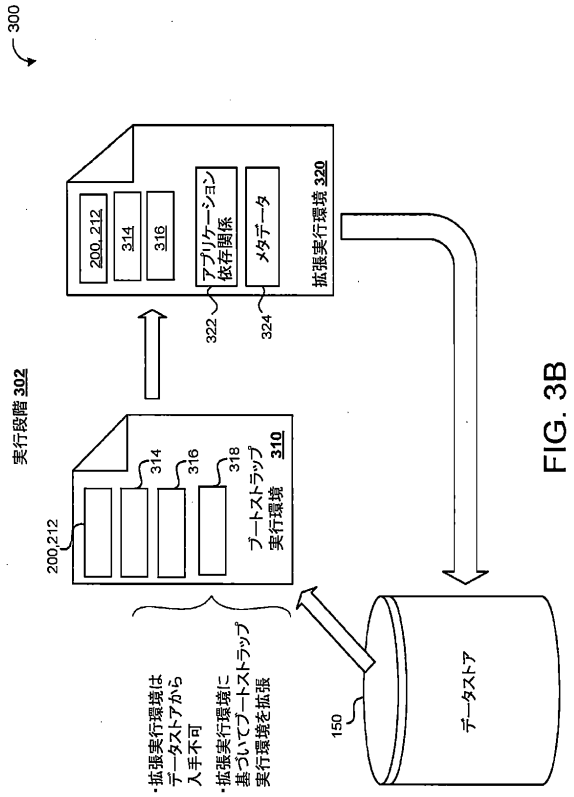

FIG. 3B

【図３Ｃ】

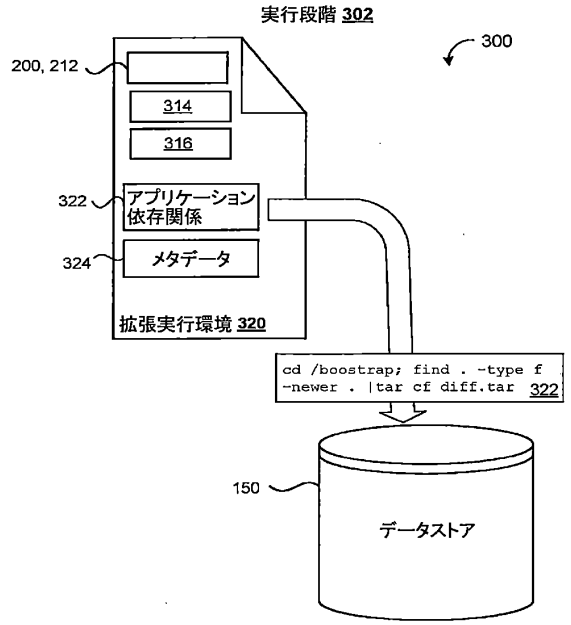

FIG. 3C

【図３Ｄ】

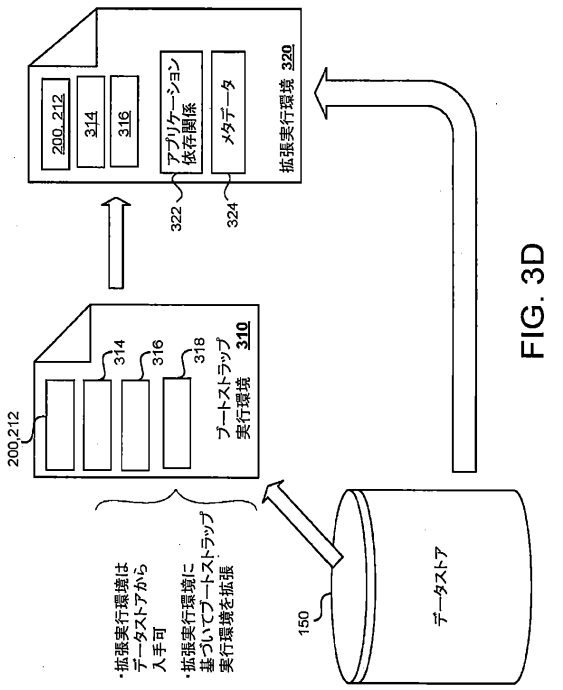

FIG. 3D

【図３Ｅ】

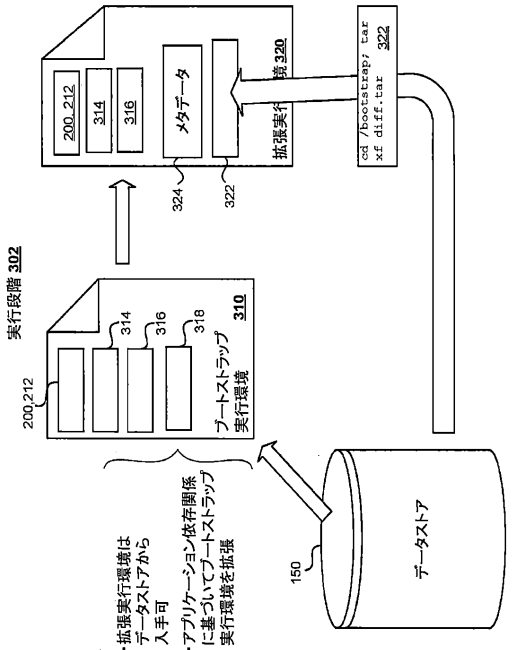

FIG. 3E

【図４】

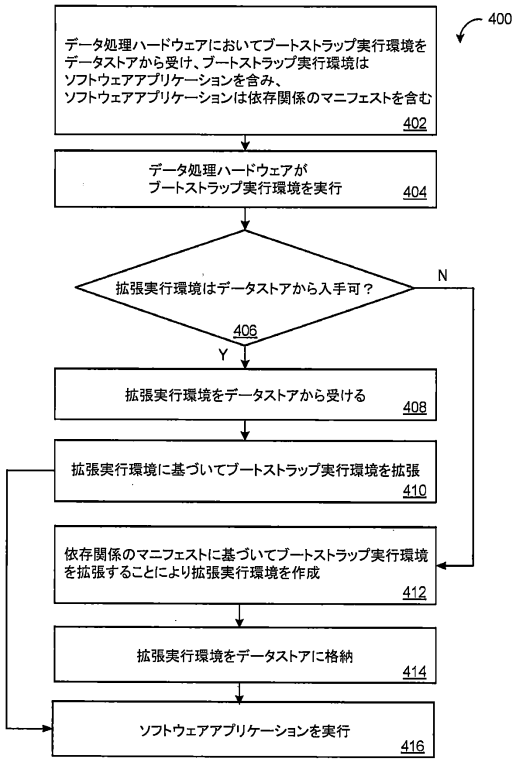

FIG. 4

【図５】

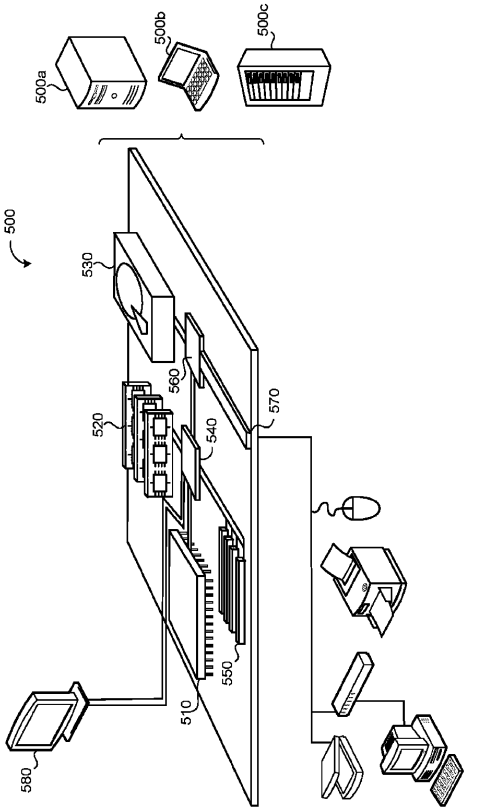

FIG. 5

フロントページの続き

(72)発明者　デイ，ライアン
アメリカ合衆国、９４０４３　カリフォルニア州、マウンテン・ビュー、アンフィシアター・パークウェイ、１６００

審査官　山本　俊介

(56)参考文献　米国特許第８２１９９８７（ＵＳ，Ｂ１）
特開２００８－２０９９８２（ＪＰ，Ａ）
国際公開第２０１９／１８１８６０（WO，Ａ１）
米国特許第９８２３９１５（ＵＳ，Ｂ１）
特開２００４－７８７５１（ＪＰ，Ａ）

(58)調査した分野　(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　８／６０－８／６５８
Ｇ０６Ｆ　９／４４－９／４４５