(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号
特許第7410269号
(P7410269)

(45)発行日　令和6年1月9日(2024.1.9)

(24)登録日　令和5年12月25日(2023.12.25)

(51)国際特許分類
Ｇ０６Ｆ　8/41　(2018.01)

ＦＩ
Ｇ０６Ｆ　8/41　１３０

請求項の数　26　（全31頁）

(21)出願番号　特願2022-500767(P2022-500767)
(86)(22)出願日　令和2年7月24日(2020.7.24)
(65)公表番号　特表2022-542007(P2022-542007A)
(43)公表日　令和4年9月29日(2022.9.29)
(86)国際出願番号　PCT/IB2020/057017
(87)国際公開番号　WO2021/019401
(87)国際公開日　令和3年2月4日(2021.2.4)
審査請求日　令和4年12月23日(2022.12.23)
(31)優先権主張番号　16/527,362
(32)優先日　令和1年7月31日(2019.7.31)
(33)優先権主張国・地域又は機関
米国(US)

(73)特許権者　390009531
インターナショナル・ビジネス・マシーンズ・コーポレーション
ＩＮＴＥＲＮＡＴＩＯＮＡＬ　ＢＵＳＩＮＥＳＳ　ＭＡＣＨＩＮＥＳ　ＣＯＲＰＯＲＡＴＩＯＮ
アメリカ合衆国１０５０４　ニューヨーク州　アーモンク　ニュー　オーチャード　ロード
Ｎｅｗ　Ｏｒｃｈａｒｄ　Ｒｏａｄ，　Ａｒｍｏｎｋ，　Ｎｅｗ　Ｙｏｒｋ　１０５０４，　Ｕｎｉｔｅｄ　Ｓｔａｔｅｓ　ｏｆ　Ａｍｅｒｉｃａ
(74)代理人　100112690
弁理士　太佐 種一

最終頁に続く

(54)【発明の名称】　テスト・ベクタを使用した高水準コンストラクトの最適化の自動検証

(57)【特許請求の範囲】
【請求項１】
高水準コンストラクトの最適化の自動検証のためのコンピュータ実施方法であって、
前記方法が、
コンパイラによって、第１の実行可能コードを、コンピュータ・プログラムをコンパイルすることにより生成すること
を含み、前記コンピュータ・プログラムが高水準コンストラクトを含み、
前記コンパイルすることが、
前記高水準コンストラクトに対する第１の一組の機械命令を生成すること、および
前記第１の実行可能コードに、前記高水準コンストラクトに対するコンパイル時情報を格納すること
を含み、
前記方法がさらに、
バイナリ・オプティマイザによって、前記第１の実行可能コードを最適化すること
を含み、
前記最適化することが、
前記第１の実行可能コードを、前記コンピュータ・プログラムの中間言語（ＩＬ）表現に変換すること
を含み、前記ＩＬ表現が、前記コンパイル時情報に基づく、前記高水準コンストラクトに対応する一組のＩＬ命令を含み、

前記最適化することがさらに、
　前記コンピュータ・プログラムの前記ＩＬ表現に基づいて第２の実行可能コードを生成すること
を含み、
前記第２の実行可能コードを生成することが、
　前記高水準コンストラクトの前記ＩＬ表現に基づいて、前記高水準コンストラクトに対する第２の一組の機械命令を生成すること、
　前記第１の一組の機械命令と前記第２の一組の機械命令との振る舞いが一致するとの判定に基づいて、前記第２の実行可能コードに前記第２の一組の機械命令を含めること、および
　前記第１の一組の機械命令と前記第２の一組の機械命令との振る舞いが一致しないとの判定に基づいて、前記第２の実行可能コードに前記第１の一組の機械命令を含めること
を含む、
コンピュータ実施方法。
【請求項２】
　前記第１の一組の機械命令と前記第２の一組の機械命令との振る舞いが一致するとの前記判定が充足可能性ソルバに基づく、
請求項１に記載のコンピュータ実施方法。
【請求項３】
　前記第１の一組の機械命令と前記第２の一組の機械命令との振る舞いが一致すると判定することが、
　複数のテストを含むテスト・ベクタを生成すること、
　前記テスト・ベクタの中のそれぞれのテストを、前記高水準コンストラクトに関連する前記第１の一組の機械命令および前記第２の一組の機械命令に対して実行すること、ならびに
　前記テスト・ベクタの中の前記それぞれのテストの出力を比較すること
を含み、
前記第１の一組の機械命令と前記第２の一組の機械命令とに関してそれぞれの前記出力が一致することに基づいて、前記振る舞いは一致すると判定される、
請求項１または２に記載のコンピュータ実施方法。
【請求項４】
　前記テスト・ベクタの中のテストを実行することが、
　前記第１の一組の機械命令に対する第１のラッパ関数を生成すること、
　実行する前記テストに応じた入力を用いて前記第１のラッパ関数を実行し、第１の一組の出力を収集すること、
　前記第２の一組の機械命令に対する第２のラッパ関数を生成すること、および
　実行する前記テストに応じた前記入力を用いて前記第２のラッパ関数を実行し、第２の一組の出力を収集すること
を含む、
請求項３に記載のコンピュータ実施方法。
【請求項５】
　前記第１のラッパ関数が第１の例外ハンドラでエンクローズされており、前記第２のラッパ関数が第２の例外ハンドラでエンクローズされている、
請求項４に記載のコンピュータ実施方法。
【請求項６】
　テストの第１の出力が前記第１の一組の機械命令に対する例外であること、および前記テストの第２の出力が前記第２の一組の機械命令に対する前記例外であることに基づいて、前記振る舞いは一致すると判定される、
請求項３ないし５のいずれかに記載のコンピュータ実施方法。
【請求項７】

前記コンパイル時情報が、前記高水準コンストラクトの識別、前記第１の一組の機械命令の位置、前記第１の一組の機械命令に対する入力および出力の位置を含む、
請求項１ないし６のいずれかに記載のコンピュータ実施方法。
【請求項８】
前記コンパイラが、コンパイラの第１のインスタンスであり、前記コンパイラの前記第１のインスタンスの設定とは異なる設定を有する前記コンパイラの第２のインスタンスが、前記第２の実行可能コードを生成する目的に使用される、
請求項１ないし７のいずれかに記載のコンピュータ実施方法。
【請求項９】
前記方法がさらに、一組の入力について、前記第１の一組の機械命令と前記第２の一組の機械命令との振る舞いが一致しないとの判定に基づいて、
前記第１の一組の機械命令と、前記第２の一組の機械命令と、前記高水準コンストラクトへの入力を調べる追加された関数とを含む第３の実行可能コードであり、前記入力が所定の一組の入力からのものであることに応答して前記第１の一組の機械命令を実行させ、そうでないことに応答して前記第２の一組の機械命令を実行させる、前記第３の実行可能コードを生成すること
を含む、
請求項１ないし８のいずれかに記載のコンピュータ実施方法。
【請求項１０】
システムであって、
メモリと、
前記メモリに結合されたプロセッサと
を備え、前記プロセッサが、高水準コンストラクトの最適化の自動検証のための方法を実行するように構成されており、
前記方法が、
コンパイラによって、第１の実行可能コードを、コンピュータ・プログラムをコンパイルすることにより生成すること
を含み、前記コンピュータ・プログラムが高水準コンストラクトを含み、
前記コンパイルすることが、
前記高水準コンストラクトに対する第１の一組の機械命令を生成すること、および
前記第１の実行可能コードに、前記高水準コンストラクトに対するコンパイル時情報を格納すること
を含み、
前記方法がさらに、
バイナリ・オプティマイザによって、前記第１の実行可能コードを最適化すること
を含み、
前記最適化することが、
前記第１の実行可能コードを中間言語（ＩＬ）表現に変換すること
を含み、前記ＩＬ表現が、前記コンパイル時情報に基づく、前記高水準コンストラクトに対応する一組のＩＬ命令を含み、
前記最適化することがさらに、
前記ＩＬ表現に基づいて第２の実行可能コードを生成すること
を含み、前記第２の実行可能コードが、前記高水準コンストラクトに対する第２の一組の機械命令を含み、
前記最適化することがさらに、
前記バイナリ・オプティマイザによって、前記第１の一組の機械命令と前記第２の一組の機械命令とを比較すること、および
前記第１の一組の機械命令と前記第２の一組の機械命令との振る舞いが一致するとの判定に基づいて、前記第１の実行可能コードを前記第２の実行可能コードに置き換えること
を含む、

システム。
【請求項１１】
前記第１の一組の機械命令と前記第２の一組の機械命令との前記振る舞いが一致するとの前記判定が充足可能性ソルバに基づく、
請求項１０に記載のシステム。
【請求項１２】
前記第１の一組の機械命令と前記第２の一組の機械命令との振る舞いが一致すると判定することが、
複数のテストを含むテスト・ベクタを生成すること、
前記テスト・ベクタの中のそれぞれのテストを、前記高水準コンストラクトに関連する前記第１の一組の機械命令および前記第２の一組の機械命令に対して実行すること、ならびに
前記テスト・ベクタの中の前記それぞれのテストの出力を比較すること
を含み、
前記第１の一組の機械命令と前記第２の一組の機械命令とに関してそれぞれの前記出力が一致することに基づいて、前記振る舞いは一致すると判定される、
請求項１０または１１に記載のシステム。
【請求項１３】
前記テスト・ベクタの中のテストを実行することが、
前記第１の一組の機械命令に対する第１のラッパ関数を生成すること、
実行する前記テストに応じた入力を用いて前記第１のラッパ関数を実行し、第１の一組の出力を収集すること、
前記第２の一組の機械命令に対する第２のラッパ関数を生成すること、および
実行する前記テストに応じた前記入力を用いて前記第２のラッパ関数を実行し、第２の一組の出力を収集すること
を含む、
請求項１２に記載のシステム。
【請求項１４】
前記第１のラッパ関数が第１の例外ハンドラでエンクローズされており、前記第２のラッパ関数が第２の例外ハンドラでエンクローズされている、
請求項１３に記載のシステム。
【請求項１５】
テストの第１の出力が前記第１の一組の機械命令に対する例外であること、および前記テストの第２の出力が前記第２の一組の機械命令に対する前記例外であることに基づいて、前記振る舞いは一致すると判定される、
請求項１２ないし１４のいずれかに記載のシステム。
【請求項１６】
前記コンパイル時情報が、前記高水準コンストラクトの識別、前記第１の一組の機械命令の位置、前記第１の一組の機械命令に対する入力および出力の位置を含む、
請求項１０ないし１５のいずれかに記載のシステム。
【請求項１７】
前記コンパイラが、コンパイラの第１のインスタンスであり、前記コンパイラの前記第１のインスタンスの設定とは異なる設定を有する前記コンパイラの第２のインスタンスが、前記第２の実行可能コードを生成する目的に使用される、
請求項１０ないし１６のいずれかに記載のシステム。
【請求項１８】
前記方法がさらに、一組の入力について、前記第１の一組の機械命令と前記第２の一組の機械命令との振る舞いが一致しないとの判定に基づいて、
前記第１の一組の機械命令と、前記第２の一組の機械命令と、前記高水準コンストラクトへの入力を調べる追加された関数とを含む第３の実行可能コードであり、前記入力が所

定の一組の入力からのものであることに応答して前記第１の一組の機械命令を実行させ、そうでないことに応答して前記第２の一組の機械命令を実行させる、前記第３の実行可能コードを生成すること
を含む、
請求項１０ないし１７のいずれかに記載のシステム。
【請求項１９】
前記第１の一組の機械命令と前記第２の一組の機械命令との振る舞いが一致すると判定することが、
複数のテストを含むテスト・ベクタを生成すること、
前記テスト・ベクタの中のそれぞれのテストを、前記高水準コンストラクトに関連する前記第１の一組の機械命令および前記第２の一組の機械命令に対して実行すること、ならびに
前記テスト・ベクタの中の前記それぞれのテストの出力を比較すること
を含み、
前記第１の一組の機械命令と前記第２の一組の機械命令とに関してそれぞれの前記出力が一致することに基づいて、前記振る舞いは一致すると判定される、
請求項１８に記載のシステム。
【請求項２０】
高水準コンストラクトの最適化の自動検証のためのコンピュータ・プログラムであって、前記コンピュータ・プログラムはコンピュータに、
請求項１ないし９いずれか一項に記載の方法を実行させる、コンピュータ・プログラム。
【請求項２１】
請求項１ないし９いずれか一項に記載の方法を実行させるためコンピュータ・プログラムを記録したコンピュータ読み取り可能な記録媒体。
【請求項２２】
高水準コンストラクトの最適化の自動検証のためのコンピュータ実施方法であって、
前記方法が、
コンパイラによって、第１の実行可能コードを、コンピュータ・プログラムをコンパイルすることにより生成すること
を含み、前記コンピュータ・プログラムが高水準コンストラクトを含み、
前記コンパイルすることが、
前記高水準コンストラクトに対する第１の一組の機械命令を生成すること、および
前記第１の実行可能コードに、前記高水準コンストラクトに対するコンパイル時情報を格納すること
を含み、
前記方法がさらに、
オプティマイザによって、前記第１の実行可能コードを最適化すること
を含み、
前記最適化することが、
前記第１の実行可能コードを中間言語（ＩＬ）表現に変換すること
を含み、前記ＩＬ表現が、前記コンパイル時情報に基づく、前記高水準コンストラクトに対応する一組のＩＬ命令を含み、
前記最適化することがさらに、
前記ＩＬ表現に基づいて第２の実行可能コードを生成すること
を含み、前記第２の実行可能コードが、前記高水準コンストラクトに対する第２の一組の機械命令を含み、
前記最適化することがさらに、
一組の入力について、前記第１の一組の機械命令と前記第２の一組の機械命令との振る舞いが一致しないとの判定に基づいて、
前記第１の一組の機械命令と、前記第２の一組の機械命令と、前記高水準コンストラク

トへの入力を調べる追加された関数とを含む第３の実行可能コードであり、前記入力が所定の一組の入力からのものであることに応答して前記第１の一組の機械命令を実行させ、そうでないことに応答して前記第２の一組の機械命令を実行させる、前記第３の実行可能コードを生成すること
を含む、
コンピュータ実施方法。
【請求項２３】
前記第１の一組の機械命令と前記第２の一組の機械命令との振る舞いが一致すると判定することが、
複数のテストを含むテスト・ベクタを生成すること、
前記テスト・ベクタの中のそれぞれのテストを、前記高水準コンストラクトに関連する前記第１の一組の機械命令および前記第２の一組の機械命令に対して実行すること、ならびに
前記テスト・ベクタの中の前記それぞれのテストの出力を比較すること
を含み、
前記第１の一組の機械命令と前記第２の一組の機械命令とに関してそれぞれの前記出力が一致することに基づいて、前記振る舞いは一致すると判定される、
請求項２２に記載のコンピュータ実施方法。
【請求項２４】
前記テスト・ベクタの中のテストを実行することが、
前記第１の一組の機械命令に対する第１のラッパ関数を生成すること、
実行する前記テストに応じた入力を用いて前記第１のラッパ関数を実行し、第１の一組の出力を収集すること、
前記第２の一組の機械命令に対する第２のラッパ関数を生成すること、および
実行する前記テストに応じた前記入力を用いて前記第２のラッパ関数を実行し、第２の一組の出力を収集すること
を含む、
請求項２３に記載のコンピュータ実施方法。
【請求項２５】
システムであって、
メモリと、
前記メモリに結合されたプロセッサと
を備え、前記プロセッサが、高水準コンストラクトの最適化の自動検証のための方法を実行するように構成されており、
前記方法が、
コンパイラによって、第１の実行可能コードを、コンピュータ・プログラムをコンパイルすることにより生成すること
を含み、前記コンピュータ・プログラムが高水準コンストラクトを含み、
前記コンパイルすることが、
前記高水準コンストラクトに対する第１の一組の機械命令を生成すること、および
前記第１の実行可能コードに、前記高水準コンストラクトに対するコンパイル時情報を格納すること
を含み、
前記方法がさらに、
オプティマイザによって、前記第１の実行可能コードを最適化すること
を含み、
前記最適化することが、
前記第１の実行可能コードを中間言語（ＩＬ）表現に変換すること
を含み、前記ＩＬ表現が、前記コンパイル時情報に基づく、前記高水準コンストラクトに対応する一組のＩＬ命令を含み、

前記最適化することがさらに、
　前記ＩＬ表現に基づいて第２の実行可能コードを生成すること
を含み、前記第２の実行可能コードが、前記高水準コンストラクトに対する第２の一組の機械命令を含み、
前記最適化することがさらに、
　一組の入力について、前記第１の一組の機械命令と前記第２の一組の機械命令との振る舞いが一致しないとの判定に基づいて、
　前記第１の一組の機械命令と、前記第２の一組の機械命令と、前記高水準コンストラクトへの入力を調べる追加された関数とを含む第３の実行可能コードであり、前記入力が所定の一組の入力からのものであることに応答して前記第１の一組の機械命令を実行させ、そうでないことに応答して前記第２の一組の機械命令を実行させる、前記第３の実行可能コードを生成すること
を含む、
システム。
【請求項２６】
　前記第１の一組の機械命令と前記第２の一組の機械命令との振る舞いが一致すると判定することが、
　複数のテストを含むテスト・ベクタを生成すること、
　前記テスト・ベクタの中のそれぞれのテストを、前記高水準コンストラクトに関連する前記第１の一組の機械命令および前記第２の一組の機械命令に対して実行すること、ならびに
　前記テスト・ベクタの中の前記それぞれのテストの出力を比較すること
を含み、
前記第１の一組の機械命令と前記第２の一組の機械命令とに関してそれぞれの前記出力が一致することに基づいて、前記振る舞いは一致すると判定される、
請求項２５に記載のシステム。
【発明の詳細な説明】
【技術分野】
【０００１】
　本開示は一般にコンピューティング技術に関し、より詳細には、実行可能プログラム・コードの性能を向上させるための方法、コンピュータ・プログラム製品、およびデータ処理システムに関する。
【背景技術】
【０００２】
　コンピュータ・ソフトウェアは、データ処理システムによって実行される一組の命令を含む。コンピュータ・ソフトウェアは、データ処理システムが、例えばワード・プロセッシング・デバイス、スプレッドシート・デバイス、またはインターネット・ブラウジング・デバイス、データ・レポジトリなどとして機能することを可能にする命令を提供する。コンピュータ・ソフトウェアを使用することができる幅広い多様な異なるデータ処理システムが存在する。それに応じて、本明細書で使用されるとき、用語「データ処理システム」は幅広い意味を有することが意図されており、この用語は、パーソナル・コンピュータ、ラップトップ・コンピュータ、パームトップ・コンピュータ、ハンドヘルド・コンピュータ、ネットワーク・コンピュータ、サーバ、メインフレーム、ワークステーション、移動電話および同種の無線デバイス、パーソナル・デジタル・アシスタント、ならびにコンピュータ・ソフトウェアをインストールすることができる他の電子デバイスを含むことがある。用語「コンピュータ」、「コンピュータ・ソフトウェア」、「コンピュータ・プログラム」、「コンピュータ・プログラミング」、「ソフトウェア」、「ソフトウェア・プログラム」および関連用語も、同様に幅広い意味を有することが意図されている。
【０００３】
　一般に、現代のコンピュータ・ソフトウェアは元来、プログラマが理解できる構文コン

ストラクト（syntactic construct）を使用してソフトウェアの中に実装する命令を表現する「高水準（high level）」コンピュータ・プログラミング言語で書かれている。高水準コンピュータ・プログラミング言語の構文コンストラクトは、プログラマがコンピュータ・ソフトウェアを作成することをより容易にするため、高水準コンピュータ・プログラミング言語は有用である。コンピュータ・ソフトウェアの作成が容易になるのは、データ処理システムが直接に理解するであろう言語でプログラマが命令を書く必要がないためである。データ処理システムが直接に理解するであろう言語と任意の人間の言語とはほとんどまたは全く似ていないため、データ処理システムが直接に理解するであろう言語で命令を書く方がはるかに難しいであろう。

【０００４】

しかしながら、データ処理システムは一般に、高水準コンピュータ・プログラミング言語で書かれた命令を直接に理解および実施することができない。したがって、高水準コンピュータ・プログラミング言語で書かれたコンピュータ・プログラムは、データ処理システムによって使用され得る前に、最初に、ターゲット・データ処理システムによって理解される言語に「コンパイル」される。コンパイルは、普通は「コンパイラ」と呼ばれるコンピュータ・プログラムによって実行されるプロセスであり、このプロセスでは、本質的に、高水準コンピュータ・プログラミング言語の構文コンストラクトが、ターゲット・データ処理システムによって理解される言語で書かれた命令に（ことによると中間ソフトウェア層を介して）翻訳される。「コンパイル（compilingまたはcompilation）」プロセスの結果は、「実行可能コード」または「バイナリ・コード」として知られており、これは、データ処理システムが直接にまたは中間ソフトウェア層によって実行することができるコンピュータ・プログラム・コードを意味し、ターゲット・データ処理システムによって理解される命令は「機械」命令を意味する。

【０００５】

高水準コンピュータ・プログラミング言語は、幅広い２つのタイプの言語、すなわち静的にコンパイルされる言語（statically compiled language）（以後、静的コンパイル言語）および動的にコンパイルされる言語（dynamically compiled language）（以後、動的コンパイル言語）のうちの一方に含まれると考えることができる。静的コンパイル言語では、コードが実行される前にコンパイル・プロセスが一度実行される。コンパイルの結果は、中間ソフトウェア層なしでデータ処理システムによって直接に実施することができる実行可能コードである。静的コンパイル言語はＣ、Ｃ＋＋、ＦＯＲＴＲＡＮ、ＰＬ/Ｉ、ＣＯＢＯＬ、およびＡｄａを含む。Ｊａｖａ（Ｒ）（ＴＭ）などの動的コンパイル言語では、ソース・コードが最初に、Ｊａｖａ（Ｒ）バーチャル・マシン（ＪＶＭ）などの中間ソフトウェア層によって実施することができる中間形態にコンパイルされる。Ｊａｖａ（Ｒ）では、この中間形態が「バイトコード」として知られている。必ずというわけではないが、中間ソフトウェア層は通常、コンピュータ・プログラムが実行されるたびに追加のコンパイルを実行して、普通は、ソース・コードの中間形態を、データ処理システムによって直接に実行することができる実行可能コードに翻訳する。Ｊａｖａ（Ｒ）およびＪａｖａ（Ｒ）をベースにした全ての商標およびロゴは、Ｏｒａｃｌｅ（Ｒ）もしくはその系列会社またはその両方の商標または登録商標である。

【０００６】

したがって、当技術分野では、上述の課題を解決することが求められている。

【発明の概要】

【０００７】

第１の態様から見ると、本発明は、高水準コンストラクトの最適化の自動検証のためのコンピュータ実施方法を提供し、この方法は、コンパイラによって、第１の実行可能コードを、コンピュータ・プログラムをコンパイルすることにより生成することを含み、このコンピュータ・プログラムは高水準コンストラクトを含み、コンパイルすることは、高水準コンストラクトに対する第１の一組の機械命令を生成すること、および第１の実行可能コードに、高水準コンストラクトに対するコンパイル時情報（compile-time informatio

n）を格納することを含み、この方法はさらに、バイナリ・オプティマイザ（binary optimizer）によって、第１の実行可能コードを最適化することを含み、最適化することは、第１の実行可能コードを、コンピュータ・プログラムの中間言語（intermediate language）（ＩＬ）表現に変換することを含み、このＩＬ表現は、コンパイル時情報に基づく、高水準コンストラクトに対応する一組のＩＬ命令を含み、最適化することはさらに、コンピュータ・プログラムのＩＬ表現に基づいて第２の実行可能コードを生成することを含み、第２の実行可能コードを生成することは、高水準コンストラクトのＩＬ表現に基づいて、高水準コンストラクトに対する第２の一組の機械命令を生成すること、第１の一組の機械命令と第２の一組の機械命令との振る舞い（behavior）が一致するとの判定に基づいて、第２の実行可能コードに第２の一組の機械命令を含めること、および第１の一組の機械命令と第２の一組の機械命令との振る舞いが一致しないとの判定に基づいて、第２の実行可能コードに第１の一組の機械命令を含めることを含む。

【０００８】

別の一態様から見ると、本発明はシステムを提供し、このシステムは、メモリと、このメモリに結合されたプロセッサとを備え、このプロセッサは、高水準コンストラクトの最適化の自動検証のための方法を実行するように構成されており、この方法は、コンパイラによって、第１の実行可能コードを、コンピュータ・プログラムをコンパイルすることにより生成することを含み、このコンピュータ・プログラムは高水準コンストラクトを含み、コンパイルすることは、高水準コンストラクトに対する第１の一組の機械命令を生成すること、および第１の実行可能コードに、高水準コンストラクトに対するコンパイル時情報を格納することを含み、この方法はさらに、バイナリ・オプティマイザによって、第１の実行可能コードを最適化することを含み、最適化することは、第１の実行可能コードを中間言語（ＩＬ）表現に変換することを含み、このＩＬ表現は、コンパイル時情報に基づく、高水準コンストラクトに対応する一組のＩＬ命令を含み、最適化することはさらに、ＩＬ表現に基づいて第２の実行可能コードを生成することを含み、第２の実行可能コードは、高水準コンストラクトに対する第２の一組の機械命令を含み、最適化することはさらに、バイナリ・オプティマイザによって、第１の一組の機械命令と第２の一組の機械命令とを比較すること、および第１の一組の機械命令と第２の一組の機械命令との振る舞いが一致するとの判定に基づいて、第１の実行可能コードを第２の実行可能コードに置き換えることを含む。

【０００９】

別の一態様から見ると、本発明は、高水準コンストラクトの最適化の自動検証のためのコンピュータ・プログラム製品を提供し、このコンピュータ・プログラム製品は、処理回路によって可読のコンピュータ可読ストレージ媒体であり、本発明のステップを実行するための方法を実行するためにこの処理回路が実行するための命令を記憶した、コンピュータ可読ストレージ媒体を含む。

【００１０】

別の一態様から見ると、本発明は、コンピュータ可読媒体上に記憶された、デジタル・コンピュータの内部メモリにロード可能なコンピュータ・プログラムであって、このプログラムがコンピュータ上で実行されたときに本発明のステップを実行するためのソフトウェア・コード部分を含む、コンピュータ・プログラムを提供する。

【００１１】

別の一態様から見ると、本発明は、プログラム命令が実装されたコンピュータ可読ストレージ媒体を含むコンピュータ・プログラム製品を提供し、このプログラム命令は、高水準コンストラクトの最適化の自動検証のための方法をコンピュータ・プロセッサに実行させるために、このコンピュータ・プロセッサによって実行可能であり、この方法は、コンパイラによって、第１の実行可能コードを、コンピュータ・プログラムをコンパイルすることにより生成することを含み、このコンピュータ・プログラムは高水準コンストラクトを含み、コンパイルすることは、高水準コンストラクトに対する第１の一組の機械命令を生成すること、および第１の実行可能コードに、高水準コンストラクトに対するコンパイ

ル時情報を格納することを含み、この方法はさらに、バイナリ・オプティマイザによって、第１の実行可能コードを最適化することを含み、最適化することは、第１の実行可能コードを、コンピュータ・プログラムの中間言語（ＩＬ）表現に変換することを含み、このＩＬ表現は、コンパイル時情報に基づく、高水準コンストラクトに対応する一組のＩＬ命令を含み、最適化することはさらに、コンピュータ・プログラムのＩＬ表現に基づいて第２の実行可能コードを生成することを含み、第２の実行可能コードを生成することは、高水準コンストラクトのＩＬ表現に基づいて、高水準コンストラクトに対する第２の一組の機械命令を生成すること、第１の一組の機械命令と第２の一組の機械命令との振る舞いが一致するとの判定に基づいて、第２の実行可能コードに第２の一組の機械命令を含めること、および第１の一組の機械命令と第２の一組の機械命令との振る舞いが一致しないとの判定に基づいて、第２の実行可能コードに第１の一組の機械命令を含めることを含む。

【００１２】

別の一態様から見ると、本発明は、高水準コンストラクトの最適化の自動検証のためのコンピュータ実施方法を提供し、この方法は、コンパイラによって、第１の実行可能コードを、コンピュータ・プログラムをコンパイルすることにより生成することを含み、このコンピュータ・プログラムは高水準コンストラクトを含み、コンパイルすることは、高水準コンストラクトに対する第１の一組の機械命令を生成すること、および第１の実行可能コードに、高水準コンストラクトに対するコンパイル時情報を格納することを含み、この方法はさらに、オプティマイザによって、第１の実行可能コードを最適化することを含み、最適化することは、第１の実行可能コードを中間言語（ＩＬ）表現に変換することを含み、このＩＬ表現は、コンパイル時情報に基づく、高水準コンストラクトに対応する一組のＩＬ命令を含み、最適化することはさらに、ＩＬ表現に基づいて第２の実行可能コードを生成することを含み、第２の実行可能コードは、高水準コンストラクトに対する第２の一組の機械命令を含み、最適化することはさらに、一組の入力について、第１の一組の機械命令と第２の一組の機械命令との振る舞いが一致しないとの判定に基づいて、第１の一組の機械命令と、第２の一組の機械命令と、高水準コンストラクトへの入力を調べる追加された関数とを含む第３の実行可能コードであり、この入力が所定の一組の入力からのものであることに応答して第１の一組の機械命令を実行させ、そうでないことに応答して第２の一組の機械命令を実行させる、第３の実行可能コードを生成することを含む。

【００１３】

別の一態様から見ると、本発明はシステムを提供し、このシステムは、メモリと、このメモリに結合されたプロセッサとを備え、このプロセッサは、高水準コンストラクトの最適化の自動検証のための方法を実行するように構成されており、この方法は、コンパイラによって、第１の実行可能コードを、コンピュータ・プログラムをコンパイルすることにより生成することを含み、このコンピュータ・プログラムは高水準コンストラクトを含み、コンパイルすることは、高水準コンストラクトに対する第１の一組の機械命令を生成すること、および第１の実行可能コードに、高水準コンストラクトに対するコンパイル時情報を格納することを含み、この方法はさらに、オプティマイザによって、第１の実行可能コードを最適化することを含み、最適化することは、第１の実行可能コードを中間言語（ＩＬ）表現に変換することを含み、このＩＬ表現は、コンパイル時情報に基づく、高水準コンストラクトに対応する一組のＩＬ命令を含み、最適化することはさらに、ＩＬ表現に基づいて第２の実行可能コードを生成することを含み、第２の実行可能コードは、高水準コンストラクトに対する第２の一組の機械命令を含み、最適化することはさらに、一組の入力について、第１の一組の機械命令と第２の一組の機械命令との振る舞いが一致しないとの判定に基づいて、第１の一組の機械命令と、第２の一組の機械命令と、高水準コンストラクトへの入力を調べる追加された関数とを含む第３の実行可能コードであり、この入力が所定の一組の入力からのものであることに応答して第１の一組の機械命令を実行させ、そうでないことに応答して第２の一組の機械命令を実行させる、第３の実行可能コードを生成することを含む。

【００１４】

本発明の１つまたは複数の実施形態によれば、高水準コンストラクトの最適化の自動検証のためのコンピュータ実施方法は、コンパイラによって、第１の実行可能コードを、コンピュータ・プログラムをコンパイルすることにより生成することを含む。このコンピュータ・プログラムは高水準コンストラクトを含み、コンパイルすることは、高水準コンストラクトに対する第１の一組の機械命令を生成すること、および第１の実行可能コードに、高水準コンストラクトに対するコンパイル時情報を格納することを含む。この方法はさらに、バイナリ・オプティマイザによって、第１の実行可能コードを最適化することを含む。最適化することは、第１の実行可能コードを、コンピュータ・プログラムの中間言語（ＩＬ）表現に変換することを含む。このＩＬ表現は、コンパイル時情報に基づく、高水準コンストラクトに対応する一組のＩＬ命令を含む。最適化することはさらに、コンピュータ・プログラムのＩＬ表現に基づいて第２の実行可能コードを生成することを含み、第２の実行可能コードを生成することは、高水準コンストラクトのＩＬ表現に基づいて、高水準コンストラクトに対する第２の一組の機械命令を生成することを含む。第１の一組の機械命令と第２の一組の機械命令との振る舞いが一致するとの判定に基づいて、第２の実行可能コードに第２の一組の機械命令が含められる。あるいは、第１の一組の機械命令と第２の一組の機械命令との振る舞いが一致しないとの判定に基づいて、第２の実行可能コードに第１の一組の機械命令が含められる。

【００１５】

１つまたは複数の例では、第１の一組の機械命令と第２の一組の機械命令との振る舞いが一致するとの判定が充足可能性ソルバ（satisfiability solver）に基づく。第１の一組の機械命令と第２の一組の機械命令との振る舞いが一致すると判定することは、複数のテストを含むテスト・ベクタ（test vector）を生成すること、テスト・ベクタの中のそれぞれのテストを、高水準コンストラクトに関連する第１の一組の機械命令および第２の一組の機械命令に対して実行することを含む。判定することはさらに、テスト・ベクタの中のそれぞれのテストの出力を比較することを含み、第１の一組の機械命令と第２の一組の機械命令とに関してそれぞれの出力が一致することに基づいて、振る舞いは一致すると判定される。

【００１６】

テスト・ベクタの中のテストを実行することは、第１の一組の機械命令に対する第１のラッパ関数（wrapper function）を生成すること、および実行するテストに応じた入力を用いて第１のラッパ関数を実行し、第１の一組の出力を収集することを含む。さらに、テストを実行することは、第２の一組の機械命令に対する第２のラッパ関数を生成すること、および実行するテストに応じた入力を用いて第２のラッパ関数を実行することを含む。テストを実行することはさらに、第２の一組の出力を収集することを含む。１つまたは複数の例では、第１のラッパ関数が第１の例外ハンドラでエンクローズされており（enclosed in）、第２のラッパ関数が第２の例外ハンドラでエンクローズされている。１つまたは複数の例では、テストの第１の出力が第１の一組の機械命令に対する例外であること、および前記テストの第２の出力が第２の一組の機械命令に対する前記例外であることに基づいて、振る舞いは一致すると判定される。

【００１７】

１つまたは複数の例では、高水準コンストラクトに対するコンパイル時情報を格納することが、高水準コンストラクトに対応する前記テストに入力を格納することを含む。

【００１８】

１つまたは複数の例では、コンパイル時情報が、高水準コンストラクトの識別（identity）、第１の一組の機械命令の位置、第１の一組の機械命令に対する入力および出力の位置を含む。

【００１９】

１つまたは複数の例では、第１のコンパイラが、コンパイラの第１のインスタンス（instance）であり、第１のコンパイラの設定とは異なる設定を有する前記コンパイラの第２のインスタンスが、第２の実行可能コードを生成する目的に使用される。

【００２０】
本発明の１つまたは複数の実施形態によれば、高水準コンストラクトの最適化の自動検証のための方法は、コンパイラによって、第１の実行可能コードを、コンピュータ・プログラムをコンパイルすることにより生成することを含み、このコンピュータ・プログラムは高水準コンストラクトを含む。コンパイルすることは、高水準コンストラクトに対する第１の一組の機械命令を生成すること、および第１の実行可能コードに、高水準コンストラクトに対するコンパイル時情報を格納することを含む。この方法はさらに、オプティマイザによって、第１の実行可能コードを最適化することを含む。最適化することは、第１の実行可能コードをプログラムの中間言語（ＩＬ）表現に変換することを含み、この変換は、高水準コンストラクトを、コンパイル時情報に基づいて、高水準コンストラクトのＩＬ表現に変換することを含む。最適化することは、プログラムのＩＬ表現に基づいて第２の実行可能コードを生成することを含み、この第２の実行可能コードは、高水準コンストラクトのＩＬ表現に基づく、高水準コンストラクトに対する第２の一組の機械命令を含む。最適化することはさらに、一組の入力について、第１の一組の機械命令と第２の一組の機械命令との振る舞いが一致しないとの判定に基づいて、第１の一組の機械命令と、第２の一組の機械命令と、高水準コンストラクトへの入力を調べる追加された関数とを含む第３の実行可能コードであり、この入力が所定の一組の入力からのものであることに応答して第１の一組の機械命令を実行させ、そうでないことに応答して第２の一組の機械命令を実行させる、第３の実行可能コードを生成することを含む。
【００２１】
１つまたは複数の例では、第１の一組の機械命令と第２の一組の機械命令との振る舞いが一致すると判定することが、複数のテストを含むテスト・ベクタを生成すること、およびテスト・ベクタの中のそれぞれのテストを、高水準コンストラクトに関連する第１の一組の機械命令および第２の一組の機械命令に対して実行することを含む。テスト・ベクタの中のそれぞれのテストの出力が比較され、第１の一組の機械命令と第２の一組の機械命令とに関してそれぞれの出力が一致することに基づいて、振る舞いは一致すると判定される。さらに、テスト・ベクタの中のテストを実行することは、第１の一組の機械命令に対する第１のラッパ関数を生成すること、実行するテストに応じた入力を用いて第１のラッパ関数を実行すること、および第１の一組の出力を収集することを含む。さらに、テストを実行することは、第２の一組の機械命令に対する第２のラッパ関数を生成すること、実行するテストに応じた入力を用いて第２のラッパ関数を実行すること、および第２の一組の出力を収集することを含む。
【００２２】
上述の特徴はさらに、少なくともシステム、コンピュータ・プログラム製品、および機械によって提供することができる。
【００２３】
本発明の技術によって、追加の特徴および利点が実現される。本明細書には、本発明の他の実施形態および態様が詳細に記載されており、それらは、請求の発明の一部と見なされる。本発明ならびに本発明の利点および特徴をより十分に理解するため、以下の説明および図面を参照されたい。
【００２４】
本発明と見なされる主題は、本明細書の末尾の特許請求の範囲に具体的に示されており、明確に主張されている。本発明の上記の特徴および利点ならびにその他の特徴および利点は、添付図面とともに解釈される以下の詳細な説明から明らかである。
【図面の簡単な説明】
【００２５】
【図１】本発明の１つまたは複数の実施形態による、クラウド・コンピューティング環境を示す図である。
【図２】本発明の１つまたは複数の実施形態による、クラウド・コンピュータ環境の抽象化モデル層を示す図である。

【図３】本発明の１つまたは複数の実施形態を実施するのに利用することができる例示的なコンピュータ処理システムを示すブロック図である。
【図４】本発明の１つまたは複数の実施形態による、最適化システムのブロック図である。
【図５】本発明の１つまたは複数の実施形態による、テスト・ベクタを使用して高水準コンストラクトの最適化を自動的に検証する方法の流れ図である。
【図６】本発明の１つまたは複数の実施形態による、どの実行可能コードを第２の実行可能プログラムに含めるのかを決定するための方法の流れ図である。
【発明を実施するための形態】
【００２６】
本明細書に示された図は例示のためのものである。本発明の範囲を逸脱しない、図または図に記載された操作に対する多くの変形態様が存在し得る。例えば、動作を異なる順序で実行することができ、または動作を追加、削除もしくは変更することができる。また、用語「結合された」およびその変異語は、２つの要素間に通信経路を有することを示し、要素間に介在要素／接続がない要素間の直接接続を含意しない。これらの変形態様は全て本明細書の一部と見なされる。
【００２７】
添付図および記載された実施形態の以下の詳細な説明では、図に示されたさまざまな要素が２桁または３桁の参照符号を有する。少数の例外を除き、それぞれの参照符号の左端の数字はその要素が最初に示された図に対応する。
【００２８】
本明細書では、本発明のさまざまな実施形態が関連図を参照して説明される。本発明の範囲を逸脱することなく本発明の代替実施形態を考案することができる。以下の説明および図面には、要素間のさまざまな接続および位置関係（例えば上、下、隣りなど）が示されている。これらの接続もしくは位置関係またはその両方は、特に指定されていない限り、直接的なものであることまたは間接的なものであることができ、この点に関して本発明は限定を意図したものではない。したがって、実体（entity）の結合は、直接結合または間接結合のいずれかであることができ、実体間の位置関係は、直接的位置関係または間接的位置関係であることができる。さらに、本明細書には詳細に記載されていない追加のステップまたは機能を有する、より包括的な手順またはプロセスに、本明細書に記載されたさまざまなタスクおよびプロセス・ステップを組み込むことができる。
【００２９】
特許請求の範囲および本明細書の解釈のために、以下の定義および略語が使用される。本明細書で使用されるとき、用語「備える（comprises）」、「備えている（comprising）」、「含む（includes）」、「含んでいる（including）」、「有する（has）」、「有している（having）」、「含有する（contains）」もしくは「含有している（containing）」、またはこれらの用語の任意の他の変異語は、非排他的包含（non-exclusive inclusion）をカバーすることが意図されている。例えば、要素のリストを含む組成物、混合物、プロセス、方法、物品、または装置は、必ずしもそれらの要素だけに限定されるわけではなく、明示的にはリストに入れられていない他の要素、あるいはこのような組成物、混合物、プロセス、方法、物品、または装置に固有の他の要素を含み得る。
【００３０】
さらに、本明細書では、用語「例示的な」が、「一例、事例、または実例として役立つ」ことを意味するものとして使用されている。本明細書に「例示的」として記載された任意の実施形態または設計は必ずしも、他の実施形態または設計よりも好ましいまたは有利であるとは解釈されない。用語「少なくとも１つの」および「１つまたは複数の」は、１以上の任意の整数、すなわち１、２、３、４などを含むと理解してもよい。用語「複数の」は、２以上の任意の整数、すなわち２、３、４、５などを含むと理解してもよい。用語「接続」は、間接「接続」と直接「接続」の両方を含むことがある。
【００３１】
用語「約」、「実質的に」、「およそ」、およびこれらの用語の変異語は、特定の数量

の大きさに関連した、本出願の提出時に利用可能な機器に基づく誤差の程度を含むことが意図されている。例えば、「約」は、所与の値の±８％または５％または２％の範囲を含み得る。

【００３２】

簡潔にするため、本明細書では、本発明の態様を製作および使用することに関係した従来の技術が、詳細に説明されることもあり、または詳細には説明されないこともある。特に、本明細書に記載されたさまざまな技術的特徴を実施するためのコンピュータ・システムおよび特定のコンピュータ・プログラムのさまざまな態様はよく知られている。したがって、本明細書では、簡潔にするために、よく知られたシステムもしくはプロセスまたはその両方の詳細を提供することなく、従来の多くの実施態様の詳細が簡単にしか言及されておらず、または完全に省かれている。

【００３３】

本発明の１つまたは複数の実施形態は、クラウド・コンピューティングを使用して実施することができる。とは言え、本開示はクラウド・コンピューティングに関する詳細な説明を含むものの、本明細書に記載された教示の実施態様はクラウド・コンピューティング環境だけに限定されないことを予め理解されたい。むしろ、本発明の実施形態は、現在知られているまたは後に開発される他の任意のタイプのコンピューティング環境に関連して実施することができる。

【００３４】

クラウド・コンピューティングは、最小限の管理労力またはサービスのプロバイダとの最小限のインタラクションで迅速に供給およびリリースすることができる構成可能なコンピューティング・リソース（例えばネットワーク、ネットワーク・バンド幅、サーバ、処理、メモリ、ストレージ、アプリケーション、仮想機械、およびサービス）の共用プールへの便利なオンデマンド・ネットワーク・アクセスを可能にするサービス配信モデルである。このクラウド・モデルは、少なくとも５つの特徴、少なくとも３つのサービス・モデル、および少なくとも４つの展開（deployment）モデルを含むことができる。

【００３５】

特徴は以下のとおりである。

オンデマンド・セルフサービス：クラウド・コンシューマは、サーバ時間およびネットワーク・ストレージなどのコンピューティング機能を、このサービスのプロバイダとのヒューマン・インタラクションを必要とすることなく必要に応じて自動的に一方向的に設定することができる。

ブロード・ネットワーク・アクセス：機能は、ネットワーク上で利用可能であり、機能には、異種のシンまたはシック・クライアント・プラットフォーム（例えば移動電話、ラップトップ、およびＰＤＡ）による使用を促進する標準的機構を通してアクセスされる。

リソース・プーリング（resource pooling）：マルチテナント・モデルを使用して多数のコンシューマにサービスを提供するために、プロバイダのコンピューティング・リソースがプールされており、要求に応じて、異なる物理および仮想リソースが動的に割当ておよび再割当てされる。コンシューマは一般に、提供されたリソースの正確な位置を制御できずまたは正確な位置を知らないが、より高次の抽象化レベル（例えば国、州、またはデータセンター）で位置を指定することができることがあるという意味で、位置独立の感覚がある。

ラピッド・エラスティシティ（rapid elasticity）：機能は、素早くスケールアウトするために迅速かつ弾力的に、場合によっては自動的に割り当てることができ、素早くスケールインするために迅速にリリースすることができる。コンシューマにとって、割当てに利用可能な機能はしばしば無限であるように見え、いつでも好きな量だけ購入することができる。

メジャード・サービス（measured service）：クラウド・システムは、サービスのタイプ（例えば、ストレージ、処理、バンド幅、および使用中ユーザ・アカウント）に対して適切なある抽象化レベルで計測機能に介入することによって、リソースの使用状況を自

動的に制御および最適化する。リソースの使用状況を監視、制御、および報告して、利用されているサービスのプロバイダとコンシューマの両方に透明性を提供することができる。

ソフトウェア・アズ・ア・サービス（ＳａａＳ）：コンシューマに提供されるこの機能は、クラウド・インフラストラクチャ上で実行しているプロバイダのアプリケーションを使用する機能である。ウェブ・ブラウザなどのシン・クライアント・インタフェース（例えばウェブ・ベースの電子メール）を通してさまざまなクライアント・デバイスからそれらのアプリケーションにアクセス可能である。場合によっては限られたユーザ固有のアプリケーション構成の設定を除けば、コンシューマは、ネットワーク、サーバ、オペレーティング・システム、ストレージ、または個々のアプリケーション機能さえを含む基礎をなすクラウド・インフラストラクチャを管理も制御もしない。

プラットフォーム・アズ・ア・サービス（ＰａａＳ）：コンシューマに提供されるこの機能は、クラウド・インフラストラクチャ上で、プロバイダがサポートするプログラミング言語およびツールを使用して作成されたコンシューマ作成または取得のアプリケーションを展開する機能である。コンシューマは、ネットワーク、サーバ、オペレーティング・システム、またはストレージを含む基礎をなすクラウド・インフラストラクチャを管理も制御もしないが、展開されたアプリケーションおよび場合によってはアプリケーション・ホスティング環境構成は制御することができる。

インフラストラクチャ・アズ・ア・サービス（ＩａａＳ）：コンシューマに提供されるこの機能は、処理、ストレージ、ネットワーク、および他の基本的なコンピューティング・リソースを割り当てる機能であり、ここで、コンシューマは任意のソフトウェアを展開および実行することができ、これらのソフトウェアは、オペレーティング・システムおよびアプリケーションを含むことができる。コンシューマは、基礎をなすクラウド・インフラストラクチャを管理も制御もしないが、オペレーティング・システム、ストレージ、および展開されたアプリケーションは制御することができ、場合によっては、選択されたネットワーク構成要素（例えばホスト・ファイアウォール）を限定的に制御することができる。

【００３６】

展開モデルは以下のとおりである。

プライベート・クラウド：このクラウド・インフラストラクチャは、組織体のためだけに運営される。インフラストラクチャは、その組織体または第三者が管理することができ、オンプレミス（on-premises）またはオフプレミス（off-premises）で存在することができる。

コミュニティ・クラウド：このクラウド・インフラストラクチャは、いくつかの組織体によって共有され、利害（例えばミッション、セキュリティ要件、ポリシー、およびコンプライアンス上の問題）を共有する特定のコミュニティをサポートする。インフラストラクチャは、その組織体または第三者が管理することができ、オンプレミスまたはオフプレミスで存在することができる。

パブリック・クラウド：このクラウド・インフラストラクチャは、一般大衆または大きな産業グループが利用可能であり、クラウド・サービスを販売している組織体によって所有される。

ハイブリッド・クラウド：このクラウド・インフラストラクチャは、固有の実体を維持しているが、データおよびアプリケーション・ポータビリティを可能にする標準化された技術または独占技術（例えばクラウド間のロード・バランシングのためのクラウド・バースティング（cloud bursting））によって１つに結合された２つ以上のクラウド（プライベート、コミュニティ、またはパブリック）の合成体である。

【００３７】

クラウド・コンピューティング環境は、無国籍、低結合、モジュール性、および意味論的相互運用性（semantic interoperability）に重きを置くサービス指向の環境である。クラウド・コンピューティングの中心には、相互接続されたノードのネットワークを含むインフラストラクチャがある。

【００３８】
次に図１を参照すると、例示的なクラウド・コンピューティング環境５０が示されている。示されているとおり、クラウド・コンピューティング環境５０は１つまたは複数のクラウド・コンピューティング・ノード１０を含み、クラウド・コンシューマによって使用されるローカル・コンピューティング・デバイス、例えばパーソナル・デジタル・アシスタント（ＰＤＡ）もしくは移動電話５４Ａ、デスクトップ・コンピュータ５４Ｂ、ラップトップ・コンピュータ５４Ｃ、または他の何らかのコンピュータ・システムもしくはデバイスあるいはこれらの組合せは、これらのノードと通信することができる。ノード１０は互いに通信することができる。それらのノードは、本明細書において上で説明したように、プライベート、コミュニティ、パブリック、もしくはハイブリッド・クラウド、またはこれらの組合せなどの１つまたは複数のネットワークに、物理的にまたは仮想的にグループ分けされていることがある（図示せず）。これによって、クラウド・コンピューティング環境５０は、インフラストラクチャ、プラットフォーム、もしくはソフトウェア、またはこれらの組合せをサービスとして提供することができ、そのため、クラウド・コンシューマは、ローカル・コンピューティング・デバイス上にリソースを維持する必要がない。図１に示されたタイプのコンピューティング・デバイス５４Ａ～５４Ｃは単なる例であることが意図されていること、ならびにコンピューティング・ノード１０およびクラウド・コンピューティング環境５０は、任意のタイプのネットワーク上もしくはアドレス指定可能なネットワーク接続上またはその両方で（例えばウェブ・ブラウザを使用して）、コンピュータ化された任意のタイプのデバイスと通信することができることが理解される。
【００３９】
次に図２を参照すると、クラウド・コンピューティング環境５０によって提供される一組の機能抽象化層が示されている。図２に示されている構成要素、層、および機能は単なる例であることが意図されており、本発明の実施形態はそれらに限定されないことを予め理解しておくべきである。図示のとおり、以下の層および対応する機能が提供される。ハードウェアおよびソフトウェア層６０は、ハードウェア構成要素およびソフトウェア構成要素を含む。ハードウェア構成要素の例は、メインフレーム６１、ＲＩＳＣ（縮小命令セット・コンピュータ）アーキテクチャ・ベースのサーバ６２、サーバ６３、ブレード・サーバ（blade server）６４、ストレージ・デバイス６５、ならびにネットワークおよびネットワーキング構成要素６６を含む。いくつかの実施形態では、ソフトウェア構成要素が、ネットワーク・アプリケーション・サーバ・ソフトウェア６７およびデータベース・ソフトウェア６８を含む。仮想化層７０は抽象化層を提供し、この層から、仮想実体の以下の例を提供することができる：仮想サーバ７１、仮想ストレージ７２、仮想専用ネットワークを含む仮想ネットワーク７３、仮想アプリケーションおよびオペレーティング・システム７４、ならびに仮想クライアント７５。
【００４０】
一例では、管理層８０が以下の機能を提供することができる。リソース供給８１は、クラウド・コンピューティング環境内でタスクを実行する目的に利用されるコンピューティング・リソースおよびその他のリソースの動的調達を提供する。計量および価格決定８２は、クラウド・コンピューティング環境内でリソースが利用されたときの費用追跡、およびこれらのリソースの消費に対する課金または請求を提供する。一例では、これらのリソースがアプリケーション・ソフトウェア・ライセンスを含むことがある。セキュリティは、クラウド・コンシューマおよびタスクの識別確認ならびにデータおよび他のリソースの保護を提供する。ユーザ・ポータル８３は、コンシューマおよびシステム管理者に、クラウド・コンピューティング環境へのアクセスを提供する。サービス水準管理８４は、必要なサービス水準が達成されるようなクラウド・コンピューティング・リソースの割振りおよび管理を提供する。サービス水準合意（Service Level Agreement）（ＳＬＡ）計画および履行８５は、ＳＬＡに従って将来必要になると予想されるクラウド・コンピューティング・リソースの事前調整および調達を提供する。
【００４１】

ワークロード層９０は、クラウド・コンピューティング環境を利用することができる機能の例を提供する。この層から提供することができるワークロードおよび機能の例は、マッピングおよびナビゲーション９１、ソフトウェア開発およびライフサイクル管理９２、仮想教室教育配信９３、データ解析処理９４、トランザクション処理９５、ならびに高水準プログラム命令のコンパイルを実行するための学習モデル処理９６を含む。

【００４２】

図３を参照すると、本明細書の教示を実施するために通信ネットワークを横切ってクラウド・コンピューティング環境５０の１つまたは複数のノード１０と通信する、一般にコンピュータ・システム１００と呼ぶデータ処理システムの一実施形態が示されている。コンピュータ・システム１００は、１つまたは複数の中央処理ユニット（プロセッサ）１２１ａ、１２１ｂ、１２１ｃなど（これらをひとまとめにしてまたは総称してプロセッサ１２１と呼ぶ）を有する。１つまたは複数の実施形態では、プロセッサ１２１がそれぞれ、縮小命令セット・コンピュータ（reduced instruction set computer）（ＲＩＳＣ）マイクロプロセッサを含むことができる。プロセッサ１２１は、システム・バス１３３を介してシステム・メモリ（ＲＡＭ）１３４および他のさまざまな構成要素に結合されている。システム・バス１３３にはリード・オンリ・メモリ（ＲＯＭ）１２２が結合されており、ＲＯＭ１２２は、コンピュータ・システム１００のある種の基本機能を制御する基本入出力システム（basic input/output system）（ＢＩＯＳ）を含むことができる。

【００４３】

図３はさらに、システム・バス１３３に結合された入力／出力（Ｉ／Ｏ）アダプタ１２７およびネットワーク・アダプタ１２６を示している。Ｉ／Ｏアダプタ１２７は、ハード・ディスク１２３もしくはテープ・ストレージ・ドライブ１２５またはその両方あるいは他の任意の同種の構成要素と通信するスモール・コンピュータ・システム・インタフェース（small computer system interface）（ＳＣＳＩ）アダプタとすることができる。本明細書では、Ｉ／Ｏアダプタ１２７、ハード・ディスク１２３、およびテープ・ストレージ・デバイス１２５をひとまとめにしてマス・ストレージ（mass storage）１２４と呼ぶ。

【００４４】

マス・ストレージ１２４に、処理システム１００上で実行するオペレーティング・システム１４０を記憶することができる。しかしながら、オペレーティング・システム１４０を、コンピュータ・システム１００のＲＡＭ１３４に記憶することもできる。本発明の実施形態によるオペレーティング・システムは例えば、ＵＮＩＸ（Ｒ）（ＴＭ）、Ｌｉｎｕｘ（Ｒ）（ＴＭ）、Ｍｉｃｒｏｓｏｆｔ（Ｒ） ＸＰ（ＴＭ）、ＡＩＸ（Ｒ）（ＴＭ）、およびＩＢＭ（Ｒ）のｉ５／ＯＳ（ＴＭ）を含む。

【００４５】

ネットワーク・アダプタ１２６がバス１３３を外部ネットワーク１３６に相互接続して、コンピュータ・システム１００が他の同様のシステムと通信することを可能にする。システム・バス１３３にはディスプレイ・アダプタ１３２によってスクリーン（例えばディスプレイ・モニタ）１３５が接続されており、ディスプレイ・アダプタ１３２は、グラフィクス集中型アプリケーション（graphics intensive application）の性能を向上させるグラフィクス・アダプタ、およびビデオ・コントローラを含むことができる。一実施形態では、アダプタ１２７、１２６、および１３２を、中間バス・ブリッジ（図示せず）を介してシステム・バス１３３に接続された１つまたは複数のＩ／Ｏバスに接続することができる。ハード・ディスク・コントローラ、ネットワーク・アダプタ、およびグラフィクス・アダプタなどの周辺デバイスを接続するための適当なＩ／Ｏバスは通常、ペリフェラル・コンポーネント・インターコネクト（Peripheral Component Interconnect）（ＰＣＩ）などの一般的なプロトコルを含む。ユーザ・インタフェース・アダプタ１２８およびディスプレイ・アダプタ１３２を介してシステム・バス１３３に接続されたように追加の入力／出力デバイスが示されている。ユーザ・インタフェース・アダプタ１２８を介してバス１３３にキーボード１２９、マウス１３０、およびスピーカ１３１が相互接続され

ており、ユーザ・インタフェース・アダプタ１２８は例えば、多数のデバイス・アダプタを単一の集積回路に統合したＳｕｐｅｒ　Ｉ／Ｏチップを含むことができる。

【００４６】

例示的な実施形態では、コンピュータ・システム１００がグラフィクス処理ユニット１４１を含む。グラフィクス処理ユニット１４１は、ディスプレイに出力することが意図されたフレーム・バッファ内での画像の生成を加速するようメモリを操作および変更するように設計された専門電子回路（specialized electronic circuit）である。一般に、グラフィクス処理ユニット１４１は、コンピュータ・グラフィクスおよび画像処理の操作において非常に効率的であり、大きなデータ・ブロックの処理を並行して実行するアルゴリズムに関して、グラフィクス処理ユニット１４１を汎用ＣＰＵよりも効果的にする高度に並列の構造を有する。

【００４７】

したがって、図３の構成のとおり、コンピュータ・システム１００は、プロセッサ１２１の形態の処理機能、ＲＡＭ１３４およびマス・ストレージ１２４を含む記憶機能、キーボード１２９およびマウス１３０などの入力手段、ならびにスピーカ１３１およびディスプレイ１３５を含む出力機能を含む。一実施形態では、図３に示されたさまざまな構成要素の機能を調整するために、ＲＡＭ１３４およびマス・ストレージ１２４の一部分がオペレーティング・システムを共同で記憶している。

【００４８】

本明細書に記載されているとおり、静的コンパイル・プログラムの場合、コンピュータ・プログラムは、結果として生成される実行可能コードが実行される前にコンパイルされる。静的コンパイル言語に関して、技術的課題は、実行時（すなわちコンピュータ・プログラムが実行されるとき）にだけ集めることができる、コンピュータ・プログラムの効率に対して重大な影響を有し得る情報を、コンピュータ・プログラムがコンパイルされるときに、コンパイラ・プログラムが持っていないことである。追加の技術的課題は、コンパイラ・プログラムが、結果として生じる実行可能コードがその上で実行される特定のデータ処理システムを知らないことがあり、したがって、実行可能コードがその上で実行するデータ処理システムのハードウェア特徴に実行可能コードを適合させることができないことである。このような技術的課題に対処するため、および実行可能コードを効率的にするために、実行可能コードを実行するために使用されるデータ処理システムが変更されたときに、コンピュータ・プログラムは、再コンパイラによって再コンパイルされる。

【００４９】

この手法の技術的課題は、コンピュータ・プログラムを再コンパイルするのに使用されるコンパイラも、現在使用されているデータ処理システムのハードウェア変更および他の変更を利用するために、変更されていることがあることである。さらに、ＣＯＢＯＬ言語などの使用されているプログラミング言語は、未定義の振る舞い（undefined behavior）のインスタンスを有し得る。例えば、特にＣＯＢＯＬのような言語に関しては、コンパイラのあるバージョンで使用されるデータ型が、別のバージョンの同じデータ型の仕様に従わない不適格なデータ（ill-formed data）を有することが起こり得る。さらに、問題は、より古いコンパイラ・バージョンによる特定の低水準コード生成選択により、この不適格なデータに起因する観察可能な問題が偶然に隠され得る／隠蔽され得ることである。新しいコンパイラ・バージョンが、仕様が準拠されていることを前提とするコードを生成したとき、この不適格なデータに起因するこれらの問題は表面化する。

【００５０】

全てのビット・パターンが１つの値に対応するように２進値が使用される言語の場合、これが問題にならないこともあることに留意すべきである。例えば、ＣＯＢＯＬで使用される一般的なデータ型は、「パック１０進数」型および「ゾーン１０進数」型を含む。これらの２進化１０進数（Binary Coded Decimal）（ＢＣＤ）型は無効符号化の範囲を有する。無効符号化に対する特定のプログラム・振る舞いは言語によって明示的に定義されず、特定の無効符号化とこの無効データに対して動作するコンパイラによって生成された

正確な機械命令シーケンスとの産物である。この説明では、本発明の実施形態がＣＯＢＯＬ言語に関して説明される。しかしながら、本発明の実施形態は、他のプログラミング言語にも適用可能である。

【００５１】

ＣＯＢＯＬの場合、データ型によるこのような未定義の振る舞いのため、より新しいコンパイラを用いてプログラムを再コンパイルすることは、大部分の言語よりもリスクが大きい。未定義の振る舞いの結果は予測不可能であり、コンパイラごとに、または同じコンパイラ内のコンパイラ最適化レベル間で異なる可能性がある。未定義の振る舞いを含むプログラムを再コンパイルすることは、より古いコンパイラによって生成された（あるいは、同じコンパイラであっても、異なる環境で実行したとき、またはコンパイラ最適化レベルもしくは任意の他のコンパイラ・オプションなどの異なるコンパイラ・オプションが使用されたときに生成された）古い実行可能コードとは異なる実行に帰着する実行可能コードを生成し得る。ＣＯＢＯＬのような言語における可能な不適格なデータの典型的な例として、ゾーン１０進数データ項目の中のそれぞれのバイトの上位４ビットが適正に初期化されないソース・コードを考える。この言語仕様によれば、これらの４つのビットは０ｘＦであるはずであり、符号付き項目に関して、最下位バイトの４つのビットは有効符号コード０ｘＡ－＞０ｘＦであるはずである。別の可能性は、パックまたはゾーン１０進数データ項目の符号コードを不適正にセットすることを含む。別の潜在的な課題は、ＣＯＢＯＬの２進数２の補数データ項目が、指定された数の１０進数とともに宣言されることに起因する。したがって、宣言された数の１０進数によって実際に表現することができる数よりも多くの２進ビットを含めることによって、故意にまたは偶然に、２進項目を「過密にする（over-populate）」ことが可能である。別の例は、偶数パック１０進数データ項目（evenpacked decimal data item）の最上位バイトの上位４ビットをクリアしない（または偶然にセットする）ことである。偶数精密パック１０進数項目（even precision packed decimal item）は、それらの全ての桁を表現するために最上位バイトの半分だけを必要とし、そのため、このバイトの上位４ビットは、言語仕様に従ってゼロにセットされると仮定される。さまざまな他の例が可能である。

【００５２】

このような技術的課題に対処するため、本明細書ではバイナリ・オプティマイザと呼ぶ２進２進最適化ツール（binary to binary optimization tool）が使用される。このツールはバイナリ・トランスレータと呼ばれることもある。バイナリ・オプティマイザは、（古いバージョン／より古い最適化レベルを有する）コンパイルされた実行可能コードの中の古いバイナリ・モジュールを別の形態に変換し、その形態のバイナリ・モジュールを、異なる（例えばより新しい）コンパイラ技術を使用して最適化する。このケースでは、「変換」が、バイナリ・コードをソース・コードに戻すことを指していないことに留意すべきである。そうではなしに、バイナリ・トランスレータまたはバイナリ・オプティマイザは、元のバイナリ・コードをプログラムの中間言語（ＩＬ）表現に変換するため、したがって、新しい最適化およびコンパイル技術を使用して最適化し、次いで新しいバイナリ・モジュールを生成するために、（より古いバージョンもしくはより低い最適化レベルまたはその両方による）元のバイナリ・コードを分解および分析する。バイナリ・オプティマイザは、コンパイルされたバイナリ実行可能コードを入力として使用するため、未定義の振る舞いの場合であっても、最適化されたバージョンが同じ振る舞いを有することが仮定されている。しかしながら、この手法は、より古いバージョンのコンパイラに限定される。より新しいバージョンのコンパイラは、よりアグレッシブな最適化を使用し、バイナリ・コードを分析することを極端に難しくする。

【００５３】

例えば、高水準コンストラクトの専門インライン展開の生成を考える。「高水準コンストラクト」は、ＩＢＭ（Ｒ）ｚ／Ａｒｃｈｉｔｅｃｔｕｒｅ（Ｒ）命令セットなどの高水準言語または機械言語における文（statement）または命令（instruction）であって、その実行可能表現が複数の低水準機械命令の複雑なコード・シーケンスとなるであろう十

分に複雑な文または命令である。高水準コンストラクトの例は、ＣＯＢＯＬ言語のＩＮＳＰＥＣＴ、およびＩＢＭ（Ｒ）ｚ／Ａｒｃｈｉｔｅｃｔｕｒｅ（Ｒ）命令セットのＥＤ、ＥＤＭＫである。いくつかのケースでは、より新しいバージョンのＣＯＢＯＬコンパイラが、実行時ライブラリ呼出しまたはより低速の命令シーケンスを用いるより一般的だが効率に劣るやり方で処理する代わりに、高水準コンストラクトを表現するために、専門コード・シーケンスを生成する。例えば、ＣＯＢＯＬにおいて、「ＩＮＳＰＥＣＴ」文は、ストリングを探索および変形することができる。より初期のバージョンのＣＯＢＯＬでは普通、ＣＯＢＯＬ実行時ルーチンにコールアウトすることによって、非自明な（non-trivial）ＩＮＳＰＥＣＴ文が実行された。より新しいバージョンでは、バイナリ実行可能コード自体の中でインラインのより高速な実施態様が生成される。ＩＮＳＰＥＣＴの全ての可能なバージョンを処理しなければならなかった実行時ルーチンとは違い、実行しなければならない特定のバージョンに対してインライン・コピーを専門化することができる。同様に、専門コード・シーケンスを用いることによって、以前に使用された一般的なハードウェア命令ＥＤおよびＥＤＭＫを用いるよりも高速に、数値編集型から数値データ型への変換を実行することができる。ＥＤおよびＥＤＭＫ命令は入力としてパターンをとり、可能な任意のパターンを処理しなければならない。実行可能コードで使用される特定のパターンに対して専門化されたシーケンスを生成する方が高速である。ＩＢＭ（Ｒ）およびｚ／Ａｒｃｈｉｔｅｃｔｕｒｅは、世界中の多くの法域で登録されたインターナショナル・ビジネス・マシーンズ・コーポレーションの商標である。

【００５４】

より新しい、より効率的なコンパイラによるコンパイル中、プログラムをコンパイラ最適化している間は通常、高水準コンストラクトに対応するこのような演算は高水準で表現されており、低水準専門シーケンスに変換されるのはコード生成中だけである。このことは、コンパイラ最適化が、高水準コンストラクトの意味論に関する情報を利用することを可能にする。

【００５５】

これらの高水準コンストラクトに対して生成されるコード・シーケンスは、高水準コンストラクトの詳細によって大幅に変化し得る。このようなコード・シーケンスの範囲および目的を、コンパイルされたバイナリ・コードだけに基づいて識別することは、極端に難しい技術的課題である。現状技術のバイナリ・オプティマイザの能力は、対応するＩＬ表現を回復することだけに限定される。このようなバイナリ・オプティマイザは、より古いバージョンのコンパイラによってコンパイルされたコードを変換することができ、第１の実行可能コードはＩＬ表現に簡単に変換することができ、このコンパイラは活発な開発下にない。そのような場合、高水準コンストラクトのＩＬ表現は容易に決定でき、変化を受けないため、このバイナリ・オプティマイザは高水準コンストラクトを最適化することができる。しかし、開発下のコンパイラ、したがって新しいコンパイラ最適化技術を使用するコンパイラに関してはそうではない。その結果、（ＩＮＳＰＥＣＴまたは数値編集のような既存の高水準コンストラクトに対するものであっても）より多くのタイプのまたは異なるタイプのこれらのシーケンスが生成され得る。

【００５６】

本発明の１つまたは複数の実施形態は、このような技術的課題を解決し、より新しいバージョンのコンパイラによってコンパイルされたコードを、本発明の１つまたは複数の実施形態によるバイナリ・オプティマイザによって最適化することを容易にする。このことは、そのバイナリ・オプティマイザが、新しい、より効率的なコード・シーケンスを生成することを可能にする。これは、新規のコード・シーケンスを生成するためにコンパイラが変更されるたびにバイナリ・オプティマイザを変更することを必要とすることなく実行され、バイナリ・オプティマイザが既存の振る舞いを維持することを容易にする。

【００５７】

図４は、本発明の１つまたは複数の実施形態による、最適化システムのブロック図を示している。本明細書で使用されるとき、「コンピュータ・プログラムをコンパイルするこ

と」は、ソース言語で書かれたコンピュータ・プログラムから、バイナリ実行可能コード、すなわちプロセッサが実行することができる機械命令を生成することを含む。さらに、本明細書で使用されるとき、「実行可能コードを最適化すること」は、新たなバイナリ実行可能コードを生成することにより、コンピュータ・プログラムを、バイナリ実行可能コードから最適化することを指す。新たなバイナリ実行可能コードを生成することは、最初のバイナリ実行可能コードをＩＬ表現に変換することを含む。本明細書で使用されるとき、ＩＬは、プログラムを表現することができる任意の言語であることができることに留意すべきである。例えば、ＩＬは、内部言語（internal language）であることができ、コンピュータ・プログラムのソース言語であることができ、内部アノテーション（internal annotation）を有するソース言語であることができ、別のバイナリ言語であることができ、または他の任意のタイプの表現であることができる。これらは全て、第１の実行可能コードと第２の実行可能コードとの間の中間にあるものの例であり、第２の実行可能コードは第１の実行可能コードを最適化したものである。

【００５８】

図示のシナリオでは、第１のバイナリ実行可能コード４３０を生成するために、コンピュータ・プログラム４１０がコンパイラ４２０によってコンパイルされる。コンピュータ・プログラム４１０は、１つまたは複数の高水準コンストラクト４１２を含むことができる。１つのコンストラクト４１２だけが示されているが、プログラム４１０は、１つまたは複数のコンストラクト４１２を含むことができることが理解される。

【００５９】

コードを最適化するとき、例えば新しいハードウェアに対してコードを最適化するときには、バイナリ・オプティマイザ４４０が使用される。バイナリ・オプティマイザ４４０は実行可能コード４３０の変換を実行することができ、さらに、第２の実行可能コード４７０を生成するための最適化および最終コード生成を実行することができる。１つまたは複数の例では、最終コード生成（またはそれよりも低いレベルの追加の最適化）のために第２のコンパイラ（図示せず）に渡される、第１の実行可能コード４３０のＩＬ（またはさらにソース）表現を生成するなど、他の可能性に対してバイナリ・オプティマイザ４４０を使用することができることに留意すべきである。例えば、第２のコンパイラはコンパイラ４２０よりも新しいコンパイラである。あるいは、第２のコンパイラは、使用される最適化レベルなど異なる設定を使用する、コンパイラ４２０のインスタンスである。

【００６０】

バイナリ・オプティマイザ４４０は、第１の実行可能コード４３０を入力として受け取り、実行可能コード４３０を変換して、コンピュータ・プログラムのＩＬ表現４５０を出力する。バイナリ・オプティマイザ４４０は次いで、コンピュータ・プログラムのＩＬ表現４５０を最適化して、第２のバイナリ実行可能コード４７０を生成する。すなわち、コンパイラ４２０は、後にさらに説明する追加のコンパイル時情報を含むことができる第１の実行可能コード４３０を生成し、バイナリ・オプティマイザ４４０は、第１の実行可能コード４３０およびコンパイル時情報を消費する。コンパイラ４２０によって格納される任意のコンパイル時情報は、最適化された第２の実行可能コード４７０を生成する最適化のための情報である。

【００６１】

上述のとおり、コンパイラ４２０のさまざまなバージョン、特に、より新しいコンパイラ技術によって、第１のバイナリ実行可能コード４３０を分析することは技術的に難しい。これは、このような分析を移動目標（moving target）とし得る実行時指示文（runtime directive）に起因する。現在の現状技術のバイナリ・オプティマイザは、バイナリ実行可能コード４３０をＩＬ表現４５０に変換することができず、その結果として、コンパイラ４２０が通常生成する高水準コンストラクト４１２の実行可能表現から、高水準コンストラクト４１２を変換することができない。

【００６２】

この技術的課題を解決するため、およびこのようなコンパイラ（例えばより新しいコン

パイラ）を用いてコンパイルされたプログラムに対してバイナリ・オプティマイザ４４０を使用することを可能にするために、本発明の１つまたは複数の実施形態は、上述のとおり、コンパイルされたバイナリ実行可能コード４３０に追加のコンパイル時情報を含めることを容易にする。この追加情報を使用して、プログラム４１０の意味に関する情報を伝達することができる。

【００６３】

本発明の１つまたは複数の実施形態では、バイナリ・オプティマイザ４４０が、コンストラクト４１２を、プログラムのＩＬ表現４５０に含めるコンストラクト４１２のＩＬ表現に変換すること、高水準コンストラクトのＩＬ表現を最適化すること、および高水準コンストラクトを第２の実行可能コード４７０に潜在的に含めるための新しい命令シーケンスを生成することを可能にするために、コンパイラ４２０が、第１の実行可能コード４３０に、プログラム４１０の中のそれぞれの高水準コンストラクト４１２に関するコンパイル時情報を含める。

【００６４】

この最適化により、いくつかの入力上で、新たに最適化された命令が、古い命令と同じ振る舞いを持たないことが起こり得る。これに対処するため、本発明の１つまたは複数の実施形態は、テスト・ベクタ、すなわち自動的に生成された一組の入力を使用して、高水準コンストラクトに対して生成された新しい命令シーケンスと古い命令シーケンスとの振る舞いを比較する。これらの２つのシーケンスが、テスト・ベクタの中の全ての入力に対して同じ出力を与える場合、高水準コンストラクトに対する新しい命令を第２の実行可能コード４７０に含めることは安全であると見なされる。さらに、最適化中に、新しい命令シーケンスの振る舞いが古いシーケンスと比較されるため、高水準コンストラクトに対する最適化されたコード４７０は、バグまで含めた互換性（bug-for-bug compatibility）を維持する。

【００６５】

図５は、本発明の１つまたは複数の実施形態による、テスト・ベクタを使用して高水準コンストラクトの最適化を自動的に検証する方法の流れ図を示している。この方法は、５１０で、コンパイラ４２０を使用してプログラム４１０をコンパイルすることを含む。このコンパイルは、５１２で、プログラム４１０の中の高水準コンストラクト４１２のそれぞれのインライン展開に関するコンパイル時情報をエミットする（emit）ことを含む。この情報は、展開中の高水準コンストラクト４１２の識別子、例えば「ｅｄｉｔ－ａｎｄ－ｍａｒｋ」または「ｉｎｓｐｅｃｔ　ｔａｌｌｙｉｎｇ」を含む。この情報はさらに、第１の実行可能コード４３０内における、インライン展開によって生成された一組の命令の位置を含む。コンパイラ４２０によって実行された、命令スケジューリングなどの最適化のため、これらの命令は、連続した範囲を形成していないことがある。

【００６６】

コンパイル時情報はさらに、高水準コンストラクト４１２への入力ごとに、その入力がどこにあるのかに関する情報を含む。例えば、ｅｄｉｔ－ａｎｄ－ｍａｒｋ演算に入力される値がレジスタ内またはメモリ内にあることがある。コンパイル時情報はさらに、高水準コンストラクトの出力ごとに、その出力が、レジスタ内またはメモリ内など、どこに置かれるのかに関する情報を含む。入力が読み出される／出力が書き込まれるメモリもしくはレジスタまたはその両方はそれぞれ、メモリ・アドレスまたはレジスタ・アドレスを使用して識別される。この情報は、実行可能バイナリ・コード４３０に格納される。

【００６７】

５２０で、コンパイラ４２０を使用して第１の実行可能コード４３０が生成され、このコードは、バイナリ・オプティマイザ４４０を使用してこのコードが最適化されると決められるまで、実行に使用することができる。このコードが最適化されると決められるのは、例えば、より新しいバイナリ・オプティマイザまたはより新しいハードウェアが使用可能になったときである。しかしながら、他のいくつかの理由でこの最適化を実行することもできる。５３０で、バイナリ・オプティマイザ４４０が、第１の実行可能コード４３０

を、プログラムのＩＬ表現４５０に変換する。この変換中に、５３２で、バイナリ・オプティマイザ４４０は、実行可能バイナリ・コード４３０に格納された、エミットされたコンパイル時情報を使用して、高水準コンストラクト４１２を、そのＩＬ表現として生成する。

【００６８】

ＩＬ表現４５０の中に、高水準コンストラクト４１２に対して生成された機械命令は明示的に表現されていない。バイナリ・オプティマイザ４４０の省略時動作は通常、元の全ての入力機械命令を、それを最適化するために、ＩＬ表現４５０の中の対応する構造体として表現することである。しかしながら、本明細書で論じられている（ｉｎｓｐｅｃｔおよび数値編集のような）より高水準のコンストラクトに関しては、高水準コンストラクトのＩＬ表現４５０を生成するこのやり方は避けるべきである。このように表現することは、バイナリ・オプティマイザ４４０が、これらの入力命令の１つの範囲にわたって実行されている全体的なより高水準の演算を決定することを非常に難しくし、または不可能にし得る。したがって、入力命令の１つの範囲にわたるこのようなより高水準の演算に関する情報がバイナリ・コード４３０の中に存在するとき、バイナリ・オプティマイザ４４０は、実行されているより高水準の演算をカプセル化するために、同様に高水準のＩＬ表現を選択する。

【００６９】

さらに、５４０で、プログラム４１０の最適化のため、バイナリ・オプティマイザ４４０は、高水準コンストラクト４１２のＩＬ表現４５０から、高水準コンストラクト４１２に対する最適化された新しい命令を生成するために使用される。これらの新しい命令は、コンパイラ４２０によって生成された命令と比較される。

【００７０】

この時点で、第１のバイナリ実行可能コード４３０の中のコンパイル時情報およびステップ５４０の完了によって得られた情報に基づいて、高水準コンストラクト４１２ごとに以下の情報が使用可能である。コンパイル時情報は、高水準コンストラクト４１２の識別、コンパイラ４２０によって生成された、高水準コンストラクト４１２に対する古い命令の位置、ならびにコンパイラ４２０によるそれらの古い命令に対する入力および出力の位置を含み、ステップ５４０の完了によって得られた情報は、バイナリ・オプティマイザ４４０によって生成された、高水準コンストラクト４１２に対する新しい命令、ならびにバイナリ・オプティマイザ４４０によるそれらの新しい命令に対する入力および出力の位置を含む。

【００７１】

この方法はさらに、５５０で、コンパイル時情報を使用して、高水準コンストラクト４１２ごとに、２つのラッパ関数、すなわちＦ＿ｎｅｗおよびＦ＿ｏｌｄを生成することを含む。「Ｆ＿ｎｅｗ」および「Ｆ＿ｏｌｄ」は例示的な名称であること、ならびに他の例では他の名称／識別を使用することができることが理解される。これらのラッパ関数を生成することは、５５２で、入力および出力パラメータ、ならびにラッパ関数の一部である命令を構成することを含む。これらの関数への入力は高水準コンストラクト４１２への入力である。これらの関数の戻り値は高水準コンストラクト４１２の出力である。内部的には、それぞれの関数は、Ｆ＿ｏｌｄに関しては古い命令の、Ｆ＿ｎｅｗに関しては新しい命令のコピーとして実装され、古い命令は、コンパイラ４２０による命令であり、新しい命令は、バイナリ・オプティマイザ４４０による命令である。

【００７２】

さらに、５６０で、高水準コンストラクト４１２の型ごとに、テスト・ベクタを生成するための手順が定義される。テスト・ベクタは、コンパイラ４２０およびバイナリ・オプティマイザ４４０によって生成されたコード・シーケンスの振る舞いを比較するテストのリストである。テスト・ベクタは、高水準コンストラクトへの有効入力（valid input）と無効入力（invalid input）の両方を含む。任意選択で、バイナリ・オプティマイザ４４０が、（以前の演算を調べることによって）その無効入力がこの点に到達することがで

きないと規定した場合には、ある種の無効入力を省略するように、テスト・ベクタに制約を課すことができる。テスト・ベクタは、高水準コンストラクト５１２の型に基づく所定の一組のテストを含むことができ、ここで、高水準コンストラクト４１２の型は、コンパイル時情報からの識別に基づいて決定される。

【００７３】

５７０で、テスト・ベクタの中のテストごとに、Ｆ＿ｎｅｗおよびＦ＿ｏｌｄラッパ関数が実行され、対応する出力が収集される。５８０で、高水準コンストラクト４１２に対してどの命令を第２の実行可能コード４７０に含めるのかを決定するために、２つのラッパ関数からの出力が比較される。本発明の１つまたは複数の実施形態によれば、出力を収集するために、Ｆ＿ｎｅｗおよびＦ＿ｏｌｄの実行が例外ハンドラでラップされる。ラッパ関数への無効入力はハードウェア例外を引き起こし得る。古い命令と新しい命令が同じハードウェア例外を引き起こす限り、このことは許容され得る。テスト・ベクタは、古い命令と新しい命令との振る舞いが実質的に同一であることを保証するために使用される。

【００７４】

高水準コンストラクト４１２に対するテストの出力の比較に基づいて、その高水準コンストラクト４１２に対してどの命令を第２の実行可能コード４７０に含めるのかが決定される。さらに、この方法では、これらの出力が一致しない場合、たとえテスト・ベクタからの単一の出力が一致しない場合であっても、５８０で、第２のバイナリ実行可能コード４７０に含める、高水準コンストラクト４１２に対する命令が決定される。新しい命令と古い命令とが同じ出力を提供しない場合にはいくつかの選択肢がある。これらの選択肢のうちのどの１つの選択肢をこれらの選択肢から選択するのかは、後にさらに説明するようにいくつかの因子に基づく。この選択に基づくそれぞれの高水準コンストラクトに対する命令の振る舞いが、第１の実行可能プログラム４３０の中の高水準コンストラクトに対する対応する命令と一致した後、５９０で、バイナリ・オプティマイザ４４０は、選択されたそれぞれの高水準コンストラクトに対する対応する命令を含む第２の実行可能プログラム４７０を生成およびエミットする。

【００７５】

図６は、本発明の１つまたは複数の実施形態による、高水準コンストラクト４１２に対するどの命令を第２の実行可能プログラム４７０に含めるのかを決定するための方法の流れ図を示している。６１０で、テスト・ベクタの中の全ての入力に関してＦ＿ｎｅｗとＦ＿ｏｌｄとが同じ出力を与える場合、６２０で、第２のバイナリ実行可能プログラム４７０に新しい命令が含められる。

【００７６】

本発明の１つまたは複数の実施形態では、高水準コンストラクト４１２に対する新しい命令が古い命令の振る舞いと一致しない場合、６３０で、ユーザは、第２の実行可能プログラム４７０にどの命令を含めるのかを選択するよう促され得る。本発明の１つまたは複数の実施形態によれば、このようなケースが生じた場合に第２の実行可能プログラム４７０にどの命令を含めるのかを選択するための選択肢が、オプティマイザ４４０に対して、オプティマイザ４４０が実行される前に設定される。その代わりに、またはそれに加えて、本発明の１つまたは複数の実施形態では、それに対してそのテストが結果として生じる高水準コンストラクト４１２に対する新しい命令が一致しない場合、６３０で、コンパイラ４２０からの元の命令が第２の実行可能コードにエミットされる。これを、６３０で、実行される選択の省略時動作とすることができる。１つまたは複数の例では、振る舞いが一致することを保証するために、テスト・ベクタからのテストが再び実行される。

【００７７】

その代わりに、またはそれに加えて、この選択が、新しい命令による性能向上（performancegain）に基づいて容易にされる。例えば、高水準コンストラクト４１２の新しいコード・シーケンスによる予想される性能向上が、古いコード・シーケンスの性能向上に比べて十分に大きい場合、ユーザは、よりアグレッシブな選択肢を選択することができる。ここで、十分に大きな性能向上は、実行時間、コンピュータ・リソース使用量、または任

意の他の指標（metric）に基づいて決定することができる。古いコードと新しいコードとのこのような指標の比較が、少なくとも所定のしきい値向上に帰結する場合、新しいコードは十分に大きな性能向上を有すると見なすことができる。例えば、実行時間に関して、実行時間が、少なくとも１０％、２５％、５０％だけ、または任意の他の同様の比率（もしくは他のタイプのしきい値）だけ向上する場合、新しいコードは十分に大きな性能向上を有すると見なすことができ、ユーザは、（たとえそれが異なる振る舞いを有するとしても）新しいコードを選択することができる。

【００７８】

その代わりに、またはそれに加えて、新しいコード・シーケンスが、（あるタイプの）無効データに関して異なる結果を与えるだけであり、（それらのタイプの）無効データを検出するための実行時テストのコストが、新しいコード・シーケンスによる性能向上よりも小さい場合、バイナリ・オプティマイザ４４０は、古いコード・シーケンスと新しいコード・シーケンスの両方を含む新たな（第３の）バイナリ実行可能コードであって、この実行時テストによって監視された(guarded)新たな（第３の）バイナリ実行可能コードを生成することができる。このテストに合格した場合には新しいコード・シーケンスが使用される。このテストに合格しなかった場合には古いコード・シーケンスが使用される。これは、全体的な性能向上を提供し、その一方で同一の結果を依然として保証する。

【００７９】

さらに、本発明の１つまたは複数の実施形態では、テスト・ベクタを使用する代わりに、ブール充足可能性（ＳＡＴ）ソルバまたは他の同様のツールを使用して、高水準コンストラクト４１２に対する新しいコードとその高水準コンストラクト４１２に対する古いコードとの一致が判定される。この一致テストは、バイナリ・オプティマイザ４４０による最適化の一部として実行されるため、およびＳＡＴソルバはかなりの時間を要し得るため、バイナリ・オプティマイザ４４０を使用したこの最適化は、上述のようにテスト・ベクタの使用に比べて要する時間が長すぎると考えられ得る。

【００８０】

その上さらに、１つまたは複数の例では、何らかの理由で第１の実行可能コード４３０の中にコンパイル時情報が存在しない場合に、（第１の実行可能コード４３０をソース・コードに逆に変換する真のデコンパイラ（図示せず）などの）別のツールまたはバイナリ・オプティマイザ４４０が、より高水準の演算を分析および発見し、次いで、本明細書に記載されたテスト・ベクタ手法を続けることができる。

【００８１】

それに応じて、本発明の１つまたは複数の実施形態は、ＣＯＢＯＬなどの言語に対する静的コンパイル・コードの最適化を向上させる実用的なアプリケーションを提供する。それに応じて、本発明の１つまたは複数の実施形態は、コンピューティング技術、特にコンパイラおよびオプティマイザの改良を提供する。本発明の１つまたは複数の実施形態は、テスト・ベクタを使用して高水準コンストラクトの最適化の検証を容易にする。

【００８２】

例えば、この検証は、スマート・バイナリ対応コンパイラ（smart-binary-enabled compiler）を使用してプログラムをコンパイルすることを含む。「スマート・バイナリ対応」コンパイラは、展開中の高水準コンストラクトを含むそれぞれのインライン展開に関する情報を、生成中のバイナリ・コードにエミットおよび格納する。格納された情報は、インライン展開によって生成された一組の命令の識別を含む。この情報はさらに、高水準コンストラクトへの入力ごとに、入力する１つまたは複数の入力パラメータがどこに位置するのかに関する情報を含む。例えば、入力パラメータの位置を、メモリ内の位置アドレス、ファイル内の位置、または任意の他の同様の指示として提供することができる。この情報はさらに、高水準コンストラクトの出力ごとに、出力がどこに置かれるのかに関する情報を含む。例えば、出力の位置を、１つまたは複数の出力パラメータが格納される、メモリ内の位置アドレス、ファイル内の位置、または任意の他の同様の指示として提供することができる。

【００８３】
生成されたコード（古い命令）を含むプログラムを（バイナリ・オプティマイザなどの）オプティマイザによって最適化することは、生成された（バイナリ）コードをＩＬ表現に変換することを含む。生成されたコードは、プログラムのソース・コードの中で使用される高水準コンストラクトに対する１つまたは複数の命令を含み得る。さらに、プログラムが最適化されたことに応答して、別のバイナリ実行可能コード（新しい命令）が生成され、このコードは、高水準コンストラクトに対する改訂された一組の命令を含むことができる。１つまたは複数の例では、コンパイル時情報が、高水準コンストラクトの識別、古い命令の位置、古い命令に対する入力および出力の位置を含む。
【００８４】
この検証はさらに、コンピュータによって、高水準コンストラクトごとに、ラッパ関数Ｆ＿ｎｅｗおよびＦ＿ｏｌｄを生成することを含み、これらの関数への入力は高水準コンストラクトへの入力であり、これらの関数の戻り値は高水準コンストラクトの出力である。本発明の１つまたは複数の実施形態では、古い命令と新しい命令との振る舞いを比較するように設計されたテストのリストを含むテスト・ベクタを生成するための手順が生成され、このテスト・ベクタは有効入力および無効入力を含む。Ｆ＿ｎｅｗ関数およびＦ＿ｏｌｄ関数はそれぞれ、テスト・ベクタの中のそれぞれのテストに対して、それぞれの出力を収集するために実行され、テストすることは、プログラムの最適化の一部として実行される。Ｆ＿ｎｅｗ関数とＦ＿ｏｌｄ関数とが、テスト・ベクタの中の全ての入力に対して同じ出力を与えるとコンピュータが判定してことに応答して、最適化されたプログラムの中の新しい命令の使用は安全である（すなわち古い振る舞いと一致する）との指示が提供される。Ｆ＿ｎｅｗ関数とＦ＿ｏｌｄ関数とが、テスト・ベクタの中の全ての入力に対して同じ出力を与えるわけではないとコンピュータが判定したことに応答して、コンピュータは、一組の所定の選択肢からの選択を容易にすることができる。
【００８５】
本発明は、システム、方法、もしくはコンピュータ・プログラム製品、またはこれらの組合せであることがある。コンピュータ・プログラム製品は、本発明の態様をプロセッサに実行させるためのコンピュータ可読プログラム命令をその上に有するコンピュータ可読ストレージ媒体を含むことがある。
【００８６】
このコンピュータ可読ストレージ媒体は、命令実行デバイスが使用するための命令を保持および記憶することができる有形のデバイス（tangible device）とすることができる。このコンピュータ可読ストレージ媒体は例えば、限定はされないが、電子ストレージ・デバイス、磁気ストレージ・デバイス、光学ストレージ・デバイス、電磁気ストレージ・デバイス、半導体ストレージ・デバイス、またはこれらの任意の適当な組合せとすることができる。コンピュータ可読ストレージ媒体のより具体的な例の非網羅的なリストは、ポータブル・コンピュータ・ディスケット、ハード・ディスク、ランダム・アクセス・メモリ（ＲＡＭ）、リード・オンリ・メモリ（ＲＯＭ）、消去可能なプログラマブル・リード・オンリ・メモリ（ＥＰＲＯＭまたはフラッシュ・メモリ）、スタティック・ランダム・アクセス・メモリ（ＳＲＡＭ）、ポータブル・コンパクト・ディスク・リード・オンリ・メモリ（ＣＤ－ＲＯＭ）、デジタル・バーサタイル・ディスク（ＤＶＤ）、メモリ・スティック、フロッピー（Ｒ）・ディスク、機械的にコード化されたデバイス、例えばパンチカードまたはその上に命令が記録された溝の中の一段高くなった構造体、およびこれらの任意の適当な組合せを含む。本明細書で使用されるとき、コンピュータ可読ストレージ媒体は、それ自体が一過性の信号、例えば電波もしくは他の自由に伝搬する電磁波、ウェーブガイドもしくは他の伝送体内を伝搬する電磁波（例えば光ファイバ・ケーブル内を通る光パルス）、または電線を通して伝送される電気信号であると解釈されるべきではない。
【００８７】
本明細書に記載されたコンピュータ可読プログラム命令は、コンピュータ可読ストレージ媒体からそれぞれのコンピューティング／処理デバイスにダウンロードすることができ

、またはネットワークを介して、例えばインターネット、ローカル・エリア・ネットワーク、ワイド・エリア・ネットワーク、もしくは無線ネットワーク、またはこれらの組合せを介して外部コンピュータもしくは外部ストレージ・デバイスにダウンロードすることができる。このネットワークは、銅伝送ケーブル、光伝送ファイバ、無線伝送、ルータ、ファイアウォール、スイッチ、ゲートウェイ・コンピュータ、もしくはエッジ・サーバ、またはこれらの組合せを含むことができる。それぞれのコンピューティング／処理デバイス内のネットワーク・アダプタ・カードまたはネットワーク・インタフェースは、コンピュータ可読プログラム命令をネットワークから受信し、それらのコンピュータ可読プログラム命令を、それぞれのコンピューティング／処理デバイス内のコンピュータ可読ストレージ媒体に記憶するために転送する。

【００８８】

本発明の動作を実行するためのコンピュータ可読プログラム命令は、アセンブラ命令、命令セット・アーキテクチャ（ＩＳＡ）命令、機械命令、機械依存命令、マイクロコード、ファームウェア命令、もしくは状態設定データであってもよく、またはＳｍａｌｌｔａｌｋ（Ｒ）、またはＣ＋＋などのオブジェクト指向プログラミング言語、および「Ｃ」プログラミング言語または同種のプログラミング言語などの従来の手続き型プログラミング言語を含む、１つまたは複数のプログラミング言語の任意の組合せで書かれた、ソース・コードもしくはオブジェクト・コードのいずれかであってもよい。このコンピュータ可読プログラム命令は、全体がユーザのコンピュータ上で実行されてもよく、一部がユーザのコンピュータ上で実行されてもよく、独立型ソフトウェア・パッケージとして実行されてもよく、一部がユーザのコンピュータ上で、一部がリモート・コンピュータ上で実行されてもよく、または全体がリモート・コンピュータもしくはリモート・サーバ上で実行されてもよい。上記の最後のシナリオでは、リモート・コンピュータが、ローカル・エリア・ネットワーク（ＬＡＮ）もしくはワイド・エリア・ネットワーク（ＷＡＮ）を含む任意のタイプのネットワークを介してユーザのコンピュータに接続されてもよく、またはこの接続が、外部コンピュータに対して（例えばインターネット・サービス・プロバイダを使用してインターネットを介して）実施されてもよい。いくつかの実施形態では、本発明の態様を実施するために、例えばプログラム可能論理回路、フィールド・プログラマブル・ゲート・アレイ（ＦＰＧＡ）、またはプログラム可能論理アレイ（ＰＬＡ）を含む電子回路が、このコンピュータ可読プログラム命令の状態情報を利用してその電子回路をパーソナライズすることにより、このコンピュータ可読プログラム命令を実行してもよい。

【００８９】

本明細書では、本発明の態様が、本発明の実施形態による方法、装置（システム）、およびコンピュータ・プログラム製品の流れ図もしくはブロック図またはその両方の図を参照して説明される。それらの流れ図もしくはブロック図またはその両方の図のそれぞれのブロック、およびそれらの流れ図もしくはブロック図またはその両方の図のブロックの組合せは、コンピュータ可読プログラム命令によって実施することができることが理解される。

【００９０】

これらのコンピュータ可読プログラム命令は、コンピュータまたは他のプログラム可能データ処理装置のプロセッサによって実行される命令が、流れ図もしくはブロック図またはその両方の図のブロックに指定された機能／動作を実施する手段を生成するような態様で、汎用コンピュータ、専用コンピュータ、または他のプログラム可能データ処理装置のプロセッサに提供されてマシンを作り出すものであってよい。これらのコンピュータ可読プログラム命令はさらに、命令が記憶されたコンピュータ可読ストレージ媒体が、流れ図もしくはブロック図またはその両方の図のブロックに指定された機能／動作の態様を実施する命令を含む製品を含むような態様で、コンピュータ可読ストレージ媒体に記憶され、コンピュータ、プログラム可能データ処理装置、もしくは他のデバイス、またはこれらの組合せに特定の方式で機能するように指示することができるものであってもよい。

【００９１】

これらのコンピュータ可読プログラム命令はさらに、コンピュータ、他のプログラム可能装置、または他のデバイス上で実施される命令が、流れ図もしくはブロック図またはその両方の図のブロックに指定された機能／動作を実施するような態様で、コンピュータによって実施されるプロセスを生み出すために、コンピュータ、他のプログラム可能データ処理装置、または他のデバイス上にロードされ、コンピュータ、他のプログラム可能装置、または他のデバイス上で一連の動作ステップを実行させるものであってもよい。

【００９２】

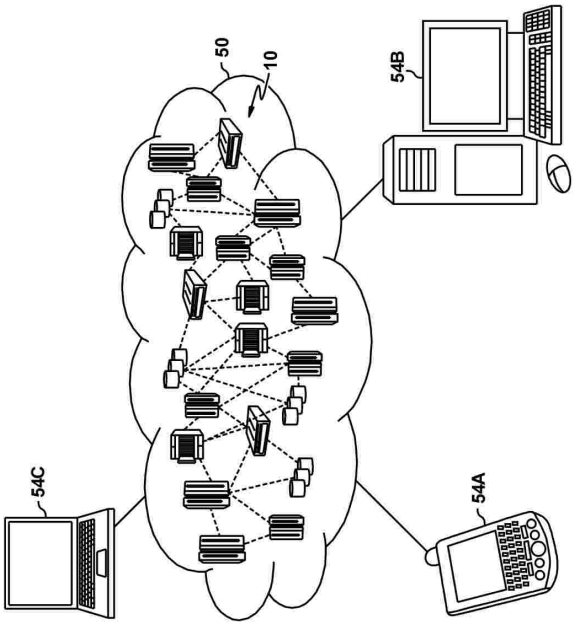
添付図中の流れ図およびブロック図は、本発明のさまざまな実施形態によるシステム、方法、およびコンピュータ・プログラム製品の可能な実施態様のアーキテクチャ、機能、および動作を示す。この点に関して、それらの流れ図またはブロック図のそれぞれのブロックは、指定された論理機能を実施するための１つまたは複数の実行可能命令を含む、命令のモジュール、セグメント、または部分を表すことがある。いくつかの代替実施態様では、ブロックに示された機能を、図に示された順序とは異なる順序で実行することができる。例えば、連続して示された２つのブロックが、実際は、実質的に同時に実行されることがあり、または、含まれる機能によってはそれらのブロックが逆の順序で実行されることもある。それらのブロック図もしくは流れ図またはその両方の図のそれぞれのブロック、ならびにそれらのブロック図もしくは流れ図またはその両方の図のブロックの組合せを、指定された機能もしくは動作を実行しまたは専用ハードウェアとコンピュータ命令との組合せを実施するハードウェアベースの専用システムによって実施することができることにも留意すべきである。

【００９３】

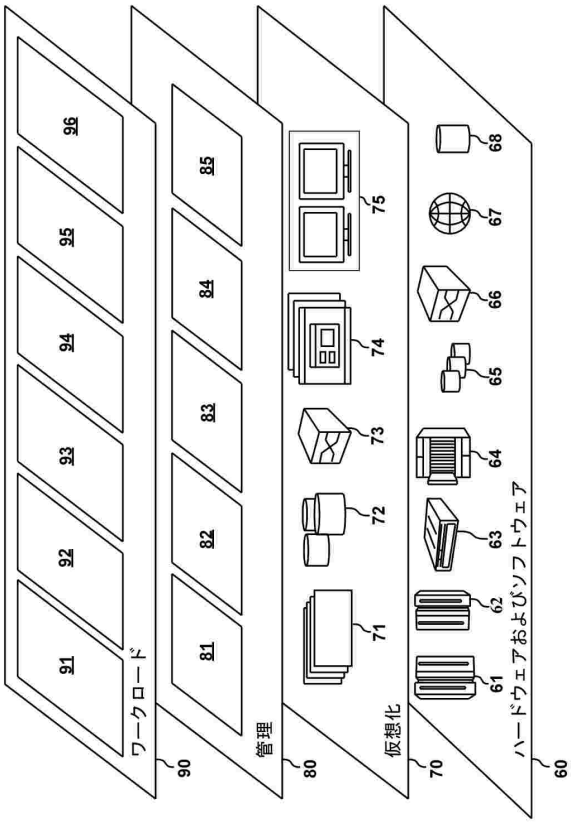
本発明のさまざまな実施形態の説明は例示のために示したものであり、それらの説明が網羅的であること、または開示された実施形態に限定されることは意図されていない。当業者には、記載された実施形態の範囲および思想を逸脱しない多くの変更および変形が明らかとなろう。本明細書で使用されている用語は、実施形態の原理、実際的用途、もしくは市場に出ている技術には見られない技術的改良を最もうまく説明するように、または本明細書に開示された実施形態を他の当業者が理解することができるように選択した。

【図面】

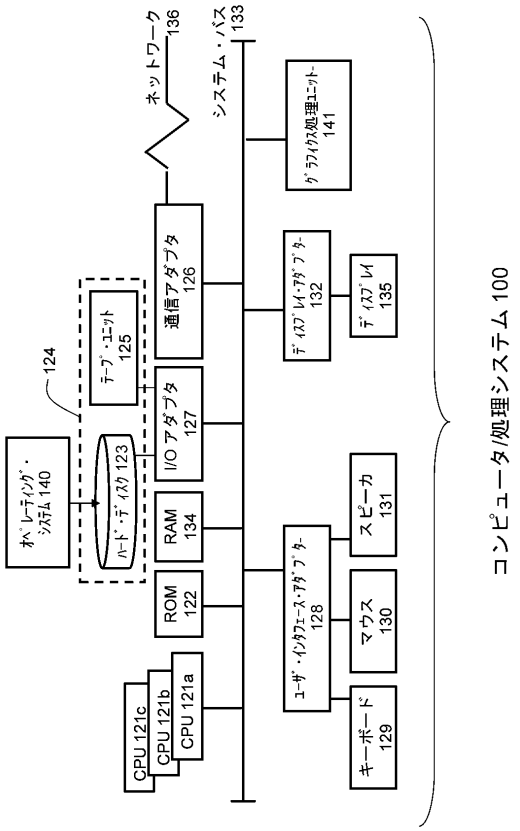
【図１】


50
10
54B
54C
54A

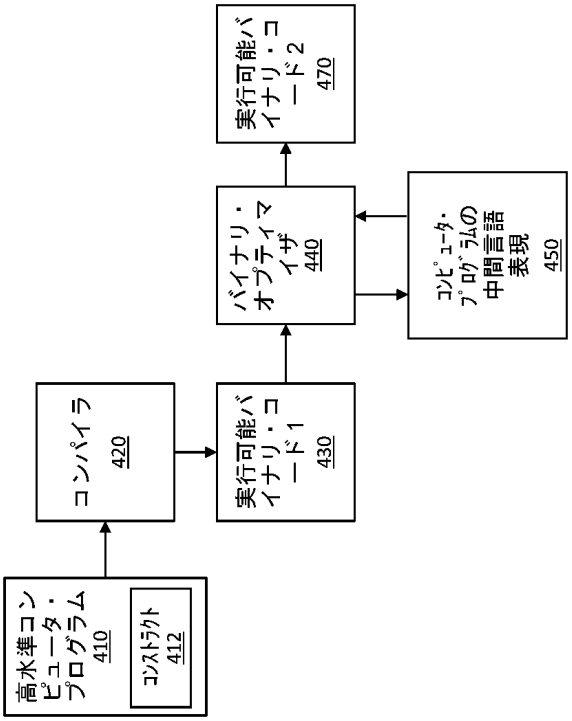
【図２】


96
95
94
93
92
91
ワークロード
90
85
84
83
82
81
管理
80
75
74
73
72
71
仮想化
70
68
67
66
65
64
63
62
61
ハードウェアおよびソフトウェア
60

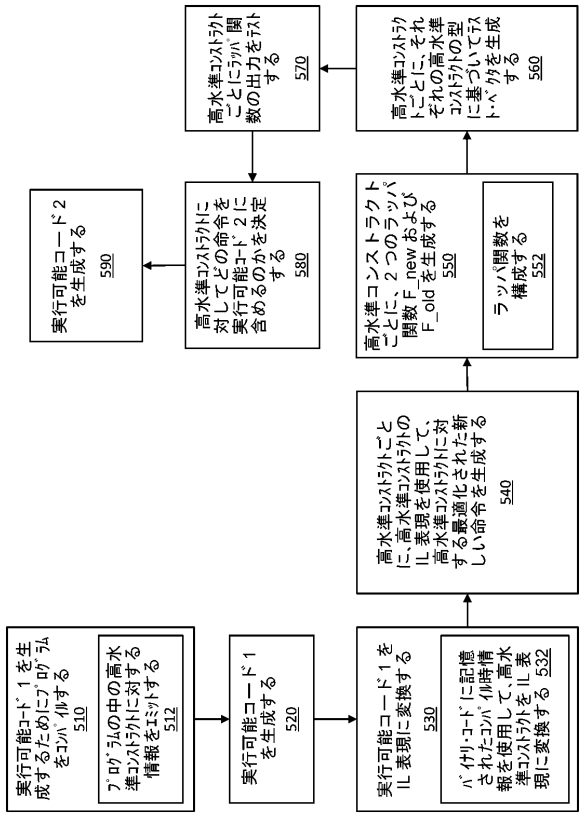
【図３】


ネットワーク
136
システム・バス
133
124
オペレーティング・システム 140
ハード・ディスク 123
テープ・ユニット 125
通信アダプタ 126
I/O アダプタ 127
RAM 134
ROM 122
CPU 121c
CPU 121b
CPU 121a
ユーザ・インタフェース・アダプタ 128
キーボード 129
マウス 130
スピーカ 131
ディスプレイ・アダプタ 132
ディスプレイ 135
グラフィクス処理ユニット 141
コンピュータ/処理システム 100

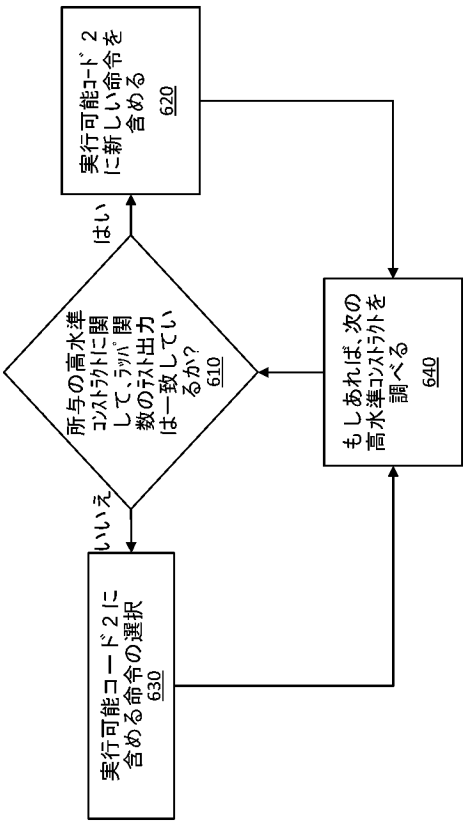
【図４】


高水準コンピュータ・プログラム 410
コンストラクト 412
コンパイラ 420
実行可能バイナリ・コード1 430
バイナリ・オプティマイザ 440
コンピュータ・プログラムの中間言語表現 450
実行可能バイナリ・コード2 470

【図５】


実行可能コード1を生成するためにプログラムをコンパイルする 510
プログラムの中の高水準コンストラクトに対する情報をエミットする 512
実行可能コード1を生成する 520
実行可能コード1をIL表現に変換する 530
バイナリ・コードに記憶されたコンパイル時情報を使用して、高水準コンストラクトをIL表現に変換する 532
高水準コンストラクトごとに、高水準コンストラクトのIL表現を使用して、高水準コンストラクトに対する最適化された新しい命令を生成する 540
高水準コンストラクトごとに、2つのラッパ関数F_newおよびF_oldを生成する 550
ラッパ関数を構成する 552
高水準コンストラクトごとに、それぞれの高水準コンストラクトの型に基づいてテスト・ベクタを生成する 560
高水準コンストラクトごとにラッパ関数の出力をテストする 570
高水準コンストラクトに対してどの命令を実行可能コード2に含めるのかを決定する 580
実行可能コード2を生成する 590

【図６】


実行可能コード2に新しい命令を含める 620
所与の高水準コンストラクトに関してラッパ関数のテスト出力は一致しているか? 610
はい
いいえ
もしあれば、次の高水準コンストラクトを調べる 640
実行可能コード2に含める命令の選択 630

フロントページの続き

(72)発明者　アイルランド、イアン、アレクサンダー
カナダ エル３アール ９ゼット７ オンタリオ州マークハム スティールス・アベニュー・イースト３６００
(72)発明者　コープランド、レイド
カナダ エル６ジー １シー７ オンタリオ州マークハム トロント・ラボ ワーデン・アベニュー８２００
(72)発明者　キールストラ、アラン
カナダ エル３アール ９ゼット７ オンタリオ州マークハム スティールス・アベニュー・イースト３６００
(72)発明者　シーグワルト、デイヴィッド、ケヴィン
イギリス エスオー２１ ２ジェイエヌ ハンプシャー州ウィンチェスター ハースリーパーク
(72)発明者　孝壽 俊彦
東京都中央区日本橋箱崎町１９番２１号 日本アイ・ビー・エム株式会社内
審査官　児玉 崇晶

(56)参考文献　特開２０１５－１５３１９１（ＪＰ，Ａ）
特開２００４－２９５３９８（ＪＰ，Ａ）
米国特許出願公開第２０１８／０２６０１９８（ＵＳ，Ａ１）
米国特許出願公開第２００９／０１６４５０１（ＵＳ，Ａ１）

(58)調査した分野　(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　８／４１