

(19) 日本国特許庁 (JP)

(12) 特 許 公 報 (B2)

(11) 特許番号

特許第5719930号

(P5719930)

(45) 発行日 平成27年5月20日 (2015. 5. 20)

(24) 登録日 平成27年3月27日 (2015. 3. 27)

(51) Int. Cl. F I
G 0 6 F 11/28 (2006. 01) G O 6 F 11/28 3 4 O A
G O 6 F 11/34 (2006. 01) G O 6 F 11/34 S

請求項の数 20 (全 15 頁)

(21) 出願番号	特願2013-518216 (P2013-518216)	(73) 特許権者	512333674
(86) (22) 出願日	平成23年3月15日 (2011. 3. 15)		ヒョンダイ モーター カンパニー
(65) 公表番号	特表2013-535069 (P2013-535069A)		大韓民国, ソウル, ソチョーグ, ヤンゼー
(43) 公表日	平成25年9月9日 (2013. 9. 9)		ドン, 2 3 1 ボンジ
(86) 国際出願番号	PCT/KR2011/001803	(73) 特許権者	512297594
(87) 国際公開番号	W02012/002635		キアモータース コーポレーション
(87) 国際公開日	平成24年1月5日 (2012. 1. 5)		大韓民国, ソウル 1 3 7 - 9 3 8, ソチョ
審査請求日	平成25年1月7日 (2013. 1. 7)		ーグ, ヤンゼードン 2 3 1
(31) 優先権主張番号	PCT/KR2010/006068		
(32) 優先日	平成22年9月7日 (2010. 9. 7)		
(33) 優先権主張国	韓国 (KR)		
(31) 優先権主張番号	10-2010-0061455		
(32) 優先日	平成22年6月28日 (2010. 6. 28)		
(33) 優先権主張国	韓国 (KR)		

最終頁に続く

(54) 【発明の名称】 システムテスト装置

(57) 【特許請求の範囲】

【請求項 1】

複数のプロセス及び前記複数のプロセスに対するランタイム実行情報を含むプロセス制御ブロックを含むシステムに対するシステムテスト装置において、

前記プロセス制御ブロックにテストエージェントをするモジュールと、

テスト対象と関連したイベントが発生する場合、前記テストエージェントを利用してテスト対象をテストコードにフッキングするフッキングモジュールと、

前記テスト対象がフッキングされた場合、前記テスト対象と関連するイベントが発生したプロセスに対するテスト情報を収集するスキャンングモジュールと、

前記スキャンングモジュールにより収集されたテスト情報を格納するロギングモジュールと、を含み、

前記テストエージェントは、前記プロセス制御ブロックに含まれている情報に基づき、
 前記テスト対象を前記テストコードにフッキングし、

前記フッキングモジュールは、前記テストエージェントを介して前記プロセス制御ブロックの原本コードの動作を含むテストコードが行われるように、前記テスト対象を前記テストコードにフッキングすることを特徴とするシステムテスト装置。

【請求項 2】

前記プロセス制御ブロックは、前記複数のプロセスに対する資源使用現況、プロセス状態、優先順位及び関数テーブルのうち少なくとも1つを含むことを特徴とする請求項 1 に

10

20

記載のシステムテスト装置。

【請求項 3】

前記テストコードは、欠陥の類型に従い定義されることを特徴とする請求項 2 に記載のシステムテスト装置。

【請求項 4】

前記フッキングモジュールは、前記欠陥の類型に伴うテスト位置に従いメモリ領域を識別し、識別されたメモリ領域をテストコードにフッキングすることを特徴とする請求項 3 に記載のシステムテスト装置。

10

【請求項 5】

前記テスト情報は、プログラム実行情報、プロセスコンテキスト情報及びシステム状態情報のうち少なくとも 1 つであることを特徴とする請求項 3 に記載のシステムテスト装置。

【請求項 6】

前記スキャニングモジュールは、前記プロセス制御ブロックの性能情報を測定することを特徴とする請求項 2 に記載のシステムテスト装置。

20

【請求項 7】

前記プロセス制御ブロックの性能情報は、プロセッサ使用量、メモリ使用量、ページフォールト (Page Fault)、プロセスコンテキスト情報、プロセスコールスタック、プロセスヒープ使用量、プロセス状態、プロセス優先順位のうち少なくとも 1 つであることを特徴とする請求項 6 に記載のシステムテスト装置。

【請求項 8】

予め設定された時間間隔ごとに前記プロセス制御ブロックの性能情報を測定することを特徴とする請求項 6 に記載のシステムテスト装置。

30

【請求項 9】

前記プロセス制御ブロックのプロセス及びスレッド (Thread) に対する性能情報を測定することを特徴とする請求項 6 に記載のシステムテスト装置。

【請求項 10】

前記プロセス制御ブロックのプロセスに対する性能要素は、アイディー (ID)、プロセスの状態、プロセスの優先順位、ヒープ (Heap) 使用量、プロセスの動作時間、使用時間、カーネル使用時間中 1 つ以上を含むことを特徴とする請求項 9 に記載のシステムテスト装置。

40

【請求項 11】

前記プロセス制御ブロックのスレッドに対する性能要素は、アイディー、運営状態 (Run State)、基本優先順位、現在優先順位、使用時間、カーネル使用時間、コールスタック (Call Stack) 中 1 つ以上を含むことを特徴とする請求項 9 に記載のシステムテスト装置。

【請求項 12】

前記テスト対象と関連するイベントは、前記複数のプロセス間のメッセージ送信、共有データ送信、共有メモリ使用、メッセージキュー使用、同期化オブジェクト使用のうち少なくとも 1 つであることを特徴とする請求項 2 に記載のシステムテスト装置。

50

【請求項 13】

前記テストコードは、前記複数のプロセス間のメッセージ送信、メッセージキュー、共有メモリ及び同期化オブジェクトのうち少なくとも1つの欠陥のタイプを確認するためのコードであることを特徴とする請求項12に記載のシステムテスト装置。

【請求項 14】

前記テスト情報は、前記複数のプロセスのコールスタック情報、前記複数のプロセス間に送受信される通信タイプ、通信アイディ、通信データ及び前記複数のプロセス間の通信で発生した遅延時間のうち少なくとも1つであることを特徴とする請求項12に記載のシステムテスト装置。

10

【請求項 15】

前記テスト情報は、システム動的状態情報、プロセス状態情報、及び前記複数のプロセス間の通信情報を含むことを特徴とする請求項12に記載のシステムテスト装置。

【請求項 16】

複数のプロセス及び前記複数のプロセスに対するランタイム実行情報を含むプロセス制御ブロックを含むシステムに対するシステムテストシステムにおいて、

前記システムを駆動させるための信号の入力を受ける入力装置と、

20

前記入力装置を介し入力された信号により前記システムが動作する間、前記プロセス制御ブロックにテストエージェントを入力し、前記テストエージェントを利用してフッキングモジュールがテスト対象をテストコードにフッキングし、前記複数のプロセスのうちテスト対象に関連するイベントが発生したプロセスに対するテスト情報を収集するシステムテスト装置と、

前記システムテスト装置により収集されたテスト情報を分析する分析装置と、を含み、

前記テストエージェントは、前記プロセス制御ブロックに含まれている情報に基づき、前記テスト対象を前記テストコードにフッキングし、

前記フッキングモジュールは、前記テストエージェントを介して前記プロセス制御ブロックの原本コードの動作を含むテストコードが行われるように、前記テスト対象を前記テストコードにフッキングすることを特徴とするシステムテストシステム。

30

【請求項 17】

前記プロセス制御ブロックは、前記複数のプロセスに対する資源使用現況、プロセス状態、優先順位及び関数テーブルのうち少なくとも1つを含むことを特徴とする請求項16に記載のシステムテストシステム。

【請求項 18】

前記テストコードがフッキングされるテスト位置は、欠陥のタイプに従い変更され得ることを特徴とする請求項16に記載のシステムテストシステム。

40

【請求項 19】

前記テスト情報は、プログラム実行情報、プロセスコンテキスト情報、システム状態情報、プロセッサ使用量、メモリ使用量、ページフォールト (Page Fault)、前記複数のプロセスのコールスタック情報、前記複数のプロセス間に送受信される通信タイプ、通信アイディ、通信データ及び前記複数のプロセス間の通信で発生した遅延時間のうち少なくとも1つであることを特徴とする請求項16に記載のシステムテストシステム。

【請求項 20】

前記分析装置により分析されたテスト結果をイメージ又は音声で出力する出力装置をさらに含む請求項16に記載のシステムテストシステム。

50

【発明の詳細な説明】

【技術分野】

【0001】

本発明はシステムテスト装置に係り、より詳しくは、システムの実行情報を含むプロセス制御ブロックを利用してシステムに対するテスト情報を実時間で収集するシステムテスト装置に関する。

【背景技術】

【0002】

多様なソフトウェア及びハードウェアを含むエンベデッドシステムにおいて、システムの運営環境で相互作用（interaction）する動作が重要である。したがって、ソフトウェアのみで構成されたテストのため仮想データでない、システムが実際運営されながら得ることになる実際データ（live environment with real data）を利用する動作が重要なので、システムをテストしようとする場合にも、実際運用データを利用してシステムテストを行ってこそ潜在的欠陥、又は実際運用上発生する運用上欠陥を効果的に探し出すことができる方案が求められる。

【0003】

さらに、システムを構成するソフトウェアそれぞれを別途独立的にテストするためには、連結される相対コンポーネントを代替するための装備を備えなければならないなど、テスト資源と費用が多く要するため、システムに含まれた全てのプロセスを実時間でテストすることができ、メモリ空間のオーバーヘッド又は当該コード遂行による性能オーバーヘッドを最小化するテスト装置が求められる。

【発明の概要】

【発明が解決しようとする課題】

【0004】

本発明は、フッキング技術を利用してシステムに対するメモリ誤謬テスト、性能テスト、及びシステムの内部に含まれたプロセス間の通信テストを行うことにより、実際運営されるシステムに対するテストが可能であり、システムのランタイム状況でも動的テストを行うことができ、システムが動作する間発生する欠陥を確認することができるシステムテスト装置を提供することにその目的がある。

【課題を解決するための手段】

【0005】

本発明によるシステムテスト装置は、複数のプロセス及び前記複数のプロセスに対するランタイム実行情報を含むプロセス制御ブロックを含むシステムに対するシステムテスト装置において、前記プロセス制御ブロックにテストエージェントを入力する入力モジュールと、テスト対象と関連したイベントが発生する場合、前記テストエージェントを利用してテスト対象をテストコードにフッキングするフッキングモジュールと、前記テスト対象がフッキングされた場合、前記テスト対象と関連するイベントが発生したプロセスに対するテスト情報を収集するスキャニングモジュールと、前記スキャニングモジュールにより収集されたテスト情報を格納するロギングモジュールと、を含み、前記テストエージェントは、前記プロセス制御ブロックに含まれている情報に基づき、前記テスト対象を前記テストコードにフッキングし、前記フッキングモジュールは、前記テストエージェントを介して前記プロセス制御ブロックの原本コードの動作を含むテストコードが行われるように、前記テスト対象を前記テストコードにフッキングすることを特徴とする。

【0006】

本発明によるシステムテストシステムは、複数のプロセス及び前記複数のプロセスに対するランタイム実行情報を含むプロセス制御ブロックを含むシステムに対するシステムテストシステムにおいて、前記システムを駆動させるための信号の入力を受ける入力装置と、前記入力装置を介し入力された信号により前記システムが動作する間、前記プロセス制

御ブロックにテストエージェントを入力し、前記テストエージェントを利用してフッキングモジュールがテスト対象をテストコードにフッキングし、前記複数のプロセスのうちテスト対象と関連するイベントが発生したプロセスに対するテスト情報を収集するシステムテスト装置と、前記システムテスト装置により収集されたテスト情報を分析する分析装置と、を含み、前記テストエージェントは、前記プロセス制御ブロックに含まれている情報に基づき、前記テスト対象を前記テストコードにフッキングし、前記フッキングモジュールは、前記テストエージェントを介して前記プロセス制御ブロックの原本コードの動作を含むテストコードが行われるように、前記テスト対象を前記テストコードにフッキングすることを特徴とする。

【発明の効果】

10

【0007】

本発明によれば、ソースコードでない実際運営されるコードを対象にするので、システムが実際に運用される間にもシステムに対するテストを行うことができる。したがって、システムが実際動作する間メモリ誤謬、システム性能、プロセス間の通信をテストすることができる。

さらに、それぞれのプロセスに対する個別的なコード挿入でない、プロセス制御ブロックに含まれた情報を活用したフッキング方法を利用するので、システムオーバーヘッドを最小化することができ、動作するシステムを妨ぐことなくシステムに対するテストを行うことができる。

さらに、プロセス間の共有資源が制限された場合も、実時間でテスト情報を収集しテスト結果を確認することができる。

20

【図面の簡単な説明】

【0008】

【図1】本発明の一実施形態に係るシステムテスト装置が利用される全体システムを示す図である。

【図2】本発明の一実施形態に係るシステムテスト装置の構成を示す図である。

【図3】本発明の一実施形態に係るシステムテスト方法を示すフローチャートである。

【図4a】本発明の一実施形態に係るメモリテスト方法、性能テスト方法及びプロセス間の通信テスト方法を示すフローチャートである。

【図4b】本発明の一実施形態に係るメモリテスト方法、性能テスト方法及びプロセス間の通信テスト方法を示すフローチャートである。

30

【図4c】本発明の一実施形態に係るメモリテスト方法、性能テスト方法及びプロセス間の通信テスト方法を示すフローチャートである。

【発明を実施するための形態】

【0009】

以下、添付の図面を参照して、本発明の好ましい実施例を説明する。

【実施例】

【0010】

本発明は、システムが実際運営される環境でのテストであり、システムが開始され終了されるまでの動的状況でシステムをテストすることができる装置及び方法に関する。本発明は、フッキング技法を利用して、システム内で動作する全てのプロセスに対するテストを行うことができる。

40

【0011】

図1は、本発明の一実施形態に係るシステムテスト装置が利用される全体システムを示す図である。図1に示す通り、全体システムはシステムテスト装置100、システム200、入力装置300、分析装置400及び出力装置500を含む。システムテスト装置100は、システム200に対するメモリ欠陥、システム200の性能又はシステム200に含まれた2つ以上のプロセス間の通信をテストするための装置である。システムテスト装置100は、図1に示す通り、システム200と独立的に備えられてもよく、システム200の内部に含まれる形態に具現されてもよい。

50

【0012】

システム200は、システムテスト装置100のテスト対象となるシステムであって、多様なソフトウェア及びハードウェアが統合されたエンベデッドシステムである。システムテスト装置100のテスト対象となるシステムは、複数のプロセスを含む1つ以上のシステムを意味する。したがって、複数のシステムのメモリ欠陥又はシステム性能もテスト対象となり得、1つのシステムに含まれる複数のプロセス間の通信だけでなく、互いに異なるシステムに含まれたプロセス等の間の通信もシステムテスト装置100のテスト対象となり得る。

【0013】

前記システム200は、マルチプロセスが通信する全てのエンベデッドシステム及びコンピュータシステムであって、例えば、インフォテインメントシステム、マルチメディアシステム、有無線ネットワークシステムなどになり得る。例えば、システム200が車両インフォテインメントシステムの場合、システム200はAVN(Audio Video Navigation)機能を中心にTDMB(Terrestrial Digital Multimedia Broadcasting)、TPEG(Transport Protocol Expert Group)、IPOD、USB、ブルートゥース(Bluetooth(登録商標))、VR(Voice Recognition)、PGS(Parking Guidance System)、RDS(Radio Data System)、VCDC(Vehicle CDC)、DIS(Driver Information System)、RSE(Rear Seat Entertainment)、HEV(Hybrid Electric Vehicle) display、AHU(Air Handling Unit) displayなどを含む。

【0014】

入力装置300は、システムテスト装置100及びシステム200を駆動させるための使用者入力装置である。入力装置300は、キーボード又はタッチスクリーンに具現されてもよく、入力装置300を介し入力される信号に応じてシステム200が駆動される。さらに、入力装置300を介し入力される信号に応じてシステムテスト装置100がシステムテストのための動作を行う。

【0015】

分析装置400は、システムテスト装置100により収集されたテスト情報を分析しシステム200のテスト結果を確認する。前記システムテスト装置100は分析装置400と1つの装置に具現され、テスト情報を収集すると共に、収集されたテスト情報を分析しテスト結果を確認することもできる。

【0016】

出力装置500は、システムテスト装置100により収集されたテスト情報、又は分析装置400により分析されたテスト結果をイメージ又は音声で出力する装置である。出力装置500を介し、使用者はシステムテスト結果を確認することができる。

【0017】

図2は、本発明の一実施形態に係るシステムテスト装置の構成を示す図である。図2に示す通り、システムテスト装置100は挿入モジュール101、フッキングモジュール103、スキャンングモジュール105及びロギングモジュール107を含む。図2で、システムテスト装置100と連結されテスト対象となるシステム200は、2つ以上のプロセスを含む1つのシステムと仮定する。

【0018】

システム200は、プロセス制御ブロック(Process Control Block)201を含む。プロセス制御ブロック201は、システム200に含まれた全てのプロセスに対するランタイム実行情報を有する。ここで、ランタイム実行情報とは、システム200が動作する間全てのプロセス間の実行情報を意味する。したがって、プロセス制御ブロック201は、システム200に含まれた全てのプロセスに対する資源使用現況

10

20

30

40

50

、プロセス状態、優先順位、関数テーブルなどを含むことができる。より詳しくは、プロセスID、プロセス名、現行プロセス、現行スレッド、システムで行われているプロセスリスト、モジュールリスト、システムページサイズ、使用可能な物理メモリサイズ、CPU使用量、使用中の資源リスト、要請中のI/Oデバイス状態、オープンされたファイル情報、レジスタ情報などを含むことができる。したがって、全てのシステム内で前記言及された情報を含むデータ構造又は装置は、本発明で意味するプロセス制御ブロックに該当する。

【0019】

挿入モジュール101は、プロセス制御ブロック201にテストエージェントを挿入する。テストエージェントは、プロセス制御ブロック201に含まれた情報に基づきテスト対象をテストコードにフッキングし、テスト情報を収集することができる。テストコードがフッキングされるテスト位置は、欠陥の類型に従い定義され得る。本発明で、テストコードにフッキングされるテスト対象は、テスト対象となるメモリ領域、システムの性能情報を含むコード、又は通信関連イベントを発生させるコードなどになり得る。本発明で、テストエージェントがプロセス制御ブロック201に挿入される前に、先ずシステム200は正常に動作しており、システム200はシステムテスト装置100の動作により影響を受けず正常に動作する。

【0020】

さらに、挿入モジュール101は、システム200で動作しているプロセスリストを確認し、確認されたプロセスリストに基づきテストエージェントを挿入することができる。フッキングモジュール103は、前記挿入モジュール101により挿入されたテストエージェントを利用してテスト対象をテストコードにフッキングする。前記テストコードはテストしようとする対象及び欠陥の類型に従い変更される。即ち、メモリ欠陥テスト、性能テスト及び通信テストに従いテストコードがそれぞれ変化し、通信テストにおいてもテストしようとする欠陥の類型に従いテストコードが変化し得る。さらに、フッキングモジュール103は、テスト対象がテストコードにフッキングされるテスト位置を変更することができる。

【0021】

テストコードがフッキングされるとのことは、動作しているシステム200内でテスト対象と関連イベントが発生する場合、挿入されたテストエージェントが原本コードの動作を含むテストコードが行われるようOSのプロセス制御ブロックをフッキングするのことを意味する。プロセス制御ブロックに含まれる情報から、フッキングモジュール103はテスト対象と関連するイベントが発生するのかわ確認することができる。

【0022】

スキニングモジュール105は、テストコードがフッキングされた場合、前記テスト対象と関連するイベントが発生したプロセスに対するテスト情報を収集する。ここで、テスト情報は、既に発生したイベントに対する情報だけでなく、テストコードフッキングに伴い新たに発生するイベントまで含むことができる。収集されるテスト情報はテスト段階、テスト目的、テストしようとする欠陥の類型に従い変化する。したがって、スキニングモジュール105は、システムテスト結果の分析に必要なシステム動的状態情報、例えば、関数パラメータ及び返還値、性能測定値、共有資源使用リストなどを収集する。

【0023】

ロギングモジュール107は、前記スキニングモジュール105により収集されたテスト情報を格納空間に格納する。格納空間はロギングモジュール107に備えられてもよく、ロギングモジュール107と別個の装置に具現されてもよい。図1に示した分析装置400は、スキニングモジュール105により収集されたテスト情報、又はロギングモジュール107により格納されたテスト情報を分析することができる。

【0024】

図3は、本発明の一実施形態に係るシステムテスト方法を示すフローチャートである。図3に示す通り、テスト対象となるシステム200のプロセス制御ブロック201にテス

10

20

30

40

50

トエージェントが挿入される（S110）。システム200は、テストエージェントが挿入される前から正常に動作している。

【0025】

次に、テスト対象と関連するイベントが発生する場合、挿入されたテストエージェントを介しテスト対象がテストコードにフッキングされる（S120）。テスト対象がテストコードにフッキングされることにより、前記イベントが発生したプロセスの代わりにテストコードが行われ得る。次に、テスト対象がテストコードにフッキングされた場合、前記テスト対象と関連するイベントが発生したプロセスに対するテスト情報が収集される（S130）。次に、収集されたテスト情報は格納される（S140）。ここで、テスト情報は、テスト対象がテストコードにフッキングされている間にのみ収集され得、テスト対象がテストコードにフッキングされた後、システムが動作する間全て収集されることもある。

10

【0026】

図4は、本発明の一実施形態に係るメモリ欠陥テスト方法、性能テスト方法及びプロセス間の通信テスト方法を示すフローチャートである。図4aは、メモリ欠陥テストを行うためのシステムテスト方法を示す図である。先ず、欠陥モデルが定義される（S150）。ここで、欠陥モデルとは、欠陥の類型に従いテストされなければならないメモリ位置及び欠陥の発見／復旧のためのテスト情報を意味する。定義された欠陥モデルに従い、システム200が正常に動作する間テスト情報を収集することができるテストコードが生成され得る。

20

【0027】

次に、システム200のプロセス制御ブロック201にテストエージェントが挿入される（S151）。テストエージェントは、システム200で動作しているプロセスリスト中少なくとも1つに挿入され得る。次に、フッキングモジュール103は、プロセス制御ブロック201に含まれた情報及び欠陥モデルに基づきフッキングするメモリ領域を識別し、識別されたメモリ領域にテストコードをフッキングする（S152）。メモリ領域を識別した結果、テスト位置がSYSTEM API（Application Process Interface）関数の場合、フッキングモジュール103はプロセス制御ブロック201のAPI Setをフッキング領域と識別する。テスト位置がDLL（Dynamic Linking Library）EXPORT関数の場合、フッキングモジュール103はプロセス制御ブロック201のIAT（Import Address Table）をフッキング領域と識別する。その他の場合、フッキングモジュール103はプロセス制御ブロック201のコード空間をフッキング領域と識別する。

30

【0028】

図4aで、テストコードがフッキングされるとのことは、動作しているシステム200内でテスト対象であるメモリ位置に、メモリに対する書込み、読取り、接近などのイベントが発生する場合、挿入されたテストエージェントが原本コードの動作を含むテストコードが行われるようOSのプロセス制御ブロックをフッキングするとのことを意味する。

【0029】

次に、テスト対象であるメモリ状態に対する情報が収集される（S153）。例えば、欠陥原因の識別に必要なプログラム実行情報が収集され、欠陥位置の識別に必要なプロセスコンテキスト情報が収集され、メモリ性能分析に必要なシステム状態情報が収集され得る。次に、S153段階で収集されたテスト情報が格納される（S154）。

40

【0030】

図4bは、性能テストを行うためのシステムテスト方法を示す図である。本発明に係るシステムテスト方法は、性能ボトルネックとその原因を分析するため、プロセス制御ブロック201のデータ中ページフォールト率、プロセッサ使用率のようなシステム実行情報等をフッキングしシステム性能を分析することができる。ここで、性能は、システムやコンポーネント等が与えられたシステムの制約条件下で機能を行う程度を意味する。性能テストは、特定性能の要求事項をシステムが満足するのかに関する評価である。このような

50

性能テストを介して性能ボトルネックとその原因を分析し、これを解決することによりシステムの性能を改善することができる。

【0031】

システムの性能ボトルネックは、メモリ、I/Oデバイス、ネットワークのような制限された資源等に対する競争でシステムの性能が低下する現象である。システム性能ボトルネックの原因は、資源不足、共有資源競合、資源独占、資源の誤った構成、又は資源の誤った動作のように多様である。先ず、欠陥モデルが定義される(S160)。ここで、欠陥モデルとは、欠陥の類型に従いテストされなければならない性能ボトルネックの位置及び性能ボトルネックを判定するシステム状態を表すテスト情報を意味する。定義された欠陥モデルに従い、システム200が正常に動作する間テスト情報を収集することができる

10

【0032】

挿入モジュール101は、プロセス制御ブロック201にテストエージェントを挿入する(S161)。挿入モジュール101は、システム200で動作しているプロセスリストを確認し、確認されたプロセスリストに基づきテストエージェントを挿入することができる。即ち、現行プロセス(Current Process)に関するプロセス制御ブロック201のbase addressは、特定のメモリ空間に管理されるか、場合に応じて予め固定されたアドレスに管理されるため、現行プロセスのプロセス制御ブロック201の情報が分かる。

【0033】

20

次に、フッキングモジュール103は、テスト対象にテストコードをフッキングする(S162)。テスト対象がテストコードにフッキングされた場合、スキャンングモジュール105はシステムの性能情報を収集する(S163)。例えば、timer interruptを用いて指定された時間間隔(e.g. 1sec、100msec)ごとに性能データを測定することができる。さらに、システム性能をシステム単位だけでなく、プロセスとスレッドの単位にまで分析するため、プロセス制御ブロック201のプロセスとスレッドリスト等を循環しながら性能情報を測定することができる。

【0034】

システム性能ボトルネックが発生する位置を詳しく追跡するため、スキャンングモジュール105はスレッド別にコールスタック(Call stack)情報を格納することができ、各モニタリング単位別に測定される性能情報は、次の通りである。プロセス制御ブロック201の性能情報は、プロセッサ使用量、メモリ使用量、ページフォールト(Page Fault)、プロセスコンテキスト情報、プロセスコールスタック、プロセスヒープ使用量、プロセス状態、プロセス優先順位中1つ以上を含むことができる。プロセス制御ブロック201のプロセスに対する性能情報は、アイディー(ID)、プロセスの状態、プロセスの優先順位、ヒープ(Heap)使用量、プロセスの動作時間、使用時間、カーネル使用時間中1つ以上を含むことができる。さらに、プロセス制御ブロック201のスレッドに対する性能情報は、アイディー、運営状態(Run State)、基本優先順位、現在優先順位、使用時間、カーネル使用時間、コールスタック情報中1つ以上を含むことができる。

30

40

【0035】

次に、ロギングモジュール107は、収集されたテスト情報を格納空間に格納する(S164)。性能テストを行うためのシステムテスト方法は、ページフォールトを基準にメモリボトルネックを判断することができる。例えば、ページフォールトが高ければメモリボトルネックであり得る。ページフォールトとは、プログラムが自分のアドレス空間には存在するがシステムのメモリには現在ないデータやコードに接近を試みた場合に発生する現象である。ページフォールトが発生すれば、運営体制はそのデータをメモリに持って来て、まるでページフォールトが全く発生しなかったもののようプログラムが継続的に作動するようにする。運営体制のページフォールトに対する例外処理のためアプリケーションの処理時間は遅延され、全体システム性能に影響を及ぼす。

50

【0036】

さらに、本発明に係るシステムテスト方法は、メモリ使用量を介し性能ボトルネックを識別することができる。システムのメモリは物理メモリ使用量、仮想メモリ使用量があり、プロセス別にヒープメモリ使用量で区分することができる。本発明に係るシステムテスト方法は、このようなメモリ使用量等の合計を基準に性能ボトルネックを判断する。

【0037】

さらに、本発明に係るシステムテスト方法は、プロセッサ使用量（或いはCPU使用量）を介し性能ボトルネックを識別することができる。例えば、本発明に係るシステムテスト方法は、プロセッサ使用量が高く維持されながら余裕メモリがある場合は、CPUにボトルネックがあると判断することができる。その反面、プロセッサ使用量が高くメモリも枯渇していれば、性能問題はCPUよりメモリボトルネックと判断することができる。

10

【0038】

さらに、本発明に係るシステムテスト方法は、プロセス使用量（Processor Usage）に基づき性能ボトルネックを識別することができる。プロセス使用量はシステムの実行時間であって、全体CPU使用量から待機時間（Idle Time）を除いた時間を意味する。さらに、本発明に係るシステムテスト方法は、使用時間（User Time）に従い性能ボトルネックを判断することができる。使用時間は、実行がユーザー空間で留まった時間を意味し、即ちアプリケーションの実行時間を意味する。さらに、本発明に係るシステムテスト方法は、カーネル使用時間（Kernel Time）に従い性能ボトルネックを判断することができる。カーネル使用時間は、実行がカーネル空間で留まった時間であって、カーネルのサービス処理時間を意味する。

20

【0039】

図4cは、複数のプロセス間の通信に対するテストを行うためのシステムテスト方法を示す図である。図4cに係るシステム方法を行うシステムテスト装置100は、システム200に含まれた全てのプロセス間の資源共有及び意思伝達が正確になされるのかを判断可能なテスト情報を収集することができる。

【0040】

先ず、欠陥モデルが定義される（S170）。ここで、欠陥モデルとは、欠陥の類型に従いテストされなければならないプロセス間の通信位置及び欠陥の発見／復旧のためのテスト情報を意味する。定義された欠陥モデルに従い、システム200が正常に動作する間テスト情報を収集することができるテストコードが生成され得る。テスト対象となるシステム200のプロセス制御ブロック201にテストエージェントが挿入される（S171）。挿入モジュール101は、前記プロセスリスト中2つ以上のプロセスをテスト対象に選定し、選定されたプロセス間の通信をテストするためのテストエージェントを挿入することもできる。

30

【0041】

次に、複数のプロセス間に通信関連イベントが発生する場合、挿入されたテストエージェントを利用しテスト対象がテストコードにフッキングされる（S172）。テスト対象がテストコードにフッキングされるとのことは、動作しているシステム200内で複数のプロセス間に通信関連イベントが発生する場合、挿入されたテストエージェントが原本コードの動作を含むテストコードが行われるようOSのプロセス制御ブロックをフッキングするとのことを意味する。プロセス制御ブロックに含まれる情報から、フッキングモジュール103は如何なるプロセスの間に如何なる種類の通信関連イベントが発生するのかを確認することができる。ここで、通信関連イベントとは、複数のプロセス間にメッセージ送信、共有データ送信、共有メモリ使用、メッセージキュー使用、同期化客体使用などを意味する。

40

【0042】

テストコードは、前記複数のプロセス間のメッセージ送信、メッセージキュー、共有メモリ又は同期化客体と関連する欠陥の類型を確認するためのコードである。例えば、テストコードはSendMessage（）、PostMessage（）のような第1の関

50

数、及びテストしようとする欠陥の類型に従い必要な第2の関数を含み、システムに含まれたプロセス等はそれぞれ第1の関数を含む。したがって、第1のプロセス及び第2のプロセスの間に通信関連イベントが発生する場合、フッキングモジュール103は、挿入された通信エージェントを利用し第1のプロセス及び第2のプロセスで第1の関数が行われる段階で、第1のプロセス及び第2のプロセスの第1の関数に代えてテストコードを行わせることにより、テストコードをフッキングする。第1のプロセス及び第2のプロセスで第1の関数が行われる段階は、プロセス制御ブロックに含まれた情報から確認される。一方、テスト対象が複数のプロセス間の通信なので、テストしようとする欠陥の類型に従い送信側又は受信側に該当する一方のプロセスにのみテストコードをフッキングすることもできる。

10

【0043】

次に、スキャンングモジュール105は、テスト対象がフッキングされた場合、前記通信関連イベントが発生した複数のプロセス間の通信に対するテスト情報を収集する（S174）。ここで、複数のプロセス間の通信とは、既に発生した通信関連イベントだけでなく、テストコードフッキングに従い発生する通信関連イベントまで含む。収集されるテスト情報はテスト段階、テスト目的、テストしようとする欠陥の類型に従い変化する。したがって、スキャンングモジュール105は、複数のプロセス間の通信に対し欠陥の判定に必要なシステム動的状態情報、例えば、関数パラメータ及び返還値、性能測定値、共有資源使用リストなどを収集する。テスト情報は、前記複数のプロセスのコールスタック情報、前記複数のプロセス間に送受信される通信類型、通信アイディ、通信データ、前記複数のプロセス間の通信で発生した遅延時間などを含むことができる。

20

【0044】

次に、ロギングモジュール107は、前記スキャンングモジュール105により収集されたテスト情報を格納空間に格納する（S175）。格納空間は、ロギングモジュール107に備えられてもよく、ロギングモジュール107と別個の装置に具現されてもよい。図1に示した分析装置400は、スキャンングモジュール105により収集されたテスト情報又はロギングモジュール107により格納されたテスト情報を分析することができる。

【0045】

複数のプロセス間の通信に対するテストを行うため利用されるテストコードは、プロセス間のメッセージ送信、メッセージキュー、共有メモリ又は同期化客体と関連する欠陥の類型を確認するためのコードになり得る。メッセージ送信（Message Passing）と関連する欠陥の類型は、指定時間内メッセージ送信失敗（同期）、メッセージデッドロック（同期）、メッセージ処理遅延（同期）、メッセージ損失（同期）、メッセージ変更（同期）、メッセージキューへの挿入失敗（非同期）、メッセージ処理遅延（非同期）、メッセージ損失（非同期）、又はメッセージ変更（非同期）に分類され得る。

30

【0046】

メッセージキュー（Message Queue）と関連する欠陥の類型は、メッセージキュー生成失敗、メッセージキュー閉じ失敗、メッセージキューからの読取り失敗、又はメッセージキューへの書込み失敗に分類され得る。共有メモリ（Shared Memory）と関連する欠陥の類型は、共有メモリ生成失敗でもあり得る。同期化客体（Synchronization Object）と関連する欠陥の類型は、生成していない同期化客体使用、生成した同期化客体使用しない、又は使用した同期化客体解除しないに分類され得る。したがって、各欠陥の類型に従いテストコードが変更される。

40

【0047】

前記のように、本発明に係るシステムテスト装置100は、動作しているシステム200に影響を与えずともテストエージェント挿入、テストコードフッキング、テスト情報収集を実施することができ、前記システムテスト装置100によるシステムテストは、システムに含まれたプロセス制御ブロックに含まれた情報に基づき実施される。

【0048】

50

本発明は、プロセッサが読取り可能な記録媒体にプロセッサが読取り可能なコードとして具現することが可能である。プロセッサが読取り可能な記録媒体は、プロセッサにより読み取られ得るデータが格納される全ての種類の記録装置を含む。プロセッサが読取り可能な記録媒体の例にはROM、RAM、CD-ROM、エムティー、フロッピー(登録商標)、光データ格納装置などがある。さらに、プロセッサが読取り可能な記録媒体はネットワークで連結されたコンピュータシステムに分散され、分散方式でプロセッサが読取り可能なコードが格納され行われ得る。

【0049】

本発明は、前述の一実施形態に限定されず、請求の範囲で請求する本発明の要旨を外れることがなく、当該発明の属する技術分野で通常の知識を有する者により多様な変形実施が可能であり、このような変形実施等は、本発明の技術的思想や展望から個別的に理解されてはならないはずである。

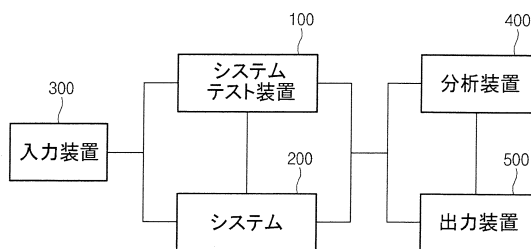
【産業上の利用可能性】

【0050】

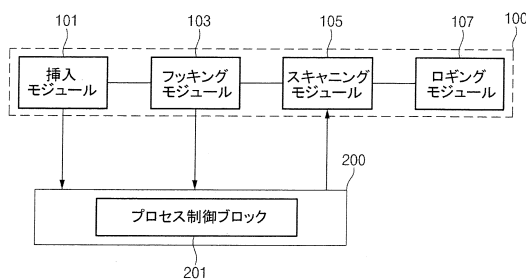
本発明は、システムが実際に動作する間メモリ誤謬、システム性能、プロセス間の通信をテストすることができる。

10

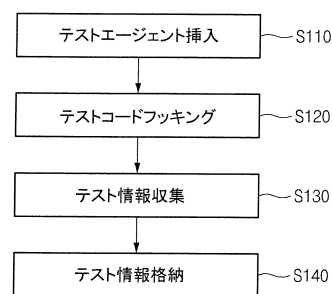
【図1】



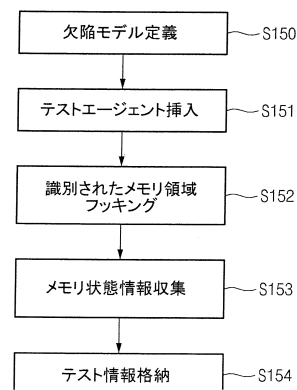
【図2】



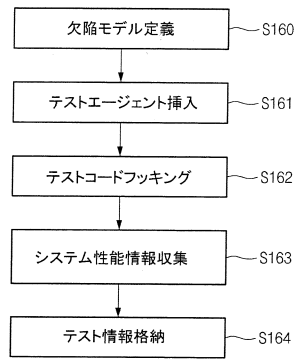
【図3】



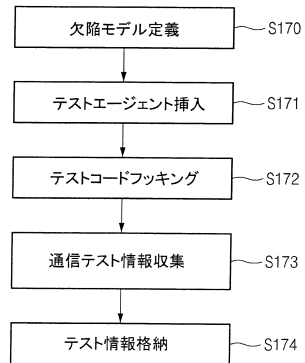
【図4a】



【図 4 b】



【図 4 c】



フロントページの続き

(73)特許権者 505469540

エファ・ユニバーシティ・インダストリー・コラボレーション・ファウンデーション
EWAH UNIVERSITY INDUSTRY COLLABORATION FOUNDATION

大韓民国120-750ソウル、ソデムンク、デヒョンドン11-1番、エファ・ウーマンズ・ユニバーシティ

(74)代理人 110000051

特許業務法人共生国際特許事務所

(72)発明者 チョエ, ビョン ジュ

大韓民国, ソウル 135-788, カンナムグ, アックジョンドン ヒョンダイアパート, 74-1305

(72)発明者 ソ, ジュ ヨン

大韓民国, ソウル 138-200, ソンパグ, ムンジョンドン, オリピック フェミリ タウン, 105-304

(72)発明者 ヤン, スン ワン

大韓民国, ギョンギード 435-704, グンポーシ, グァンジョンドン, サムソン ジャンミアパート, 1137-2202

(72)発明者 イム, ジン ヨン

大韓民国, ギョンギード 441-840, スウォンシ, グォンソング, クムゴクドン, サムイク 1チャアパート, 102-406

(72)発明者 キム, ヨン スウ

大韓民国, ギョンギード 446-588, ヨンインシ, ギフング, シンガルドン, 697ボンジ, ノクオオン マウル セチョンニョン グリンビル 509-503

(72)発明者 クォン, ヘ ヨン

大韓民国, ギョンギード 446-567, ヨンインシ, ギフング, ボジョンドン, ヒョンダイ アイパーク 1チャ アパート, 202-601

(72)発明者 オ, ジョン ソク

大韓民国, ギョンギード 428-801, グァンミョンシ, グァンミョンユドン, 70-13, 2F

(72)発明者 ジャン, スン ヨン

大韓民国, ギョンギード 412-741, ゴヤンシ, ドクヤング, ファジョン2ドン, ビョルビット マウル ゴンヨン アパート, 1004-1101

審査官 多賀 実

(56)参考文献 特開2009-265823 (JP, A)

特開平10-269105 (JP, A)

特開平04-286035 (JP, A)

Mark Russinovich著 吉川 邦夫訳, Windows NTのシステムコールをフックする, Dr. Dobbs' Journal Japan, 日本, 株式会社翔泳社, 1997年 5月 1日, 第6巻, 第6号, p. 39-44

Tim Bird 外2名, 民生家電用に最適化されたCE Linuxの全容(後編), Interface, 日本, CQ出版株式会社, 2005年 8月 1日, 第31巻, 第8号, p. 165-174

秋田英行, 第10回 各種トレースツールの利用, オープンソースマガジン, 日本, ソフトバンククリエイティブ株式会社, 2006年 9月 1日, 第15巻, 第9号, p. 109-118

(58)調査した分野(Int.Cl., DB名)

G06F11/28-11/36