(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第5367556号
(P5367556)

(45)発行日　平成25年12月11日(2013.12.11)　(24)登録日　平成25年9月20日(2013.9.20)

(51)Int.Cl.　F I
G06F 11/28 (2006.01)　G06F 11/28　J
G06F 11/28　310B
G06F 11/28　340A

請求項の数 3　(全 10 頁)

(21)出願番号　特願2009-291746(P2009-291746)
(22)出願日　平成21年12月24日(2009.12.24)
(65)公開番号　特開2011-134033(P2011-134033A)
(43)公開日　平成23年7月7日(2011.7.7)
審査請求日　平成23年12月1日(2011.12.1)

(73)特許権者　000006013
三菱電機株式会社
東京都千代田区丸の内二丁目７番３号
(74)代理人　100094916
弁理士　村上　啓吾
(74)代理人　100073759
弁理士　大岩　増雄
(74)代理人　100093562
弁理士　児玉　俊英
(74)代理人　100088199
弁理士　竹中　岑生
(72)発明者　浦　政博
東京都千代田区丸の内二丁目７番３号　三菱電機株式会社内

最終頁に続く

(54)【発明の名称】デバッグ支援方法

(57)【特許請求の範囲】
【請求項１】
同一の応用プログラムおよび同一のＯＳを搭載した第１および第２の仮想環境で、上記第１の仮想環境は１個のＣＰＵにて上記応用プログラムが実行され上記応用プログラムの第１のプログラムデータを作成し、上記第２の仮想環境は複数個のＣＰＵにて上記応用プログラムが実行され上記応用プログラムの第２のプログラムデータを作成し、第３の仮想環境は上記第１および第２の仮想環境上の上記第１および第２のプログラムデータを比較して差異を検出し、
上記第１および第２の仮想環境は上記応用プログラムの実行に関するデータをトレースしてトレースデータをそれぞれ作成し、
上記第３の仮想環境は上記第１および第２のプログラムデータを比較して差異を検出すると、上記第１および第２の仮想環境にトレース停止命令を送信し、
上記第１および第２の仮想環境は、上記トレース停止命令を受信するとトレースを停止するデバッグ支援方法。
【請求項２】
同一の応用プログラムおよび同一のＯＳを搭載した第１および第２の仮想環境で、上記第１の仮想環境は１個のＣＰＵにて上記応用プログラムが実行され上記応用プログラムの第１のプログラムデータを作成し、上記第２の仮想環境は複数個のＣＰＵにて上記応用プログラムが実行され上記応用プログラムの第２のプログラムデータを作成し、第３の仮想環境は上記第１および第２の仮想環境上の上記第１および第２のプログラムデータを比較し

て差異を検出し、
上記第１および第２の仮想環境は第１および第２のデバッガをそれぞれ有し、上記第３の仮想環境が上記第１および第２のプログラムデータを比較して差異を検出すると、上記第１および第２の仮想環境にＯＳ停止命令を送信し、
上記第１および第２の仮想環境は、上記ＯＳ停止命令を受信すると第１および第２のデバッガを起動するデバッグ支援方法。
【請求項３】
上記第１および第２の仮想環境のプログラムデータは、メッセージキュであることを特徴とする請求項１または請求項２に記載のデバッグ支援方法。
【発明の詳細な説明】
【技術分野】
【０００１】
この発明は、ＣＰＵを複数個使用した際に発生する応用プログラムの障害を容易にデバッグすることができるデバッグ支援方法に関するものである。
【背景技術】
【０００２】
従来のデバッグ支援方法は、ＣＰＵが複数個になった場合の動作が複雑になるため障害が発生した場合にその原因を追及するために多くの時間を費やしている（例えば、特許文献１参照）。
【先行技術文献】
【特許文献】
【０００３】
【特許文献１】特開２００６－３９７６３号公報
【発明の概要】
【発明が解決しようとする課題】
【０００４】
従来のデバッグ支援方法は、ＣＰＵが複数個になった場合、動作が複雑となり障害の発生原因、障害にいたるロジックが見つけにくいという問題点があった。
【０００５】
この発明は上記のような課題を解決するためになされたものであり、ＣＰＵの数が増えたことにより発生する障害の原因を容易にデバッグすることができるデバッグ支援方法を提供することを目的とする。
【課題を解決するための手段】
【０００６】
この発明は、同一の応用プログラムおよび同一のＯＳを搭載した第１および第２の仮想環境で、第１の仮想環境は１個のＣＰＵにて応用プログラムが実行され応用プログラムの第１のプログラムデータを作成し、第２の仮想環境は複数個のＣＰＵにて応用プログラムが実行され応用プログラムの第２のプログラムデータを作成し、第３の仮想環境は第１および第２の仮想環境上の第１および第２のプログラムデータを比較して差異を検出し、
上記第１および第２の仮想環境は上記応用プログラムの実行に関するデータをトレースしてトレースデータをそれぞれ作成し、
上記第３の仮想環境は上記第１および第２のプログラムデータを比較して差異を検出すると、上記第１および第２の仮想環境にトレース停止命令を送信し、
上記第１および第２の仮想環境は、上記トレース停止命令を受信するとトレースを停止するものである。
また、この発明は、同一の応用プログラムおよび同一のＯＳを搭載した第１および第２の仮想環境で、上記第１の仮想環境は１個のＣＰＵにて上記応用プログラムが実行され上記応用プログラムの第１のプログラムデータを作成し、上記第２の仮想環境は複数個のＣＰＵにて上記応用プログラムが実行され上記応用プログラムの第２のプログラムデータを作成し、第３の仮想環境は上記第１および第２の仮想環境上の上記第１および第２のプロ

グラムデータを比較して差異を検出し、
上記第１および第２の仮想環境は第１および第２のデバッガをそれぞれ有し、上記第３の仮想環境が上記第１および第２のプログラムデータを比較して差異を検出すると、上記第１および第２の仮想環境にＯＳ停止命令を送信し、
上記第１および第２の仮想環境は、上記ＯＳ停止命令を受信すると第１および第２のデバッガを起動するものである。

【発明の効果】

【０００７】

この発明のデバッグ支援方法は、上記のように行われているため、
ＣＰＵを複数個使用した際に発生する応用プログラムの障害を容易にデバッグすることができる。

【図面の簡単な説明】

【０００８】

【図１】この発明の実施の形態１のデバッグ支援方法に用いられるシステムの構成を示す図である。

【図２】図１に示したデバッグ支援方法の動作を説明するためのフローチャートである。

【図３】１個のＣＰＵにて処理を行う際の動作を示したフローチャートである。

【図４】２個のＣＰＵにて処理を行う際の動作を示したフローチャートである。

【図５】この発明の実施の形態２のデバッグ支援方法に用いられるシステムの構成を示す図である。

【図６】この発明の実施の形態３のデバッグ支援方法に用いられるシステムの構成を示す図である。

【図７】この発明の実施の形態４のデバッグ支援方法に用いられるシステムの構成を示す図である。

【図８】図５に示したデバッグ支援方法の動作を説明するためのフローチャートである。

【図９】この発明の実施の形態５のデバッグ支援方法に用いられるシステムの構成を示す図である。

【図１０】図７に示したデバッグ支援方法の動作を説明するためのフローチャートである。

【発明を実施するための形態】

【０００９】

実施の形態１.

以下、本願発明の実施の形態について説明する。図１はこの発明の実施の形態１におけるデバッグ支援方法に用いられるシステムの構成を示す図、図２は図１に示したデバッグ支援方法の動作を説明するためのフローチャート、図３は１個のＣＰＵにて処理を行う際の動作を示したフローチャート、図４は２個のＣＰＵにて処理を行う際の動作を示したフローチャートである。図において、第１の仮想環境としての第１の仮想ＯＳ１１０と第２の仮想環境としての第２の仮想ＯＳ１２０とは同一の応用プログラムと同一のＯＳとを有するものである。第１の仮想ＯＳ１１０と第２の仮想ＯＳ１２０との異なる点は、第１の仮想ＯＳ１１０は第１のＣＰＵ１４１の１個のＣＰＵのみにて処理されるのに対し、第２の仮想ＯＳ１２０は第２のＣＰＵ１４２および第３のＣＰＵ１４３の２個のＣＰＵにて処理される点である。

【００１０】

そして、第１の仮想ＯＳ１１０には、第１の仮想ＯＳ１１０にて動作する応用プログラムの第１のプログラムデータ（共有メモリ）１１１と、応用プログラムの第１のプログラムコード１１２と、第１の仮想ＯＳ１１０にて実行された応用プログラムのデータ、例えば、応用プログラム内のプログラムデータが変更した時間のデータなどがトレースされトレースデータとして保存されるトレースデータ保持部１１３とを備えている。第２の仮想ＯＳ１２０には、第１の仮想ＯＳ１１０と同様に、第２の仮想ＯＳ１２０にて動作する応用プログラムの第２のプログラムデータ（共有メモリ）１２１と、応用プログラムの第２

のプログラムコード１２２と、第２の仮想ＯＳ１２０にて実行された応用プログラムのデータがトレースされトレースデータとして保存されるトレースデータ保持部１２３とを備えている。

【００１１】

そして、第３の仮想ＯＳ１３０は、第１の仮想ＯＳ１１０の第１のプログラムデータ１１１および第２の仮想ＯＳ１２０の第２のプログラムデータ１２１をコピーするプログラムデータコピー部１３１と、プログラムデータコピー部１３１にコピーされた各プログラムデータを比較して差異を検出するプログラムデータ比較部１３２とを備えている。そして、第３の仮想ＯＳ１３０は第４のＣＰＵ１４４にて処理されている。

【００１２】

次に、上記のように構成された実施の形態１のデバッグ支援方法の動作について図２に基づいて説明する。説明に際して、まず、ＣＰＵの数による処理の差について説明する。第１の仮想ＯＳ１１０上では、図３に示すように、第１のＣＰＵ１４１のみが第１のプログラムデータ（共有メモリ）１１１への書き込みを行うため、排他制御を取る必要がなく正常に動作する。これに対し、第２の仮想ＯＳ１２０上では、図４に示すように、複数の第２のＣＰＵ１４２および第３のＣＰＵ１４３が第２のプログラムデータ（共有メモリ）１２１への書き込みを行うため、第２のＣＰＵが処理を行っている間に第３のＣＰＵが処理を行う可能性があり、第２のＣＰＵ１４２および第３のＣＰＵ１４３間の排他制御がとれず誤った動作（バグ）が発生する可能性がある。よって、ＣＰＵの数が複数になると誤動作が発生する可能性がある。このデバッグを支援するために以下の動作を行う。

【００１３】

まず、第３の仮想ＯＳ１３０を起動する（図２のステップＳＴ２０１）。このように第３の仮想ＯＳ１３０を最初に起動するのは、プログラムデータコピー部１３１が第１の仮想ＯＳ１１０および第２の仮想ＯＳ１２０の第１および第２のプログラムデータ（共有メモリ）１１１、１２１をコピーする準備を行うためである。次に、第１の仮想ＯＳ１１０、第２の仮想ＯＳ１２０を起動する（図２のステップＳＴ２０２）。次に、第１の仮想ＯＳ１１０、第２の仮想ＯＳ１２０上で動作させる応用プログラムの第１および第２のプログラムコード１１１、１２１をそれぞれ実行し、応用プログラムの第１および第２のプログラムデータを作成する（図２のステップＳＴ２０３）。

【００１４】

次に、第１および第２のプログラムコード１１１、１２１の動作中は、あらかじめ決められた第１のＯＳ１４１、第２のＯＳ１４２か第３のＯＳ１４３かで、第１および第２のプログラムコード１１１、１２１の特定の箇所を実行するごとに、該当応用プログラム部分の実行に関してデータをトレースして、トレースデータとして第１および第２のトレースデータ保持部１１３、１２３にそれぞれ記録する（図２のステップＳＴ２０４）。次に、第３の仮想ＯＳ１３０は第１の仮想ＯＳ１１０で動作する応用プログラムの第１のプログラムデータ１１１、第２の仮想ＯＳ１２０上で動作する応用プログラムの第２のプログラムデータ１２１をプログラムデータコピー部１３１に周期的にコピーする（図２のステップＳＴ２０５）。

【００１５】

次に、第３の仮想ＯＳはプログラムデータ比較部１３２を用いてプログラムデータコピー部１３１によってコピーした第１の仮想ＯＳ１１０、第２の仮想ＯＳ１２０の各プログラムデータを比較する（図２のステップＳＴ２０６）。次に、第１の仮想ＯＳ１１０、第２の仮想ＯＳ１２０のプログラムデータ間で差異を検出した場合は、その内容をユーザに通知する（図２のステップＳＴ２０７）。

【００１６】

上記のように構成された実施の形態１によれば、同一の応用プログラム、同一のＯＳを有する仮想環境をＣＰＵ数の違い以外に差異のない状況で動作させることができ、この動作を実行している瞬間のプログラムデータを別の仮想環境が監視することにより、その差異が発生した瞬間を容易に識別することができる。よって、これに基づいてデバッグを支

援することができる。また、第１の仮想ＯＳ、第２の仮想ＯＳ上で動作するトレースデータを保持しているため、その差異が検出された時点までトレースデータを比較することにより、ＣＰＵの数の差異によるＯＳ、応用プログラムの動作差異を容易に明らかにすることができる。このように、ＣＰＵが複数個になった場合のみに発生する障害の原因調査が容易となる。

【００１７】

尚、上記実施の形態１では第２の仮想ＯＳを２個のＣＰＵ上で動作させた場合のデバッグについて述べたが、図５に示すように第２の仮想ＯＳに第５のＣＰＵ３４４を追加して、３個のＣＰＵを用いて動作させた場合についても、また、それ以上の個数のＣＰＵを用いて動作させたとしても、上記実施の形態１と同様にデバッグの支援を行うことができる。

【００１８】

また、上記実施の形態１ではプログラムデータとして共有メモリを対象とする場合のデバッグについて述べたが、これに限られることはなく、例えば、図６に示すように第１および第２のプログラムデータ１１１、１２１としてメッセージキュを対象とすることもできる。この場合、上記実施の形態１と同様にデバッグすることができるのはもちろんのこと、メッセージキュは複数のメッセージを送出する機能を有しており、そのメッセージの内容と、メッセージを送出しているＣＰＵと、その送出時間とを特定できるため、ＣＰＵ間の排他制御が取れずメッセージの送出順番に誤りがあった場合でも、その動作に対するデバッグが可能になる。

【００１９】

実施の形態２．

図７はこの発明の実施の形態２におけるデバッグ支援方法に用いられるシステムの構成を示す図、図８は図７に示したデバッグ支援方法の動作を説明するためのフローチャートである。上記実施の形態１では応用プログラムが動作する仮想ＯＳがトレースデータをトレースし続ける場合について示したが、本実施の形態２においては、各仮想ＯＳ間の各プログラムデータに差異を検出した場合には、トレースデータの記録を停止し、容易にデバッグの支援を行うことができるものである。

【００２０】

図において、上記実施の形態１と同様の部分は同一符号を付して説明を省略する。第１の仮想ＯＳ１１０に配設されたトレース停止命令を受信するトレース停止命令受信部５１４と、第２の仮想ＯＳ１２０に配設されたトレース停止命令を受信するトレース停止命令受信部５２４と、第３の仮想ＯＳ１３０に配設されプログラムデータ比較部１３２にて第１の仮想ＯＳ１１０および第２の仮想ＯＳ１２０の各プログラムデータに差異があると検出されると、第１の仮想ＯＳ１１０の第１のトレース停止命令受信部５１４および第２の仮想ＯＳ１２０の第２のトレース停止命令受信部５２４にトレース停止命令を送信するトレース停止命令送信部５３３とを備える。

【００２１】

次に、上記のように構成された実施の形態２のデバッグ支援方法について図８に基づいて説明する。まず、上記実施の形態１と同様の動作を行い、第３の仮想ＯＳ１３０のプログラムデータ比較部１３２にてプログラムデータコピー部１３１によってコピーした第１の仮想ＯＳ１１０、第２の仮想ＯＳ１２０の各プログラムデータを比較して、第１の仮想ＯＳ１１０、第２の仮想ＯＳ１２０のプログラムデータ間で差異を検出した場合、第３の仮想ＯＳ１３０はトレース停止命令送信部５３３から第１の仮想ＯＳ１１０、第２の仮想ＯＳ１２０にトレース停止命令を送信する（図８のステップＳＴ６０６）。次に、第１の仮想ＯＳ１１０、第２の仮想ＯＳ１２０は第３の仮想ＯＳ１３０からのトレース停止命令を第１および第２のトレース停止命令受信部５１４、５２４で受信して、トレースを停止する（図８のステップＳＴ６０７）。

【００２２】

上記のように構成された実施の形態２によれば、上記実施の形態１と同様の効果を奏す

るのはもちろんのこと、第１の仮想ＯＳと第２の仮想ＯＳとの各プログラムデータ間で差異が検出された際に、第１の仮想ＯＳおよび第２の仮想ＯＳにおけるトレースデータの記録を停止するため、障害が発生したと想定される時点付近のトレースデータを、作業者が解析する際に容易に識別できる。

【００２３】

実施の形態３．

図９はこの発明の実施の形態３におけるデバッグ支援方法に用いられるシステムの構成を示す図、図１０は図９に示したデバッグ支援方法の動作を説明するためのフローチャートである。上記実施の形態１では応用プログラムが動作する仮想ＯＳがトレースデータをとり続ける場合について述べたが、本実施の形態３においては仮想ＯＳ間の各プログラムデータに差異が出た際に、仮想ＯＳの動作を停止してデバッガを起動するものである。

【００２４】

図において、上記実施の形態１と同様の部分は同一符号を付して説明を省略する。第１の仮想ＯＳ１１０に配設されたＯＳ停止命令を受信する第１のＯＳ停止命令受信部７１４と、第１の仮想ＯＳ１１０に配設された第１のデバッガ部７１５と、第２の仮想ＯＳ１３０に配設されたＯＳ停止命令を受信する第２のＯＳ停止命令受信部７２４と、第２の仮想ＯＳ１２０に配設された第２のデバッガ部７２５と、第３の仮想ＯＳ１３０に配設されプログラムデータ比較部１３２にて第１の仮想ＯＳ１１０および第２の仮想ＯＳ１２０の各プログラムデータに差異があると検出されると、第１の仮想ＯＳ１１０の第１のＯＳ停止命令受信部７１４および第２の仮想ＯＳ１２０の第２のＯＳ停止命令受信部７２４にＯＳ停止命令を送信するＯＳ停止命令送信部７３３とを備える。

【００２５】

次に、上記のように構成された実施の形態３のデバッグ支援方法について図１０に基づいて説明する。まず、上記実施の形態１と同様の動作を行い、第３の仮想ＯＳ１３０のプログラムデータ比較部１３２にてプログラムデータコピー部１３１によってコピーした第１の仮想ＯＳ１１０、第２の仮想ＯＳ１２０の各プログラムデータを比較して、第１の仮想ＯＳ１１０、第２の仮想ＯＳ１２０のプログラムデータ間で差異を検出した場合、第３の仮想ＯＳ１３０はＯＳ停止命令送信部７３３から第１の仮想ＯＳ１１０、第２の仮想ＯＳ１２０にＯＳ停止命令を送信する（図１０のステップＳＴ８０６）。

【００２６】

次に、第１の仮想ＯＳ１１０、第２の仮想ＯＳ１２０は第３の仮想ＯＳ１３０からのＯＳ停止命令を第１および第２のＯＳ停止命令受信部７１４、７２４で受信した後、それぞれ第１および第２のデバッガ部７１５、７２５の各デバッガを起動する（図１０のステップＳＴ８０７）。尚、第１の仮想ＯＳ１１０、第２の仮想ＯＳ１２０の停止命令が発生されると、自ずと、トレースデータの中断される。そして、このことにより、仮想ＯＳおよび応用プログラムのデータを容易に確認することができる。また、デバッガが終了し、動作が再開されると、再び中断された時点から上記動作が繰り返され、トレースも再開される。

【００２７】

上記のように構成された実施の形態３によれば、上記実施の形態１と同様の効果を奏するのはもちろんのこと、第１の仮想ＯＳと第２の仮想ＯＳとの各プログラムデータ間で差異が検出された際に、第１の仮想ＯＳおよび第２の仮想ＯＳにおいてデバッガすることができる。そして、このデバッガを起動することで障害が発生したと想定される時点における仮想ＯＳならびに応用プログラムのデータを確認することができる。

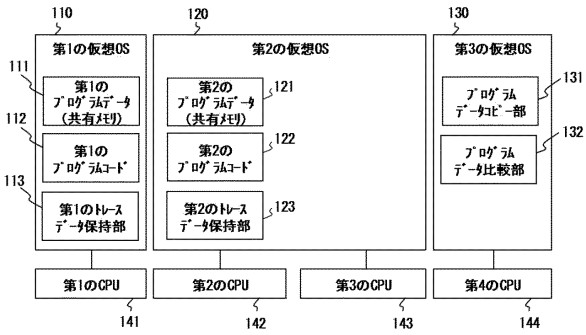
【符号の説明】

【００２８】

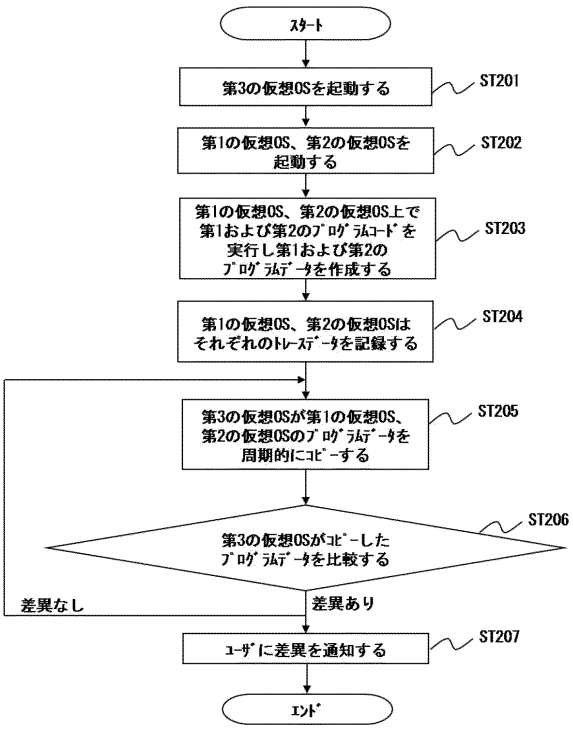
１１０　第１の仮想ＯＳ、１１１　第１のプログラムデータ、
１１２　第１のプログラムコード部、１１３　第１のトレースデータ保持部、
１２０　第２の仮想ＯＳ、１２１　第２のプログラムデータ、
１３２　第２のプログラムコード部、１２３　第２のトレースデータ保持部、

１３０　第３の仮想ＯＳ、１３１　プログラムデータコピー部、
１３２　プログラムデータ比較部、１４１　第１のＣＰＵ、１４２　第２のＣＰＵ、
１４３　第３のＣＰＵ、１４４　第４のＣＰＵ、３４４　第５のＣＰＵ、
５１４　第１のトレース停止命令受信部、５２４　第２のトレース停止命令受信部、
５３３　トレース停止命令送信部、７１４　第１のＯＳ停止命令受信部、
７１５　第１のデバッガ部、７２４　第２のＯＳ停止命令受信部、
７２５　第２のデバッガ部、７３３　ＯＳ停止命令送信部。

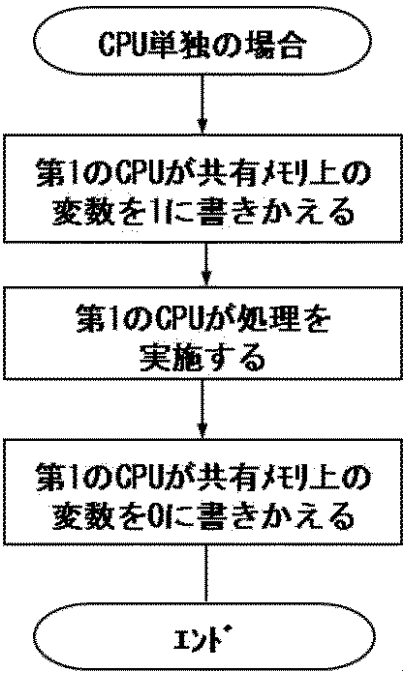
【図１】



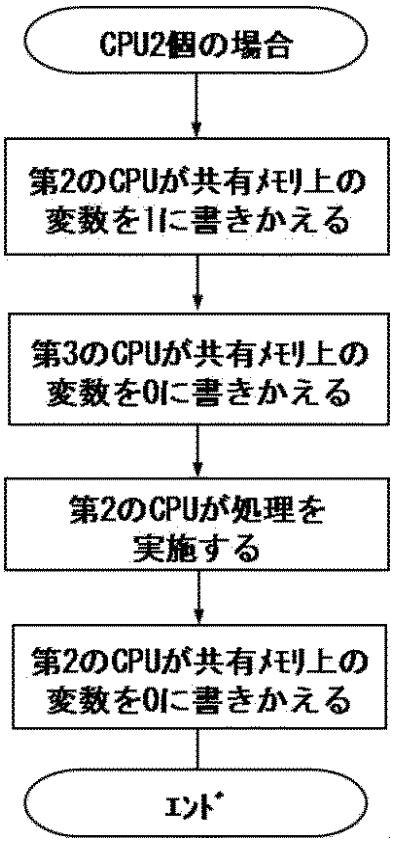
【図２】



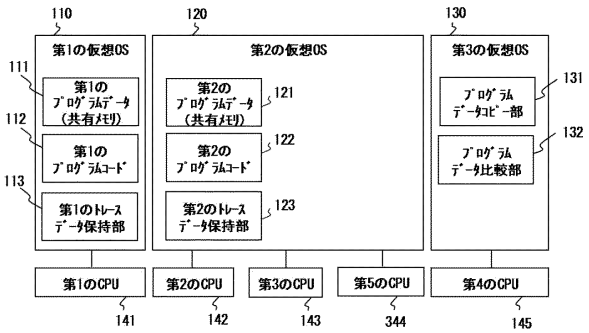
【図３】


CPU単独の場合
第1のCPUが共有メモリ上の変数を1に書きかえる
第1のCPUが処理を実施する
第1のCPUが共有メモリ上の変数を0に書きかえる
エンド

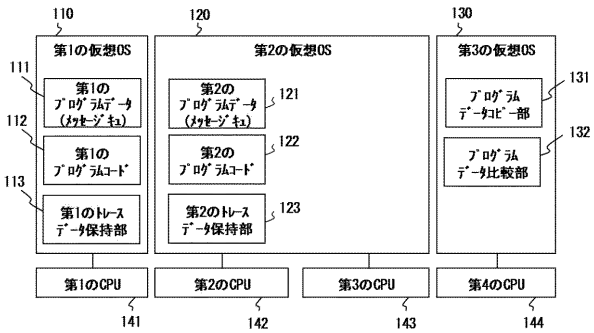
【図４】


CPU2個の場合
第2のCPUが共有メモリ上の変数を1に書きかえる
第3のCPUが共有メモリ上の変数を0に書きかえる
第2のCPUが処理を実施する
第2のCPUが共有メモリ上の変数を0に書きかえる
エンド

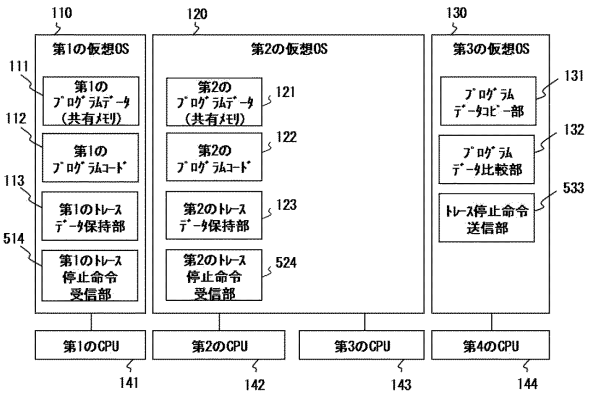
【図５】


110
120
130
第1の仮想OS
第2の仮想OS
第3の仮想OS
111
第1のプログラムデータ（共有メモリ）
112
第1のプログラムコード
113
第1のトレースデータ保持部
121
第2のプログラムデータ（共有メモリ）
122
第2のプログラムコード
123
第2のトレースデータ保持部
131
プログラムデータコピー部
132
プログラムデータ比較部
第1のCPU
第2のCPU
第3のCPU
第5のCPU
第4のCPU
141
142
143
344
145

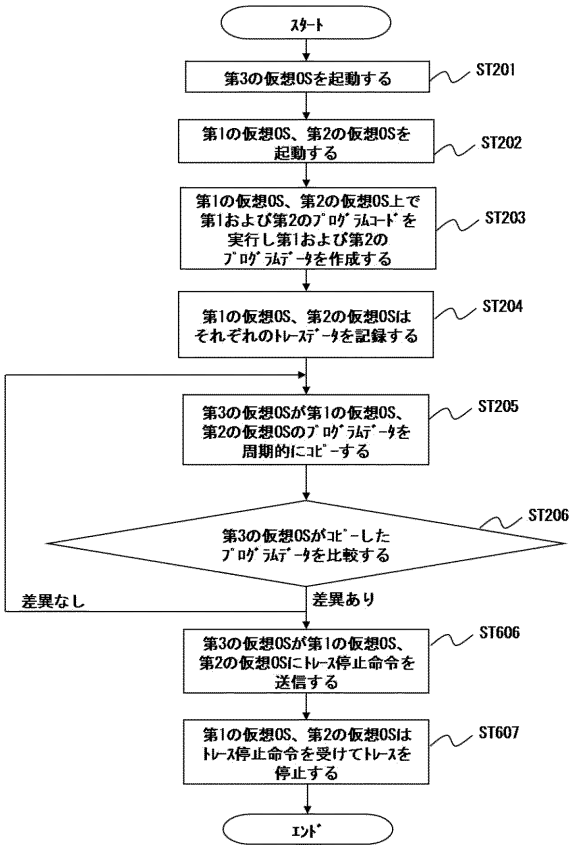
【図６】


110
120
130
第1の仮想OS
第2の仮想OS
第3の仮想OS
111
第1のプログラムデータ（メッセージキュー）
112
第1のプログラムコード
113
第1のトレースデータ保持部
121
第2のプログラムデータ（メッセージキュー）
122
第2のプログラムコード
123
第2のトレースデータ保持部
131
プログラムデータコピー部
132
プログラムデータ比較部
第1のCPU
第2のCPU
第3のCPU
第4のCPU
141
142
143
144

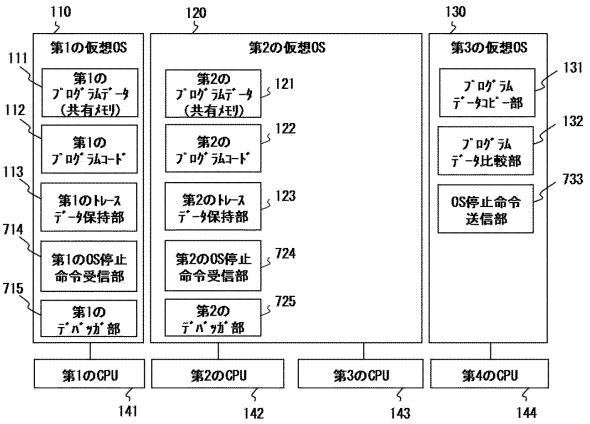
【図７】


110
120
130
第1の仮想OS
第2の仮想OS
第3の仮想OS
111
第1のプログラムデータ（共有メモリ）
112
第1のプログラムコード
113
第1のトレースデータ保持部
514
第1のトレース停止命令受信部
121
第2のプログラムデータ（共有メモリ）
122
第2のプログラムコード
123
第2のトレースデータ保持部
524
第2のトレース停止命令受信部
131
プログラムデータコピー部
132
プログラムデータ比較部
533
トレース停止命令送信部
第1のCPU
第2のCPU
第3のCPU
第4のCPU
141
142
143
144

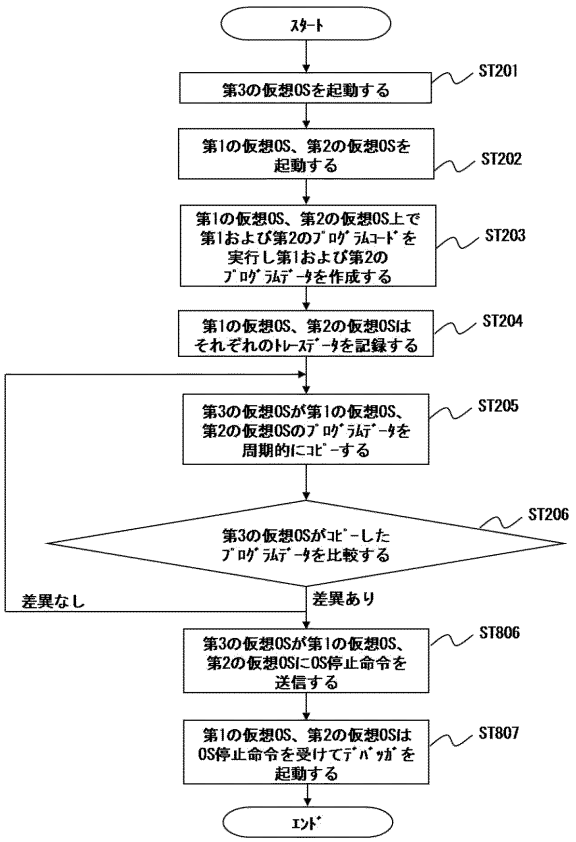
【図８】


ｽﾀｰﾄ
第3の仮想OSを起動する
ST201
第1の仮想OS、第2の仮想OSを起動する
ST202
第1の仮想OS、第2の仮想OS上で第1および第2のﾌﾟﾛｸﾞﾗﾑｺｰﾄﾞを実行し第1および第2のﾌﾟﾛｸﾞﾗﾑﾃﾞｰﾀを作成する
ST203
第1の仮想OS、第2の仮想OSはそれぞれのﾄﾚｰｽﾃﾞｰﾀを記録する
ST204
第3の仮想OSが第1の仮想OS、第2の仮想OSのﾌﾟﾛｸﾞﾗﾑﾃﾞｰﾀを周期的にｺﾋﾟｰする
ST205
第3の仮想OSがｺﾋﾟｰしたﾌﾟﾛｸﾞﾗﾑﾃﾞｰﾀを比較する
ST206
差異なし
差異あり
第3の仮想OSが第1の仮想OS、第2の仮想OSにﾄﾚｰｽ停止命令を送信する
ST606
第1の仮想OS、第2の仮想OSはﾄﾚｰｽ停止命令を受けてﾄﾚｰｽを停止する
ST607
ｴﾝﾄﾞ

【図９】


110
120
130
第1の仮想OS
第2の仮想OS
第3の仮想OS
111
第1のﾌﾟﾛｸﾞﾗﾑﾃﾞｰﾀ（共有ﾒﾓﾘ）
第2のﾌﾟﾛｸﾞﾗﾑﾃﾞｰﾀ（共有ﾒﾓﾘ）
121
131
ﾌﾟﾛｸﾞﾗﾑﾃﾞｰﾀｺﾋﾟｰ部
112
第1のﾌﾟﾛｸﾞﾗﾑｺｰﾄﾞ
第2のﾌﾟﾛｸﾞﾗﾑｺｰﾄﾞ
122
132
ﾌﾟﾛｸﾞﾗﾑﾃﾞｰﾀ比較部
113
第1のﾄﾚｰｽﾃﾞｰﾀ保持部
第2のﾄﾚｰｽﾃﾞｰﾀ保持部
123
733
OS停止命令送信部
714
第1のOS停止命令受信部
第2のOS停止命令受信部
724
715
第1のﾃﾞﾊﾞｯｶﾞ部
第2のﾃﾞﾊﾞｯｶﾞ部
725
第1のCPU
第2のCPU
第3のCPU
第4のCPU
141
142
143
144

【図１０】


ｽﾀｰﾄ
第3の仮想OSを起動する
ST201
第1の仮想OS、第2の仮想OSを起動する
ST202
第1の仮想OS、第2の仮想OS上で第1および第2のﾌﾟﾛｸﾞﾗﾑｺｰﾄﾞを実行し第1および第2のﾌﾟﾛｸﾞﾗﾑﾃﾞｰﾀを作成する
ST203
第1の仮想OS、第2の仮想OSはそれぞれのﾄﾚｰｽﾃﾞｰﾀを記録する
ST204
第3の仮想OSが第1の仮想OS、第2の仮想OSのﾌﾟﾛｸﾞﾗﾑﾃﾞｰﾀを周期的にｺﾋﾟｰする
ST205
第3の仮想OSがｺﾋﾟｰしたﾌﾟﾛｸﾞﾗﾑﾃﾞｰﾀを比較する
ST206
差異なし
差異あり
第3の仮想OSが第1の仮想OS、第2の仮想OSにOS停止命令を送信する
ST806
第1の仮想OS、第2の仮想OSはOS停止命令を受けてﾃﾞﾊﾞｯｶﾞを起動する
ST807
ｴﾝﾄﾞ

フロントページの続き

審査官　多賀　実

(56)参考文献　特開２００４－３５５２３３（ＪＰ，Ａ）
特開平０７－２７１６３１（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　９／４６
Ｇ０６Ｆ１１／１６－１１／２０
Ｇ０６Ｆ１１／２８－１１／３６