

(19) 日本国特許庁 (JP)

(12) 特 許 公 報 (B2)

(11) 特許番号

特許第6079317号  
(P6079317)

(45) 発行日 平成29年2月15日 (2017.2.15)

(24) 登録日 平成29年1月27日 (2017.1.27)

(51) Int.Cl.

F I

G O 6 F 11/34 (2006.01)

G O 6 F 11/34 1 7 1

G O 6 F 11/30 (2006.01)

G O 6 F 11/30 1 4 O C

請求項の数 5 (全 21 頁)

(21) 出願番号 特願2013-43158 (P2013-43158)  
 (22) 出願日 平成25年3月5日 (2013.3.5)  
 (65) 公開番号 特開2014-170482 (P2014-170482A)  
 (43) 公開日 平成26年9月18日 (2014.9.18)  
 審査請求日 平成27年10月7日 (2015.10.7)

(73) 特許権者 000005223  
 富士通株式会社  
 神奈川県川崎市中原区上小田中4丁目1番  
 1号  
 (74) 代理人 100107766  
 弁理士 伊東 忠重  
 (74) 代理人 100070150  
 弁理士 伊東 忠彦  
 (74) 代理人 100146776  
 弁理士 山口 昭則  
 (72) 発明者 田代 智彦  
 神奈川県川崎市中原区上小田中4丁目1番  
 1号 富士通株式会社内

最終頁に続く

(54) 【発明の名称】 仮想計算機システム及びその管理方法並びに仮想計算機システムの管理プログラム

(57) 【特許請求の範囲】

【請求項 1】

ハイパーバイザー上で動作する複数の仮想マシンを有する仮想計算機システムであって、

前記ハイパーバイザーにおいて、割り込み発生時に動作していた、前記複数の仮想マシンの1つである第1の仮想マシン上のプロセスの、動作状態と、プロセスが前記複数の仮想マシンそれぞれで動作する際に設定されるアドレス領域情報と、を含むトレース情報を、前記第1の仮想マシンに対応するコンテキスト退避領域から採取するトレース情報採取部と、

前記複数の仮想マシンそれぞれにおいて、前記複数の仮想マシンそれぞれで動作するプロセスの識別番号と、前記アドレス領域情報と、を含むシンボルマップ情報を前記複数の仮想マシンそれぞれが有する管理領域からそれぞれ採取し、前記ハイパーバイザーに送信する、複数のシンボルマップ情報採取部と、

前記ハイパーバイザーにおいて、前記トレース情報と前記シンボルマップ情報をプロセス毎に関連付ける解析処理部と、  
 を含むことを特徴とする仮想計算機システム。

【請求項 2】

請求項 1 記載の仮想計算機システムであって、

前記解析処理部は、それぞれの前記アドレス領域情報に基づき、前記トレース情報と前記シンボルマップ情報をプロセス単位で関連付けることを特徴とする仮想計算機システム

10

20

。

【請求項 3】

請求項 2 記載の仮想計算機システムであって、

前記トレース情報は、前記第 1 の仮想マシンの識別番号と、前記割り込み発生時に動作していたプロセスに割り当てられていた前記アドレス領域情報及び実行アドレスと、を含み、

前記シンボルマップ情報は、前記複数の仮想マシンそれぞれで動作するプロセスの識別番号と、前記アドレス領域情報と、当該プロセスで実行される関数の関数名およびアドレスと、を含み、

前記解析処理部は、前記トレース情報及び前記シンボルマップ情報の前記アドレス領域情報を関連付けると共に、前記トレース情報の前記実行アドレスと前記シンボルマップ情報の前記関数の関数名及びアドレスを関連付けることを特徴とする仮想計算機システム。

10

【請求項 4】

物理 CPU 毎に割り込みの設定が可能なハイパーバイザー上で動作する複数の仮想マシンを有する仮想計算機システムにおいて、定期的に CPU 割り込みを発生させ、

割り込みにより前記ハイパーバイザー上のコンテキスト退避領域に退避された、前記複数の仮想マシンの 1 つである第 1 の仮想マシン上のプロセスの、動作状態と、プロセスが前記複数の仮想マシンそれぞれで動作する際に設定されるアドレス領域情報と、を含むトレース情報を採取し、

前記複数の仮想マシンそれぞれにおいて、前記複数の仮想マシンそれぞれで動作するプロセスの識別番号と、前記アドレス領域情報と、を含むシンボルマップ情報を前記複数の仮想マシンがそれぞれ有する管理領域から採取し、

20

前記トレース情報と前記シンボルマップ情報をプロセス毎に関連付ける、処理を有することを特徴とする仮想計算機システムの管理方法。

【請求項 5】

物理 CPU 毎に割り込みの設定が可能なハイパーバイザー上で動作する複数の仮想マシンを有する仮想計算機システムにおいて、定期的に CPU 割り込みを発生させる処理と、

割り込みにより前記ハイパーバイザー上のコンテキスト退避領域に退避された、前記複数の仮想マシンの 1 つである第 1 の仮想マシン上のプロセスの、動作状態と、プロセスが前記複数の仮想マシンそれぞれで動作する際に設定されるアドレス領域情報と、を含むトレース情報を採取する処理と、

30

前記複数の仮想マシンそれぞれにおいて、前記複数の仮想マシンそれぞれで動作するプロセスの識別番号と、前記アドレス領域情報と、を含むシンボルマップ情報を前記複数の仮想マシンがそれぞれ有する管理領域から採取する処理と、

前記トレース情報と前記シンボルマップ情報をプロセス毎に関連付ける処理と、をコンピュータに実行させることを特徴とする仮想計算機システムの管理プログラム。

【発明の詳細な説明】

【技術分野】

【0001】

本発明は、仮想計算機システム及びその管理方法並びに仮想計算機システムの管理プログラムに関する。

40

【背景技術】

【0002】

近年、サーバ装置を仮想環境化することによって複数の仮想マシン (VM: Virtual Machine) を動作させてユーザにサービスを提供するシステムが開発されている。このようなサービスでは、1 つのサーバ装置で 1 つ以上の仮想マシンを動作させ、仮想マシン毎に独立した OS (Operating System) を走らせてアプリケーション・プログラムを動作させる。

【0003】

50

サーバ装置を仮想環境化する方法としては、ハイパーバイザーと呼ばれるソフトウェアを使用して複数の異なるOSを並列に実行させるものがある。この場合、ハイパーバイザーが、仮想マシンの制御を行うと共に、仮想マシンに割り当てたCPU (Central Processing Unit) のリソース量を管理する。

【0004】

仮想環境下でサーバ装置を運用するときには、複数の仮想マシンからシステムの負荷を算出し、仮想化によるオーバーヘッドを監視する必要がある。オーバーヘッドは、システムやプロトコルの処理の負荷、サーバ装置やネットワーク機器などのハードウェアの処理能力や、OSなどのシステムソフトウェアの性能など、個々の性能によってシステム全体にかかる負荷であり、本来の処理の他にサーバ装置に余分にかかる負荷である。オーバーヘッドが生じると、例えば処理のスループットが低下したり、ユーザの操作に対する応答速度が低下したりする。

10

【0005】

そこで、仮想環境を提供するサーバ装置では、オーバーヘッドを監視する仕組みが必要になる。このとき、サーバ装置内での処理の流れを可視化できると、オーバーヘッドの監視が容易になる。

【0006】

ここで、サーバ装置内での処理の流れを可視化するためには、ハイパーバイザーや各仮想マシンで処理中の命令を特定する必要がある。例えば、1つのOS下で複数のアプリケーション・プログラムを実行するときに、どのアプリケーション・プログラムの処理が実行中なのかを特定する必要がある。この場合に用いられる従来の方法としては、各アプリケーション・プログラムで実行する命令と、命令のアドレスを時系列にまとめたトレース情報を取得するものがある。このとき、トレース情報には、命令を処理中の空間ID (Identification) を付加する。さらに、空間ID毎にプログラム名と先頭アドレスとを有するプログラムマップを作成する。そして、トレース情報とプログラムマップ情報を空間IDによって対応付けて、命令を実行しているプログラム名を特定する。これによって、アプリケーション・プログラム毎の処理と実行時間とを調べることが可能になる。

20

【先行技術文献】

【特許文献】

【0007】

30

【特許文献1】特開昭63-201840

【発明の概要】

【発明が解決しようとする課題】

【0008】

ここで、仮想計算機システムにおいて複数の仮想マシンの動作に関する分析や解析を行う場合には、それぞれの仮想マシンやハイパーバイザーで処理されるプロセス等の実行時間を採取し、処理時間などを評価する必要がある。しかしながら、仮想マシンにおける処理は、ハイパーバイザーや他の仮想マシンに実行制御権が移されることにより、実際には物理CPU上で動作できない時間が存在する。この場合、仮想マシンで取得される時間が実際の時間よりも遅れ、仮想マシンで処理されるプロセス等の実行時間を正確に採取できない。仮想マシン上のプロセスの実行時間を正確に把握することは、ハイパーバイザーが仮想マシンに仮想割り込み処理を実行し、仮想マシン内のプログラムの情報を採取し、ハイパーバイザーの時間軸で一元的に各仮想マシン上のプロセスの実行時間を採取することで実現可能である。しかしながら、仮想割り込み処理という余計な処理が発生することになるので、実行処理が煩雑になると共に、処理コストが高くなる。

40

【0009】

また、ハイパーバイザーは、サーバ装置に提供される環境下で処理を実行するので、ハイパーバイザー内の処理や、仮想マシンに出力した命令を特定することは容易である。ところが、各仮想マシン上で動くアプリケーション・プログラムはユーザ毎に異なる。そして、セキュリティ上の問題から仮想マシンにおける処理の詳細をハイパーバイザー側から

50

特定できない場合がある。このために、従来の仮想計算機システムでは、システム管理者が仮想マシンにおける処理の内容や、処理の流れを把握することは容易ではなかった。

この発明は、このような事情に鑑みてなされたものであり、サーバ装置内での処理の流れを可視化してオーバーヘッドの監視を容易にすることを目的とする。

【課題を解決するための手段】

【0010】

実施形態の一観点によれば、ハイパーバイザー上で動作する複数の仮想マシンを有する仮想計算機システムであって、前記ハイパーバイザーにおいて、割り込み発生時に動作していた、前記複数の仮想マシンの1つである第1の仮想マシン上のプロセスの、動作状態と、プロセスが前記複数の仮想マシンそれぞれで動作する際に設定されるアドレス領域情報と、を含むトレース情報を、前記第1の仮想マシンに対応するコンテキスト退避領域から採取するトレース情報採取部と、前記複数の仮想マシンそれぞれにおいて、前記複数の仮想マシンそれぞれで動作するプロセスの識別番号と、前記アドレス領域情報と、を含むシンボルマップ情報を前記複数の仮想マシンそれぞれが有する管理領域からそれぞれ採取し、前記ハイパーバイザーに送信する、複数のシンボルマップ情報採取部と、前記ハイパーバイザーにおいて、前記トレース情報と前記シンボルマップ情報をプロセス毎に関連付ける解析処理部と、を含むことを特徴とする仮想計算機システムが提供される。

10

【0011】

また、実施形態の別の観点によれば、物理CPU毎に割り込みの設定が可能なハイパーバイザー上で動作する複数の仮想マシンを有する仮想計算機システムにおいて、定期的にCPU割り込みを発生させ、割り込みにより前記ハイパーバイザー上のコンテキスト退避領域に退避された、前記複数の仮想マシンの1つである第1の仮想マシン上のプロセスの、動作状態と、プロセスが前記複数の仮想マシンそれぞれで動作する際に設定されるアドレス領域情報と、を含むトレース情報を採取し、前記複数の仮想マシンそれぞれにおいて、前記複数の仮想マシンそれぞれで動作するプロセスの識別番号と、前記アドレス領域情報と、を含むシンボルマップ情報を前記複数の仮想マシンがそれぞれ有する管理領域から採取し、前記トレース情報と前記シンボルマップ情報をプロセス毎に関連付ける、処理を有することを特徴とする仮想計算機システムの管理方法が提供される。

20

【0012】

さらに、実施形態の別の観点によれば、物理CPU毎に割り込みの設定が可能なハイパーバイザー上で動作する複数の仮想マシンを有する仮想計算機システムにおいて、定期的にCPU割り込みを発生させる処理と、割り込みにより前記ハイパーバイザー上のコンテキスト退避領域に退避された、前記複数の仮想マシンの1つである第1の仮想マシン上のプロセスの、動作状態と、プロセスが前記複数の仮想マシンそれぞれで動作する際に設定されるアドレス領域情報と、を含むトレース情報を採取する処理と、前記複数の仮想マシンそれぞれにおいて、前記複数の仮想マシンそれぞれで動作するプロセスの識別番号と、前記アドレス領域情報と、を含むシンボルマップ情報を前記複数の仮想マシンがそれぞれ有する管理領域から採取する処理と、前記トレース情報と前記シンボルマップ情報をプロセス毎に関連付ける処理と、をコンピュータに実行させることを特徴とする仮想計算機システムの管理プログラムが提供される。

30

40

【発明の効果】

【0013】

ハイパーバイザーで取得した情報に基づいて仮想マシン上の処理を関数レベルで解析することが可能になるので、処理の流れを容易に把握できるようになり、メンテナンス性の向上や性能最適化が可能となる。またそれによりさらなるリソース使用の効率化も期待される。

【図面の簡単な説明】

【0014】

【図1】図1は、本発明の実施の形態に係る仮想計算機システムのハードウェア構成の一例を示すブロック図である。

50

【図 2 A】図 2 A は、本発明の実施の形態に係る仮想計算機システムのソフトウェア構成の一例を示すブロック図である。

【図 2 B】図 2 B は、本発明の実施の形態に係る仮想計算機システムの機能ブロック図である。

【図 3】図 3 は、本発明の実施の形態に係る仮想計算機システムの仮想制御データ構造体の一例を示す図である。

【図 4 A】図 4 A は、本発明の実施の形態に係る仮想計算機システムにおけるトレース情報の一例を模式的に示す図である。

【図 4 B】図 4 B は、本発明の実施の形態に係る仮想計算機システムにおけるトレース情報の具体例を示す図である。

10

【図 5】図 4 B は、本発明の実施の形態に係る仮想計算機システムにおける処理の一例を模式的に示す図である。

【図 6 A】図 6 A は、本発明の実施の形態に係る仮想計算機システムにおけるシンボルマップ情報の一例を模式的に示す図である。

【図 6 B】図 6 B は、本発明の実施の形態に係る仮想計算機システムにおけるシンボルマップ情報の具体例を示す図である。

【図 7 A】図 7 A は、本発明の実施の形態に係る仮想計算機システムにおける解析データの一例を模式的に示す図である。

【図 7 B】図 7 B は、本発明の実施の形態に係る仮想計算機システムにおける解析データの作成手順を模式的に示す図である。

20

【図 8 A】図 8 A は、本発明の実施の形態に係る仮想計算機システムの処理の一例を示すフローチャートである（その 1）。

【図 8 B】図 8 B は、本発明の実施の形態に係る仮想計算機システムの処理の一例を示すフローチャートである（その 2）。

【図 8 C】図 8 C は、本発明の実施の形態に係る仮想計算機システムの処理の一例を示すフローチャートである（その 3）。

【図 9】図 9 は、本発明の第 2 の実施の形態に係る仮想計算機システムの機能ブロック図である。

【発明を実施するための形態】

【0015】

30

発明の目的及び利点は、請求の範囲に具体的に記載された構成要素及び組み合わせによって実現され達成される。

前述の一般的な説明及び以下の詳細な説明は、典型例及び説明のためのものであって、本発明を限定するためのものではない。

【0016】

（第 1 の実施の形態）

第 1 の実施の形態について図面を参照して詳細に説明する。

図 1 に実施形態の仮想計算機システムとして機能するサーバ装置の概略構成を示す。

サーバ装置 1 は、複数の CPU 2 と、メモリ 3 と、記録装置 4 と、入力装置 5 と、出力装置 6 と、外部記録装置 7 と、通信制御装置 8 がバス接続されることでハードウェアが構築されている。そして、ハードウェアに含まれる CPU 2 が、メモリ 3 上に展開された所定のプログラムを実行することにより、仮想環境を提供するハイパーバイザー（仮想化レイヤ）が実行される。ここで、ハイパーバイザーは、OS 上でアプリケーションソフトとして動作し、ゲスト OS が動作できるようにする仮想化ソフトウェアであり、仮想化モニタ、又は仮想化 OS と呼ばれることもある。

40

【0017】

CPU 2 には、メモリ 3 に記録装置 4 からプログラムやデータを読み出して実行することによって、各種の機能を実現する。この実施の形態において、CPU 2 は 3 つ搭載されているが、1 つ又は 2 つでも良いし、4 つ以上でも良い。例えば、図 1 に示す 3 つの CPU 2 は、それぞれに CPU 番号が付与されており、いずれの CPU 2 の処理であるのかを

50

特定できるようになっている。

【0018】

また、CPU 2 には、ハードウェア関連のイベントのカウントを格納する性能カウンタが実装されている。性能カウンタは、一般的なシステム状況のチェック、及びパフォーマンス基準との比較判断に用いられるもので、1 秒当たりのカウント数を取得できる。このために、性能カウンタの機能を用いることによって、サーバ装置 1 における処理を 1 m 秒 ~ 1  $\mu$  秒の間隔で測定することが可能になる。この実施の形態では、CPU 2 の性能カウンタによるサンプリングプロファイラ方式で時間計測をすることによって、ソフトウェア的なオーバーヘッドを発生させず、かつソフトウェアの負荷に依存しない正確なデータの所得が可能になっている。

10

【0019】

メモリ 3 は、ROM (Read Only Memory) や RAM (Random Access Memory) などを有し、データやアプリケーション・プログラムが一時的に格納される。記録装置 4 は、サーバ装置 1 のオペレーティング・システム (OS) や各種のアプリケーション・プログラムが保存される。入力装置 5 は、キーボードやマウスなど、システム管理者がデータを入力するために使用する機器である。入力装置 5 は、タッチパネル式のディスプレイでも良い。出力装置 6 は、ディスプレイやプリンタなどがある。外部記録装置 7 は、不図示の外部記録媒体にデータを読み書きするための装置である。外部記録媒体に記憶させているアプリケーションやデータを外部記録装置 7 で読み出すことによって、記録装置 4 にインストールしたり、CPU 2 に展開させて実行させたりすることが可能になる。

20

【0020】

通信制御装置 8 は、ネットワークを介して他の端末装置との間の通信を制御する。また、ネットワークを介してアプリケーション・プログラムをダウンロードして記録装置 4 にインストールすることも可能である。ネットワークは、専用回線を用いて構築しても良いし、公衆の回線を利用しても良い。

【0021】

ここで、通信制御装置 8 や外部記録装置 7 は、サーバ装置 1 にデータを入力するための入力装置としても機能する。また、入力装置 5、出力装置 6、外部記録装置 7 は、実施形態に必須の構成要素ではない。また、サーバ装置 1 は、図 1 に示す構成要素以外の装置を含んでも良い。

30

【0022】

続いて、図 2 A を参照して、このサーバ装置 1 において仮想環境を構築するために用いられるハイパーバイザについて説明する。図 2 A には、一例として、アプリケーションタイプの仮想環境のソフトウェア構成を模式的に示している。

ハイパーバイザ 2 1 は、サーバ装置 1 のハードウェア 2 0 によって実現される仮想計算機システムの全体を制御するもので、システム管理者によって提供されるアプリケーション・プログラムである。例えば、ハイパーバイザ 2 1 は、第 1 の仮想マシン (VM ; Virtual Machine) 2 2 と、第 2 の仮想マシン 2 3 とを提供する。仮想マシンの数は、1 つでも良いし、3 つ以上でも良い。ここで、仮想マシン 2 2、2 3 は、ゲストと呼ばれることもある。

40

【0023】

ハイパーバイザ 2 1 上で稼動する複数の仮想マシン 2 2、2 3 は、それぞれの OS (ゲスト OS) 2 5、2 7 によって動作する。例えば、第 1 の仮想マシン 2 2 は、第 1 のゲスト OS 2 5 上で 1 つ以上のアプリケーション・プログラム 2 6 A、2 6 B、2 6 C の処理を実現する。同様に、第 2 の仮想マシン 2 3 は、第 2 のゲスト OS 2 7 上で 1 つ以上のアプリケーション・プログラム 2 8 A、2 8 B、2 8 C の処理を実現する。ここで、アプリケーション・プログラム 2 6 A ~ 2 6 C、2 8 A ~ 2 8 C は、仮想マシン 2 2、2 3 を使用するユーザが容易するもので、例えばデータベースソフトや、仮想マシンで実行されるプロセスを監視する監視プログラムなどがある。

【0024】

50

ハイパーバイザー 21 は、各ゲスト OS 25, 27 のディスパッチ処理、各ゲスト OS 25, 27 が実行する特権命令のエミュレート、ハードウェア 20 の制御などを行う。ハイパーバイザー型の仮想化アプリケーションでは、ハイパーバイザー 21 が直接ハードウェア 20 の制御を実行するために、処理の応答性に優れる。

【0025】

また、図 2B を参照して仮想計算機システムの機能について説明する。仮想計算機システムを実現するサーバ装置 1 は、トレース情報採取部 11 と、採取開始指示部 12 と、シンボルマップ情報採取部 13 と、データ送信部 14 と、データ受信部 15 と、解析処理部 16 と、プロファイル処理部 17 とに機能分割できる。

トレース情報採取部 11 は、トレース情報を採取する。トレース情報は、サンプリング割り込み処理発生時に動作していた仮想マシン 22, 23 上の処理を特定する仮想マシン番号と、その仮想マシン 22, 23 の実行アドレス値及びページテーブルアドレス値と、プロセスの実行時間とを含む。実行アドレス値及びページテーブルアドレス値は、コンテキスト退避領域から採取される。

【0026】

シンボルマップ情報採取部 13 は、シンボルマップ情報を採取する。シンボルマップ情報は、各仮想マシン 22, 23 上の各プロセスを特定する情報と、プロセスで実行される関数を特定する情報が含まれる。

データ送信部 14 は、シンボルマップ情報をデータ受信部 15 に送信する。

データ受信部 15 は、シンボルマップ情報を受け取り、対応する仮想マシン 22, 23 の識別情報を付与して保存する。

【0027】

解析処理部 16 は、トレース情報とシンボルマップ情報を対応付け、トレース情報のページテーブルアドレス値をプロセスの識別情報に変換し、解析データを作成する。解析データには、仮想マシン 22, 23 を識別する情報と、仮想マシン 22, 23 のプロセスを識別する情報と、関数を特定する情報と、プロセスの実行時間の情報が含まれる。

プロファイル処理部 17 は、第 3 のプロセス情報を使用して既存のプロファイル処理を実行する。既存のプロファイル処理としては、プロセス毎や、関数毎に処理の実行に要する時間を把握するためのデータ処理が含まれる。

【0028】

ここで、この実施の形態のサーバ装置 1 は、例えば、CPU 2 にインテル社製のプロセッサを用いており、CPU 2 に VMM (Virtual Machine Monitor) と呼ばれるプログラムを走らせ、その上で仮想マシン 22, 23 を動作させる。このような仮想計算機システムにおいては、例えば、各仮想マシン 22, 23 の実行スケジューリングや、仮想マシン 22, 23 又はハイパーバイザー 21 における処理の過程で、複数の仮想マシン 22, 23 やハイパーバイザー 21 が同じ物理リソースにアクセスすることがある。この場合には、その物理リソースに割り当てられた内容をコンテキスト退避領域に退避させたり、復元させたりすることによって各仮想マシン 22, 23 における処理の連続性を担保している。

【0029】

このために、CPU 2 は、コンテキスト退避領域と、これを制御するためのコントロールレジスタとを有する。例えば、インテル社製のプロセッサにおいては CR0 から CR4 までのコントロールレジスタを有する。そのうちの CR3 レジスタは、ページ・ディレクトリの物理アドレスを指し、ページテーブルアドレスと呼ばれるアドレス領域情報である。実行するユーザープロセスを切り替える毎に、CR3 レジスタが書き換えられることによって、そのプロセス用の拡張ページテーブルが指定される。そして、割り込み処理発生時には、動作中の仮想マシン 22, 23 の CR3 レジスタ値をコンテキスト退避領域に退避させる。そして、割り込み処理が終了して、元の仮想マシン 22, 23 の処理を再開するときには、コンテキスト退避領域の CR3 レジスタ値を再度読み込んで、プロセスを再開する。従って、CR3 レジスタを参照することによって、プロセスを区別することがで

10

20

30

40

50

きる。

【0030】

さらに、CPU2のレジスタには、命令ポインタであるプログラムカウンタ(EIP)を有する。プログラムカウンタは、次に実行すべき命令を記憶したメモリ3上のアドレスを一時的に格納する。メモリ3から命令が読み出されると、その分だけプログラムカウンタの値を増やし、次に実行すべき命令が保存されている番地を指し示すようになっている。このために、プログラムカウンタを参照することによって、命令のアドレスを取得することができる。

【0031】

また、CPU2には、拡張ページテーブル(EPT)と呼ぶVMMの制御によるページテーブル構造が実装されている。拡張ページテーブルは、仮想マシン22, 23中の物理アドレスから、VMMの物理アドレス空間(最終的な物理アドレス)への変換をサポートすることで、VMMのページテーブル管理のオーバーヘッドを低減するために設けられている。

【0032】

この実施の形態におけるCPU2は、ルートモード(カーネルモード)でハイパーバイザー21による処理を実行し、ノンルートモード(ユーザモード)に遷移すると仮想マシン22, 23の処理を実行する。例えば、ルートモードからノンルートモードへの遷移は「VM-entry」と言われ、その逆を「VM-exit」と呼ばれる。この場合、ハイパーバイザー21は、「VM-exit命令」によって仮想マシン22, 23に制御を渡す。

【0033】

例えば、VMMが仮想マシン22, 23を起動すると、CPU2の動作モードはルートモードからノンルートモードに遷移し、仮想マシン22, 23が終了するとノンルートモードからルートモードに遷移する。仮想マシン22, 23内で所定の命令を実行した場合も、ノンルートモードからルートモードに遷移する。

【0034】

そして、このように、ルートモードと、ノンルートモードとを行き来しながら各種の処理を実現するために、サーバ装置1には、CPU2の状態の保存や、モード状態の遷移を制御する仮想制御データ構造体(VMCS; Virtual Machine Control Structure)が作成される。

【0035】

図4に一例を示すように、仮想制御データ構造体30は、6つの区画で定義されている。6つの区画は、ゲストステートエリア31と、ホストステートエリア32と、VM-execution制御フィールド33と、VM-exit制御フィールド34と、VM-entry制御フィールド35と、VM-exit情報フィールド36である。これら6つの区画のデータをメモリ3に退避させたり、復元させたりすることによって動作モードの切替が可能になる。

【0036】

ここで、ゲストステートエリア31は、仮想マシン22, 23の処理を行うCPU2のコントロールレジスタであるCR3レジスタや、その他のレジスタの一部を保持する。ゲストステートエリア31のデータを取得することによって、CPU2の各種のレジスタの値を取得することが可能になる。

ホストステートエリア32は、仮想マシン22, 23用のCPU2のレジスタ値を保存する。

VM-execution制御フィールド33は、ノンルートモードでのCPU2の挙動を制御する。

VM-exit制御フィールド34は、ルートモードでのCPU2の挙動を制御する。

VM-exit情報フィールド35は、最後に発生したVM-exitの情報を保持する。

10

20

30

40

50



## 【 0 0 3 7 】

ここで、CPU 2 は、インテル社製に限定されず、例えばAMD社などその他のメーカーの製品でも良い。この場合には、CPU 2 の種類に応じて仮想制御データ構造体の構造も変化する。例えば、AMD社のCPU 2 の場合の仮想制御データ構造体は、VMCB (Virtual Machine Control Block) データ構造体と呼ばれる。

## 【 0 0 3 8 】

次に、図4AにプログラムカウンタとCR3レジスタから得られるトレース情報41 (第2の情報) の一例を示す。トレース情報41は、メモリ3から仮想制御データ構造体30のデータ、特にゲストステートエリア31を取得することによって形成される。ここで、トレース情報41は、CPU 2 の性能カウンタを使用して1m秒から1μ秒間隔のサンプリング割り込み処理を実行することによってリアルタイムで取得される。トレース情報41をリアルタイムで採取するのは、サーバ装置1内で命令アドレスで特定されるプロセスが実行された時間を正確に把握する必要があるためである。

## 【 0 0 3 9 】

図4Aに示すように、トレース情報41は、データが時系列に配列されたリストからなり、リストの各行にはCPU番号42と、プロセスID43と、スレッドID44、命令アドレス(IP)45と、命令アドレスで特定されるプロセスが実行された時間データ46と、仮想マシン22、23の実行時の追加情報47とが配列され、仮想マシン番号48 (仮想マシン22、23の識別番号) と関連付けて作成される。CPU番号42は、複数のCPU 2 のいずれかを特定する情報である。プロセスID43は、拡張ページテーブルのアドレスを特定する情報であり、仮想制御データ構造体30から取得できる。スレッドID44は、スレッド毎に付与される情報である。命令アドレス45は、実行される命令のアドレスを特定する情報であり、仮想制御データ構造体30から取得できる。時間データ46は、性能カウンタを用いてデータを取得したタイミングを示す。

## 【 0 0 4 0 】

追加情報47は、ルートモード状態では使用されず、ノンルートモード状態でのみ付加される。追加情報47としては、例えば、仮想CPU番号47Aと、CR3レジスタ情報47B (ページテーブルアドレス値) と、実行アドレス47C (命令のアドレス) があり、これらのデータが順番に配列される。仮想CPU番号47A、CR3レジスタ情報47B、及び実行アドレス47Cは、トレース情報採取部11によってサンプリング割り込み処理の割り込み元のコンテキスト退避領域から採取されるもので、例えば、図3の仮想制御データ構造体30のゲストステートエリア31のCR3レジスタや、プログラムカウンタの値を参照することで得られる。

## 【 0 0 4 1 】

続いて、図4Bにトレース情報41の具体例を示す。

トレース情報41の一行目は、CPU番号42が「0」になっている。これには物理的なCPU 2 に付与されている番号に相当する。プロセスID43は、「35965 T G I D」である。スレッドID44は、「35965 A D D R」であり、命令アドレス45は、「00000034fd c 82 b 4 0」になっている。さらに、このデータを取得した時間データ46として、「I T C 00003362 f 1 6 f 4 d c」が登録されている。このプロセスは、ハイパーバイザー21で実行される処理なので、追加情報47は付加されていない。

## 【 0 0 4 2 】

これに対して、リストの9行目及び10行目のデータは、仮想マシン22、23で実行される処理なので、追加情報47が付加されている。リストの10行目のデータは、CPU番号42は「1」になっている、スレッドID44には、ゲストOS25、27であることを示す情報「[ g u e s t O S ] : : R H E L 6 . 2 g u e s t 9 9 9」が付与されている。この情報は、この実施の形態に必須の要素ではないが、ゲストOS25、27であることを示す情報を付与することは、サーバ装置1における処理の可視化に貢献する。

10

20

30

40

50

## 【 0 0 4 3 】

さらに、リストの 1 0 行目の付加情報 4 7 である仮想 C P U 番号 4 7 A は、「 V P I D 1 」になっており、これはハイパーバイザー 2 1 によってユーザに提供される仮想 C P U の番号が「 1 」であることを示している。さらに、 C R 3 情報 4 7 B は、「 g c r 3 0 0 0 0 0 0 4 1 0 6 b c 0 0 0 」であり、実行アドレス 4 7 C は「 g R i P 0 0 0 0 0 0 3 3 b b c 0 9 a f 2 」になっている。

## 【 0 0 4 4 】

ここで、トレース情報 4 1 だけでは、具体的な処理の内容を特定することができない。そこで、この実施の形態では、トレース情報 4 1 のプロセス I D 4 3 をプロセス名に、命令アドレス 4 5 をプロセスの関数名にそれぞれ変換する。

10

## 【 0 0 4 5 】

具体的には、図 5 に模式的に示すように、サーバ装置 1 のシンボルマップ情報採取部 1 3 は、各仮想マシン 2 2 , 2 3 において実行されるプロセス名やプロセス I D に関する情報と、プロセスで用いられている関数の情報である関数名 5 9 と、関数アドレス 5 8 を含むシンボルマップ情報 5 1 を作成する。そして、シンボルマップ情報 5 1 と、ハイパーバイザー 2 1 上で作成されるトレース情報 4 1 とをプロセス毎に対応付けることによって、サーバ装置 1 における処理の流れを可視化する解析データ 7 1 を作成する。

## 【 0 0 4 6 】

図 5 に示す例で、シンボルマップ情報 5 1 は、例えば、第 1 の仮想マシン 2 2 で実行されるアプリケーション 2 6 A で生成されるプロセスについて作成される。即ち、アプリケーション 2 6 A に関するシンボルマップ情報 5 1 A には、アプリケーション 2 6 A から生成されたプロセスのプロセス名 5 6 や、関数の実行アドレスである関数アドレス 5 8 、当該プロセスで実行される関数名 5 9 の情報が含まれる。複数のプロセスが平行して処理されている場合には、それら全てのプロセスについての情報が採取される。例えば、アプリケーション 2 6 B に関するシンボルマップ情報 5 1 B などが作成される。

20

## 【 0 0 4 7 】

また、第 2 の仮想マシン 2 3 で実行されるプロセスについてもシンボルマップ情報 5 1 C が作成される。シンボルマップ情報 5 1 に含まれる、プロセス名や関数名の情報は、プロセスの実行時に動的に変更される情報ではないため、リアルタイムで採取する必要はなく、プロセスが存在する間に取得されればよい。従って、シンボルマップ情報 5 1 は、例えば、ディレイドタイム処理によって採取される。

30

## 【 0 0 4 8 】

図 6 A に模式的に示すようにシンボルマップ情報 5 1 はプロセス毎に作成され、 C P U 番号 5 2 と、プロセス I D 5 3 A と、ページテーブルアドレス 5 3 B ( アドレス領域情報 ) と、スレッド I D 5 4 と、時間 5 5 と、プロセス名 5 6 と、関数アドレス 5 8 と、関数名 5 9 とを有する。プロセス I D 5 3 A 、ページテーブルアドレス 5 3 B 、スレッド I D 5 4 、プロセス名 5 6 は、仮想マシン内でプロセス毎に作成される管理領域 5 3 ( タスク構造体 ) から取得される。また、関数アドレス 5 8 と、関数名 5 9 は、アプリケーション 2 6 A に対応するオブジェクトファイル 5 7 から取得される。

## 【 0 0 4 9 】

さらに、図 6 B にシンボルマップ情報 5 1 の具体例を示す。この具体例では、 C P U 番号 5 2 、この場合は仮想 C P U が「 0 」であり、プロセス名 5 6 が「 e x e ( s d a \_ \_ c m d ) 」というモジュールであることがわかる。ここで、 1 つの関数アドレス 5 8 から次の関数アドレス 5 8 までの間のアドレスが一つの関数に使用される。例えば、関数アドレス 5 8 が「 4 0 0 0 0 0 0 0 0 0 0 0 4 2 6 0 」から「 4 0 0 0 0 0 0 0 0 0 0 1 c 6 e 0 」の 1 つ前のアドレスまでが「 m a i n 」関数に使用されている。

40

## 【 0 0 5 0 】

次に、図 7 A を参照して、トレース情報 4 1 とシンボルマップ情報 5 1 と対応付けることによって作成される解析データ 7 1 について説明する。トレース情報 4 1 と、シンボルマップ情報 5 1 は、同じ仮想マシン番号 4 8 を有する情報が組み合われるようにして回世

50

紀される。シンボルマップ情報 5 1 における仮想マシン番号 4 8 は、情報採取後に付与される。

【 0 0 5 1 】

解析データ 7 1 は、ルートモード状態では、CPU 番号 4 2 と、プロセス名 5 6 と、関数名 5 9 と、時間データ 4 6 と、仮想マシン 2 2 , 2 3 の実行時の追加情報 4 7 とが配列される。追加情報 4 7 には、仮想 CPU 番号 4 7 A と、CR3 レジスタ情報 4 7 B と、実行アドレス 4 7 C が含まれる。即ち、この実施の形態において、解析データ 7 1 は、図 4 A におけるプロセス ID 4 3 及び命令アドレス 4 5 が、図 6 A におけるプロセス名 5 6 及び関数名 5 9 に変換されることによって形成される。なお、解析データ 7 1 は、追加情報 4 7 を有しなくても良い。

10

【 0 0 5 2 】

ここで、図 7 B に示すように、解析データ 7 1 のプロセス名 5 6 は、トレース情報 4 7 のアドレス領域情報と、シンボルマップ情報 5 1 のアドレス領域情報とをプロセス単位で関連付けることによって作成される。具体的には、トレース情報 4 7 の追加情報 4 7 の CR3 レジスタ情報 4 7 B で、シンボルマップ情報 5 1 のページテーブルアドレス 5 3 B を検索してプロセス ID 5 3 を特定し、そのプロセス ID 5 3 に割り当てられているプロセス名 5 6 を取得する。これによって、

【 0 0 5 3 】

また、解析データ 7 1 の関数名 5 9 は、トレース情報 4 7 のアドレス領域情報と、シンボルマップ情報 5 1 のアドレス領域情報とをプロセス単位で関連付けることによって作成される。即ち、解析処理部 1 6 は、トレース情報 4 1 及びシンボルマップ情報 5 1 のアドレス領域情報を関連付けると共に、トレース情報 4 1 の実行アドレス 4 7 C とシンボルマップ情報 5 1 の関数アドレス 5 8 及び関数名 5 9 を関連付ける。

20

具体的には、トレース情報 4 1 の追加情報 4 7 に含まれる実行アドレス 4 7 C で、シンボルマップ情報 5 1 のページテーブルアドレス 5 3 B を検索して関数アドレス 5 8 を特定し、実行アドレス 4 7 C が含まれる関数アドレス 5 8 の領域に割り当てられた関数名 5 9 を取得する。

【 0 0 5 4 】

さらに、解析データ 7 1 には、トレース情報 4 7 に含まれる時間データ 4 6 が添付される。これによって、CPU 2 を用いて取得した時間情報と、プロセス名 5 6 、関数名 5 9 が関連付けられる。また、ハイパーバイザー 2 1 内の処理に関しては、命令アドレス 4 5 でシンボルマップ情報 5 1 A を検索して関数名 5 9 を特定する。

30

【 0 0 5 5 】

このような解析データ 7 1 は、プロセス名 5 6 、関数名 5 9 と、時間データ 4 6 の情報が関連付けられた構成を有するので、システム管理者がサーバ装置 1 における処理の流れを把握し易い。例えば、プロセス名 5 6 と関数名 5 9 から通常の処理に必要な時間がわかるので、解析データ 7 1 上で同じプロセス名 5 6 及び関数名 5 9 の処理時間が異常に多くなっている場合には、そのプロセス及び関数がオーバーヘッドの原因になっていると判定できるようになる。

【 0 0 5 6 】

40

次に、図 8 A のフローチャートを参照してトレース情報 4 1 、シンボルマップ情報 5 1 の採取処理、及び解析データ 7 1 の作成処理について説明する。

ステップ S 1 0 1 でハイパーバイザー 2 1 が情報採取を開始する。このとき、初期情報として、例えばサンプリング間隔や、情報採取の実行回数、情報採取対象に含まれる仮想マシン 2 2 , 2 3 の情報を取得する。

続くステップ S 1 0 2 では、情報採取プロセスを起動させ、ステップ S 1 0 3 で対象として抽出された仮想マシン 2 2 , 2 3 に情報採取開始の合図を送る。

【 0 0 5 7 】

さらに、ステップ S 1 0 4 において、指定時間が経過するまで待機する。指定時間は、ステップ S 1 0 1 で取得したサンプリング間隔に相当する。サンプリング間隔は、サーバ

50

装置 1 において予め設定されている値が用いられる。指定時間の経過は、C P U 2 の性能カウンタのカウント値によって判断される。そして、指定時間が経過したら（ステップ S 1 0 4 で Y e s ）、ステップ S 1 0 5 に進んでトレース情報採取部 1 1 がトレース情報 4 1 を採取する。

【 0 0 5 8 】

さらに、端子 B から進む図 8 B のステップ S 1 0 6 で、シンボルマップ情報採取部 1 3 がハイパーバイザー 2 1 内のプロセスの E x i t イベントの有無を確認する。E x i t イベントとは、例えば、プロセスの終了処理である。E x i t イベントが発生すると、そのプロセスが終了してしまい、その後は当該プロセスについてのシンボルマップ情報 5 1 A を取得できなくなる。そこで、E x i t イベントが発生したら（ステップ S 1 0 6 で Y e s ）、ステップ S 1 0 7 で、シンボルマップ情報採取部 1 3 の処理によって E x i t イベントの対象となるプログラムを処理していたハイパーバイザー 2 1 のシンボルマップ情報 5 1 を採取する。これに対して、E x i t イベントが発生していないときは（ステップ S 1 0 6 で N o ）、現段階ではシンボルマップ情報 5 1 A を取得せずに、ステップ S 1 0 8 に進む。

【 0 0 5 9 】

ステップ S 1 0 8 では、シンボルマップ情報採取部 1 3 が情報採取の終了を判定する。情報採取を行った実行回数がステップ S 1 0 1 で取得した情報採取の実行回数を経過したら（ステップ S 1 0 8 で Y e s ）、ステップ S 1 0 9 で情報収集を終了する。一方、情報採取の実行回数に到達していなければ、実行回数を 1 つインクリメントした後、端子 A からステップ S 1 0 4 に戻り、前記の処理を繰り返す。

【 0 0 6 0 】

そして、ステップ S 1 0 9 でシンボルマップ情報収集を終了したら、ステップ S 1 1 0 でシンボルマップ情報採取部 1 3 が再びシンボルマップ情報 5 1 A を採取する。ここでは、E x i t イベントの有無とは関係なく、ハイパーバイザー 2 1 で現在実行中の全てのプロセスに関する情報を採取してシンボルマップ情報 5 1 A を作成する。さらに、シンボルマップ情報採取部 1 3 は、ステップ S 1 0 8 で採取したシンボルマップ情報 5 1 A も付加する。これによって、情報採取プロセスの実行中にハイパーバイザー 2 1 で処理された全てのプロセスについてのシンボルマップ情報 5 1 A が得られる。

【 0 0 6 1 】

一方、図 8 A に示すように、第 1 の仮想マシン 2 2 は、ステップ S 1 0 3 でハイパーバイザー 2 1 から出力された開始合図を受け取ったら、ステップ S 1 1 1 でシンボルマップ情報 5 1 の採取プロセスを起動する。そして、ステップ S 1 1 2 で E x i t イベントの有無を確認する。E x i t イベントが発生したら（ステップ S 1 1 2 で Y e s ）、ステップ S 1 1 3 でシンボルマップ情報採取部 1 3 が E x i t イベントの対象となったプログラムについてのシンボルマップ情報 5 1 を採取する。E x i t イベントが発生していないときは（ステップ S 1 1 2 で N o ）、現段階ではシンボルマップ情報 5 1 を取得せずに待機する。

【 0 0 6 2 】

同様に、第 2 の仮想マシン 2 3 は、ハイパーバイザー 2 1 から出力された開始合図を受け取ったら、ステップ S 1 1 5 でシンボルマップ情報採取部 1 3 がシンボルマップ情報 5 1 の採取プロセスを起動する。この際、ステップ S 1 1 6 で E x i t イベントの有無を確認し、E x i t イベントが発生したら（ステップ S 1 1 6 で Y e s ）、ステップ S 1 1 7 でシンボルマップ情報採取部 1 3 が E x i t イベントの対象となったプログラムについてのシンボルマップ情報 5 1 を採取する。E x i t イベントが発生していないときは（ステップ S 1 1 6 で N o ）、現段階ではシンボルマップ情報 5 1 を取得せずに待機する。

【 0 0 6 3 】

この後、図 8 B のステップ S 1 2 0 で、シンボルマップ情報採取部 1 3 の処理によって、ハイパーバイザー 2 1 が仮想マシン 2 2 , 2 3 に終了合図を出力する。この後、図 8 C において、ステップ S 1 2 1 として、トレース情報採取部 1 1 の処理によって、ハイパー

バイザー 2 1 がトレース情報 4 1 をファイル出力する。続いて、ステップ S 1 2 2 で、シンボルマップ情報採取部 1 3 の処理によって、ハイパーバイザー 2 1 がシンボルマップ情報 5 1 A をファイル出力する。

【 0 0 6 4 】

一方、第 1 の仮想マシン 2 2 は、ハイパーバイザー 2 1 から終了合図を受け取ったら、ステップ S 1 2 3 でシンボルマップ情報採取部 1 3 がシンボルマップ情報 5 1 B を採取する。ここでは、E x i t イベントの有無とは関係なく、第 1 の仮想マシン 2 2 で現在実行中の全てのプロセスに関する情報を採取してシンボルマップ情報 5 1 B を作成する。さらに、ステップ S 1 1 3 でシンボルマップ情報採取部 1 3 が採取したシンボルマップ情報 5 1 B も付加する。これによって、情報採取プロセスの実行中に第 1 の仮想マシン 2 2 で処理された全てのプロセスについてのシンボルマップ情報 5 1 B が得られる。そして、ステップ S 1 2 4 で、シンボルマップ情報採取部 1 3 の処理により、第 1 の仮想マシン 2 2 でシンボルマップ情報 5 1 B がファイル出力される。

10

【 0 0 6 5 】

同様に、第 2 の仮想マシン 2 3 は、ハイパーバイザー 2 1 から終了合図を受け取ったら、ステップ S 1 2 5 でシンボルマップ情報採取部 1 3 がシンボルマップ情報 5 1 C を採取する。E x i t イベントの有無とは関係なく、第 2 の仮想マシン 2 3 で現在実行中の全てのプロセスに関する情報を採取し、ステップ S 1 1 7 で採取したシンボルマップ情報 5 1 C も付加する。これによって、情報採取プロセスの実行中に第 2 の仮想マシン 2 3 で処理された全てのプロセスについてのシンボルマップ情報 5 1 C が得られる。そして、ステップ S 1 2 6 で、シンボルマップ情報採取部 1 3 の処理により、第 2 の仮想マシン 2 3 でシンボルマップ情報 5 1 C がファイル出力される。

20

【 0 0 6 6 】

次に、ステップ S 1 3 1 で、解析処理部 1 6 の処理として、ハイパーバイザー 2 1 がデータ収集処理を開始する。これを受けて、ステップ S 1 3 2 でデータ送信部 1 4 が第 1 の仮想マシン 2 2 のシンボルマップ情報 5 1 のファイルをハイパーバイザー 2 1 にデータ転送する。同様に、ステップ S 1 3 3 でデータ送信部 1 4 が第 2 の仮想マシン 2 3 のシンボルマップ情報 5 1 のファイルをハイパーバイザー 2 1 にデータ転送する。データ転送は、例えば、s c p (Secure Copy) や、s s h f s (SSH File System) を用いたセキュリティの高いファイル転送システムを用いて受け渡される。これによって、仮想マシン 2 2 , 2 3 の情報漏洩を防止できる。

30

【 0 0 6 7 】

ハイパーバイザー 2 1 は、データ受信部 1 5 で仮想マシン 2 2 , 2 3 のシンボルマップ情報 5 1 のファイルを受け取ったら、各シンボルマップ情報 5 1 に対応する仮想マシン番号 4 8 を付与する。さらに、ステップ S 1 3 4 でデータの受け取り通知を仮想マシン 2 2 , 2 3 に出力する。その結果、ステップ S 1 3 5 で第 1 の仮想マシン 2 2 がシンボル採取プロセスを終了する。同様に、ステップ S 1 3 6 で第 2 の仮想マシン 2 3 がシンボル採取プロセスを終了する。さらに、ステップ S 1 4 1 で解析処理部 1 6 がハイパーバイザー 2 1 上で各データファイルを結合させ、ステップ S 1 4 2 でデータ分析を行い、ステップ S 1 4 3 で解析データ 7 1 を作成する。この後、ここでの処理を終了する。

40

【 0 0 6 8 】

以上、説明したように、この実施の形態では、ハイパーバイザー 2 1 の処理を時系列に従って配列したトレース情報 4 1 に、仮想マシン 2 2 , 2 3 に指令した処理の情報を追加情報 4 7 として取得するようにした。これによって、ハイパーバイザー 2 1 の処理の中で、仮想マシン 2 2 , 2 3 が実行する処理を把握することが可能になる。仮想マシン 2 2 , 2 3 毎に仮想割り込み処理を行う場合に比べて、システム全体の負荷を低減できる。

【 0 0 6 9 】

ここで、追加情報 4 7 の取得には、仮想環境を提供するために C P U 2 に用意されたコンテキスト退避領域を参照するようにしたので、仮想マシン 2 2 , 2 2 のセキュリティを担保しつつ、追加情報 4 7 を確実に所得できる。さらに、サンプリング周期を C P U 2 の

50

性能カウンタを用いて管理し、ハイパーバイザー 21 で一元的にデータを採取するようにしたので共通時間軸によるデータ採集が可能になる。このために、ソフトウェア毎のカウンタで時間を管理する場合に比べて、プロセスが実行された時間の情報の正確さを高められる。

#### 【0070】

さらに、各仮想マシン 22, 23 では、ゲスト OS 25, 27 におけるプロセス ID 53、命令アドレス 45、プロセス名 56、及び関数名 59 をシンボルマップ情報 51B, 51C として取得するようにした。そして、これらのデータを用いてハイパーバイザー 21 で作成されるトレース情報 41 の一部のデータを置き換えることによって、解析データ 71 を作成するようにした。解析データ 71 を利用することによって、システム管理者がサーバ装置 1 の処理の流れを容易に確認でき、サーバ装置 1 におけるオーバーヘッドの原因などを解析できる。各仮想マシン 22, 23 で実行されるアプリケーション・プログラム 26A ~ 26C, 28A ~ 28c は、システム管理者が用意したものではないので、処理の詳細を把握することは困難である。この実施の形態では、プロファイル処理部 17 で解析データ 71 を参照したり、処理したりすることで、ハイパーバイザー 21 や各仮想マシン 22, 23 における処理の実行に要する時間を、プロセス毎や、関数毎に把握することができる。例えば、システム管理者が、解析データ 71 の時間データ 46 のデータを調べて通常処理に比べて実行時間が長いプロセスを探せば、オーバーヘッドの原因を特定することが可能になり、解析作業の効率が向上する。これに加えて、ハイパーバイザー 21 内の処理についても、同様に置き換え処理を実施するようにしたので、システム管理者によるサーバ装置 1 の処理の流れの確認作業がさらに容易になる。

#### 【0071】

ここで、サーバ装置 1 に接続され、解析データ 71 を作成するコンピュータ端末を解析装置として設けても良い。この場合、解析装置は、図 1 に示す解析処理部 16 を少なくとも有し、サーバ装置 1 からトレース情報 41 とシンボルマップ情報 51 を受け取って、解析データ 71 を作成するように構成される。

#### 【0072】

また、本実施の形態には、コンピュータをトレース情報採取部 11 と、シンボルマップ情報採取部 13 と、解析処理部 16 にして機能させる仮想計算機システムの管理プログラムも含まれる。さらに、仮想計算機システムの管理プログラムをコンピュータにインストール可能にアップロードしたり、記録媒体に記録したりすることも実施形態に含まれる。また、本実施の形態は、OS やハイパーバイザー 21、ドライバモジュール、性能測定及び解析ツール、性能管理アプリケーション、キャパシティ管理アプリケーションに適用することが可能である。

#### 【0073】

(第 2 の実施の形態)

第 2 の実施の形態について説明する。以下において、第 1 の実施の形態と同じ構成要素には同一の符号を付している。また、第 1 の実施の形態と重複する説明は省略する。

また、図 9 に示すように、仮想計算機システムを実現するサーバ装置 1 は、トレース情報採取部 11 と、採取開始指示部 12 と、シンボルマップ情報採取部 13 と、データ送信部 14 と、データ受信部 15 と、解析処理部 16 と、プロファイル処理部 17 とに機能分割できる。

採取開始指示部 12 は、トレース情報採取部 11 におけるデータ採取が終了した後、仮想マシン 22, 23 に対して情報採取を指示する処理を実行する。

シンボルマップ情報採取部 13 は、サンプリング割り込み処理発生後に採取開始指示部 12 からの指示を受けてシンボルマップ情報 53 を採取する。

#### 【0074】

解析データを作成するときには、最初に、サンプリング割り込み処理の実行に伴い、ホスト側 (ハイパーバイザー 21 側) においてトレース情報採取部 11 が仮想マシン 22, 23 のコンテキスト退避領域から情報を採取し、仮想マシン番号 48 と関連付けてトレース

ス情報 4 1 を作成する。トレース情報 4 1 は、トレース情報採取部 1 1 から解析処理部 1 6 に受け渡される。

トレース情報採取部 1 1 における情報採取が終了したら、採取開始指示部 1 2 がゲスト側（仮想マシン 2 2 , 2 3 側）のシンボルマップ情報採取部 1 3 に対し、シンボルマップ情報 5 3 の採取を指示する信号を送る。

ここで、シンボルマップ情報 5 3 の採取対象となる仮想マシン 2 2 , 2 3 では、サンプリング割り込み処理によってプロセス等の情報がコンテキスト退避領域に退避させられているので、シンボルマップ情報採取部 1 3 を参照し、対象となる仮想マシン 2 2 , 2 3 のシンボルマップ情報 5 3 を作成する。

【 0 0 7 5 】

10

シンボルマップ情報 5 3 は、仮想マシン 2 2 , 2 3 側のデータ送信部 1 4 からハイパーバイザー 2 1 側のデータ受信部 1 5 に送信される。データ受信部 1 5 は、シンボルマップ情報 5 3 に、対応する仮想マシン番号 4 8 を付与した後、解析処理部 1 6 に受け渡す。

【 0 0 7 6 】

解析処理部 1 6 は、仮想マシン番号 4 8 及び C R 3 レジスタ情報 4 7 B（ページテーブルアドレス値）で両者の対応を調べることによって仮想マシン 2 2 , 2 3 で処理されているプロセス及び関数名を特定する。この後、プロファイル処理部 1 7 で仮想マシン 2 2 , 2 3 の動作を解析する。

【 0 0 7 7 】

以上、説明したように、この実施の形態では、仮想マシン 2 2 , 2 3 上で採取したシンボルマップ情報 5 3 をハイパーバイザー 2 1 側で解析処理するようにした。さらに、サンプリング割り込み処理時のゲスト側のプロセスの情報と、仮想マシン 2 2 , 2 3 のプロセスの情報を取得し、両者を対応付けることにより、仮想マシン 2 2 , 2 3 の処理をシステム管理者が確認できるようにした。これによって、各仮想マシン 2 2 , 2 3 の状態や、仮想マシン 2 2 , 2 3 とハイパーバイザー 2 1 の処理を含めた仮想計算機システム全体としての処理状況を正確に把握することが可能になり、仮想計算機システムを効率良く運営することができるようになる。

20

【 0 0 7 8 】

ここで挙げた全ての例及び条件的表現は、発明者が技術促進に貢献した発明及び概念を読者が理解するのを助けるためのものであり、ここで具体的に挙げたそのような例及び条件に限定することなく解釈するものであり、また、明細書におけるそのような例の編成は本発明の優劣を示すこととは関係ない。本発明の実施形態を詳細に説明したが、本発明の精神及び範囲から逸脱することなく、それに対して種々の変更、置換及び変形を施すことができる。

30

【符号の説明】

【 0 0 7 9 】

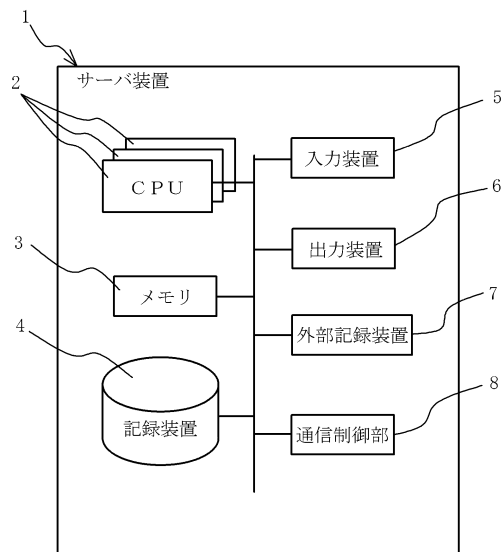
- 1 仮想計算機システム
- 1 1 トレース情報採取部
- 1 3 シンボルマップ情報採取部
- 1 6 解析処理部
- 2 1 ハイパーバイザー
- 2 2 , 2 3 仮想マシン
- 4 1 トレース情報
- 4 7 B C R 3 レジスタ情報（アドレス領域情報）
- 4 7 C 実行アドレス
- 4 8 仮想マシン識別番号
- 5 1 シンボルマップ情報
- 5 3 管理領域
- 5 3 A プロセス ID
- 5 3 B ページテーブルアドレス（アドレス領域情報）

40

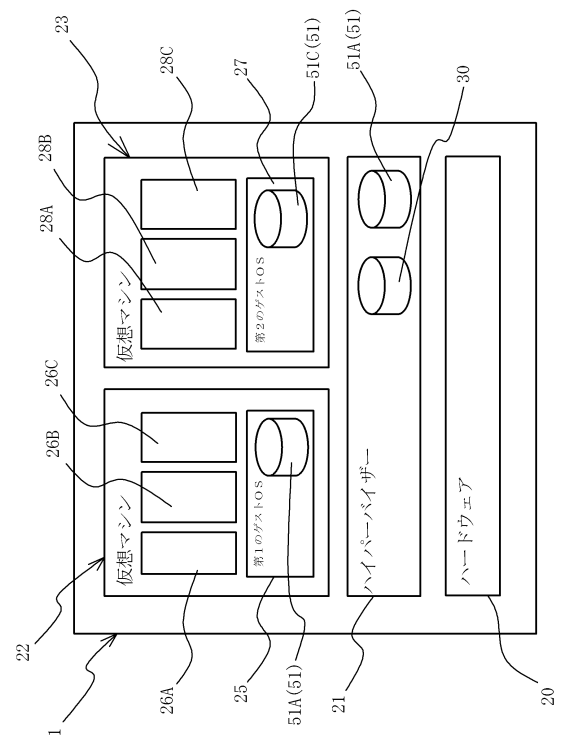
50

- 5 8 関数アドレス
- 5 9 関数名
- 7 1 解析データ

【図 1】

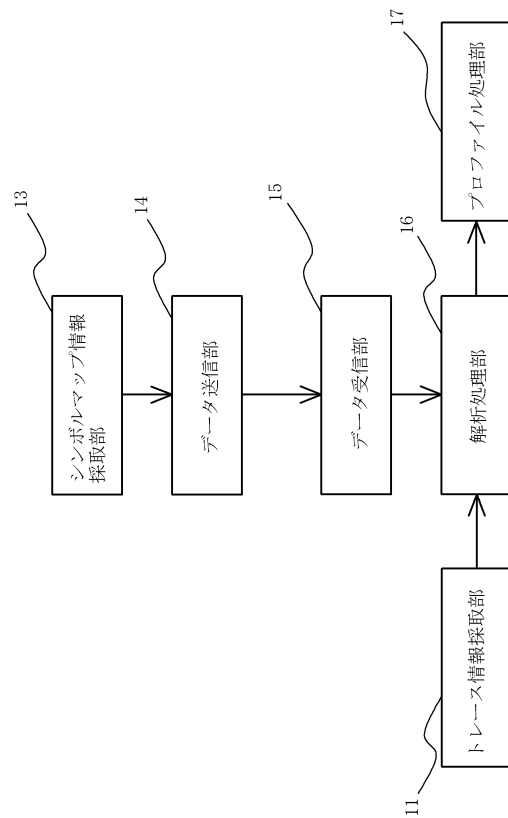


【図 2 A】

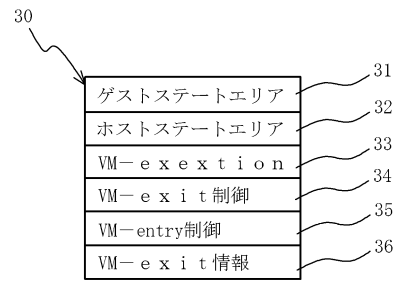




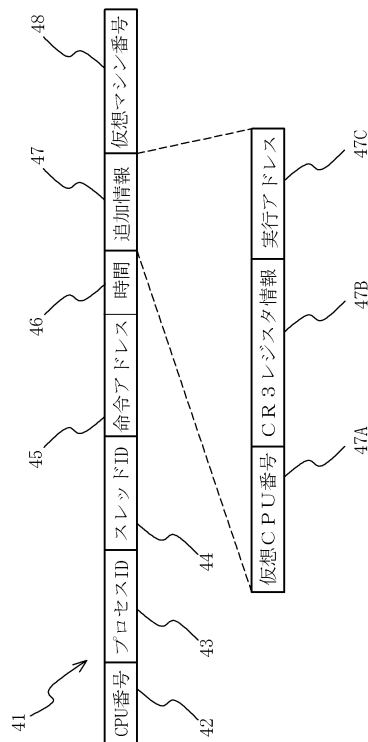
【 図 2 B 】



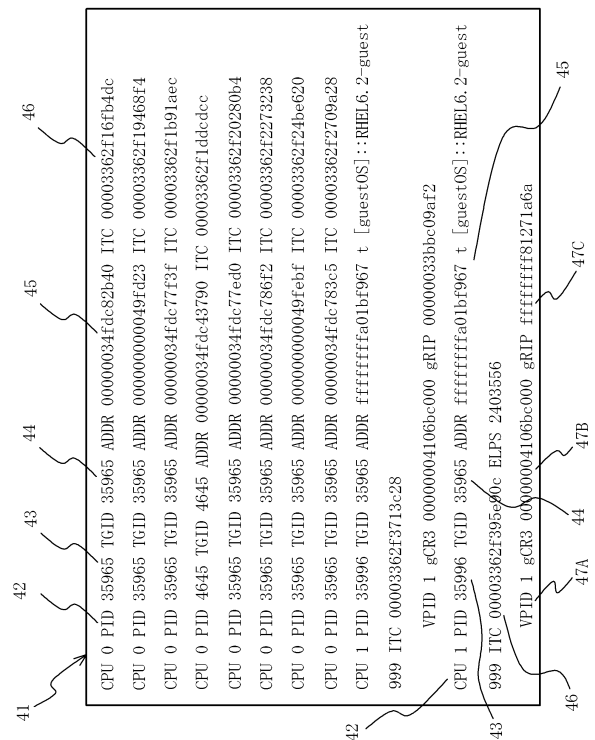
【 図 3 】



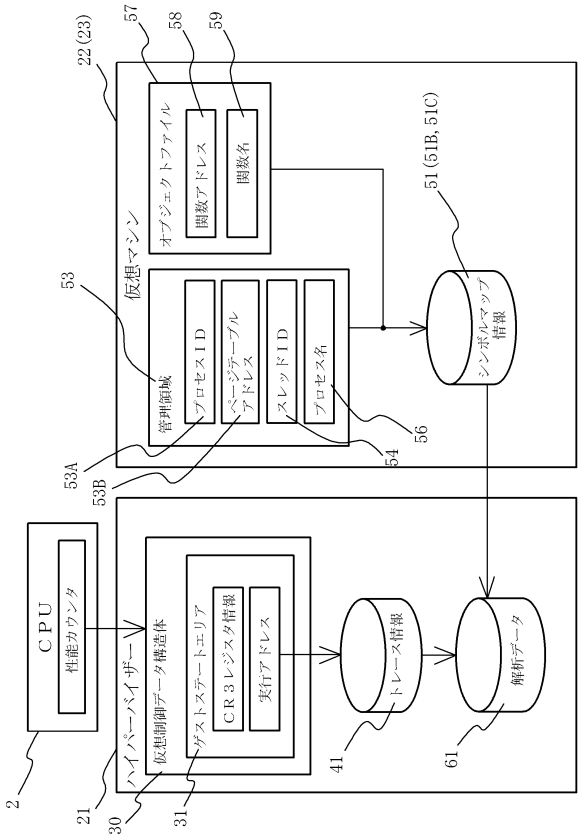
【 図 4 A 】



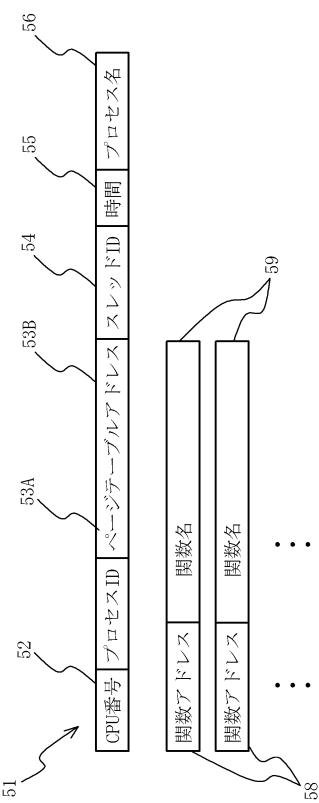
【 図 4 B 】



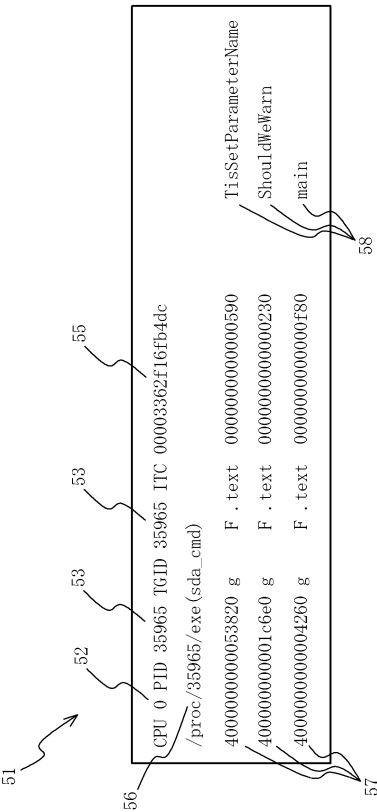
【図 5】



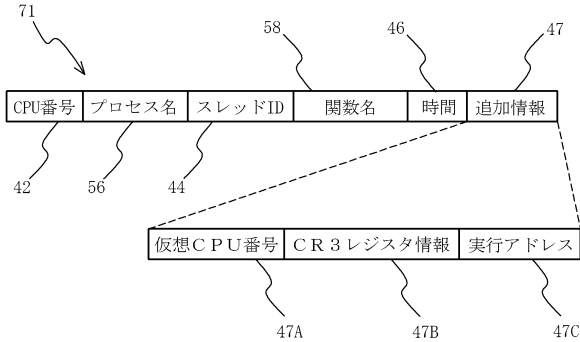
【図 6 A】



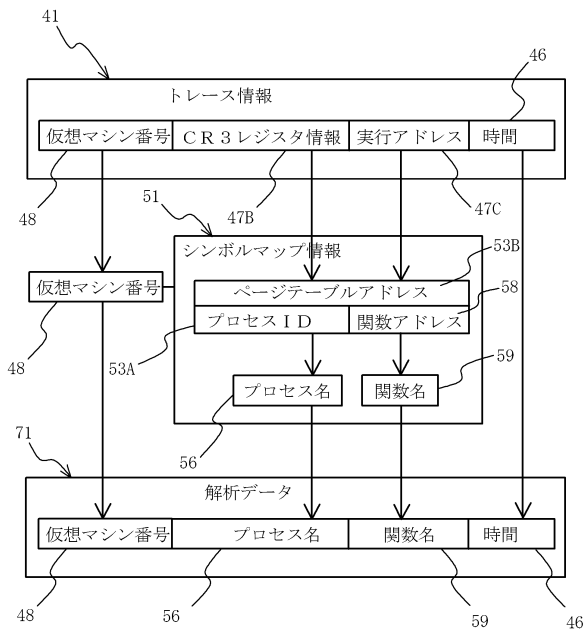
【図 6 B】



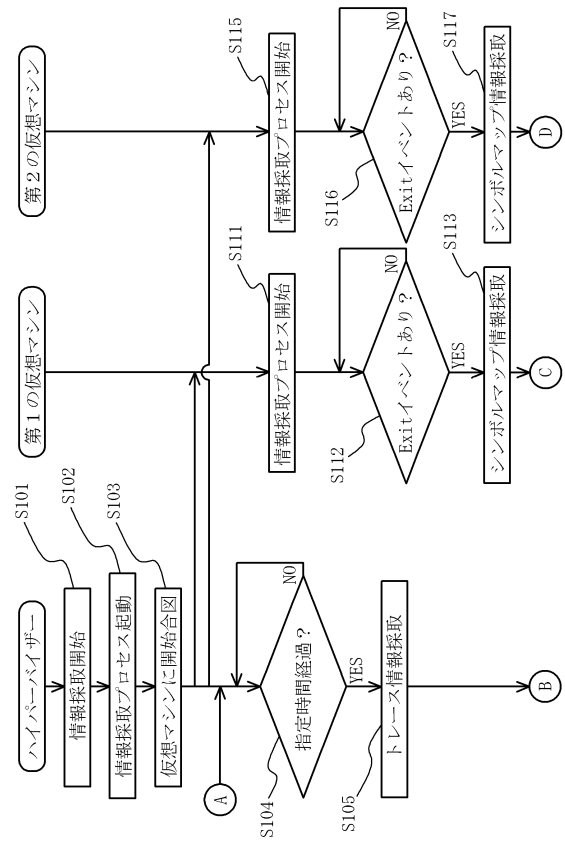
【図 7 A】



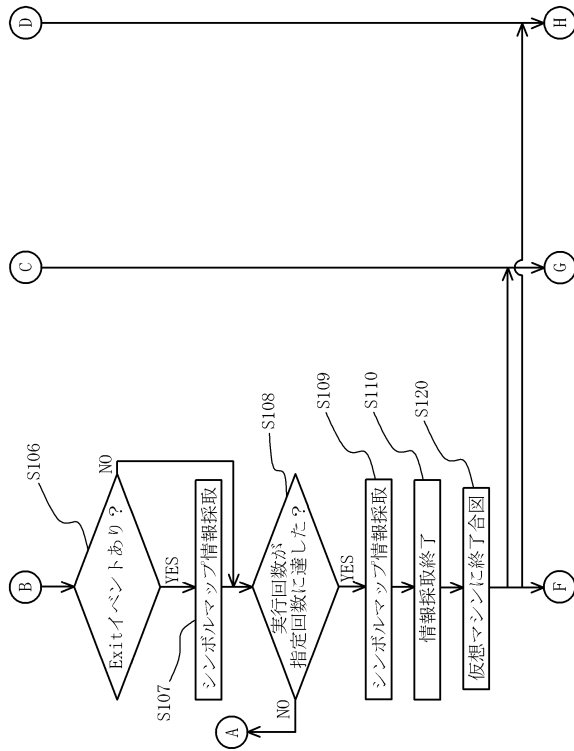
【図 7 B】



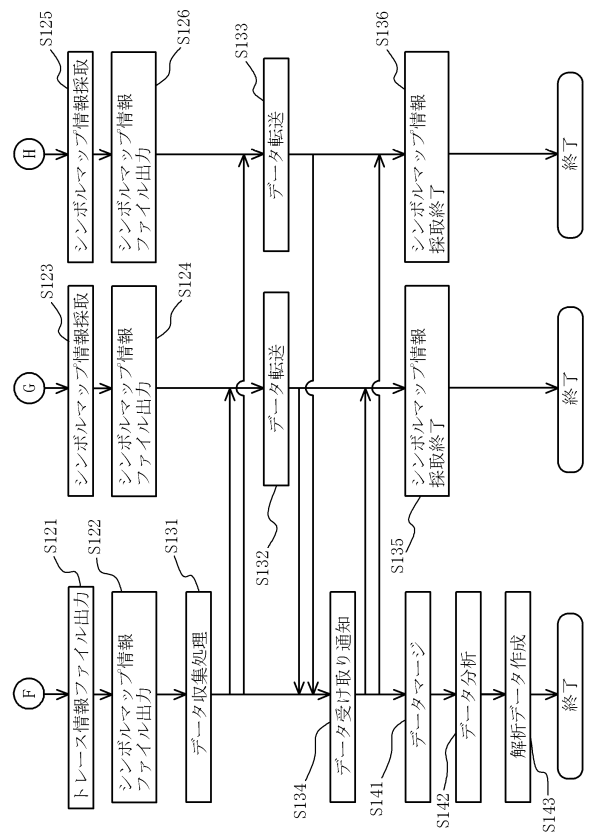
【図 8 A】



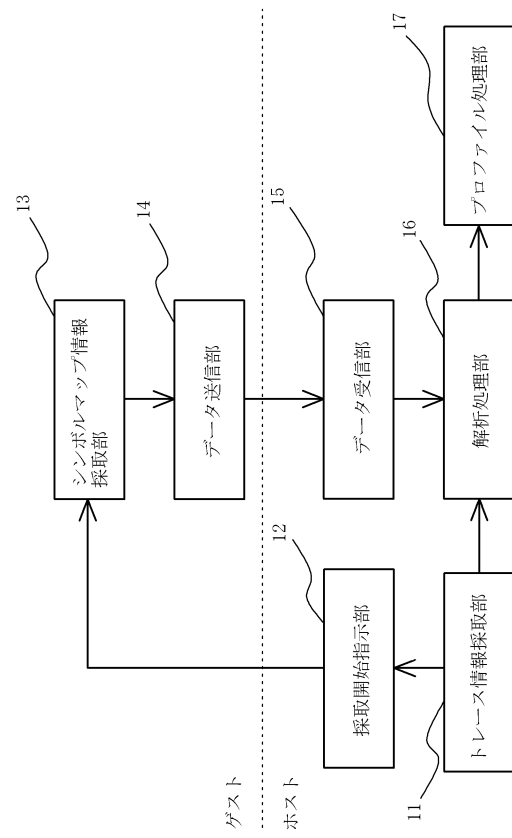
【図 8 B】



【図 8 C】



【図 9】



---

フロントページの続き

(72)発明者 山本 昌生

神奈川県川崎市中原区上小田中4丁目1番1号 富士通株式会社内

審査官 多胡 滋

(56)参考文献 国際公開第2009/147738(WO, A1)

特開平10-240581(JP, A)

山村周史, 外5名, 時系列データの統計解析によるPCクラスタシステム解析手法の提案, 先進的計算基盤システムシンポジウム SACSIS2006 論文集, 日本, 社団法人情報処理学会, 2006年 5月22日, 第2006巻, 第5号, pp.437-445

(58)調査した分野(Int.Cl., DB名)

G06F 11/34

G06F 11/30