(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4998861号
(P4998861)
(45)発行日　平成24年8月15日(2012.8.15)　(24)登録日　平成24年5月25日(2012.5.25)

(51)Int.Cl.　F I
G06F 9/445 (2006.01)　G06F 9/06 610A
G06F 9/06 610D

請求項の数 10　(全 12 頁)

(21)出願番号　特願2010-45326(P2010-45326)
(22)出願日　平成22年3月2日(2010.3.2)
(65)公開番号　特開2011-180885(P2011-180885A)
(43)公開日　平成23年9月15日(2011.9.15)
審査請求日　平成22年3月2日(2010.3.2)

(73)特許権者　000004237
日本電気株式会社
東京都港区芝五丁目７番１号
(74)代理人　100130029
弁理士　永井　道雄
(74)代理人　100166338
弁理士　関口　正夫
(74)代理人　100152054
弁理士　仲野　孝雅
(72)発明者　喜多　栄寿
東京都港区芝五丁目７番１号　日本電気株式会社内

審査官　石川　亮

最終頁に続く

(54)【発明の名称】コンピュータシステム及びそのHW抽象化方法

(57)【特許請求の範囲】
【請求項１】
ＨＷ（ハードウェア）を仮想空間のＡＣＰＩツリー上に配列したオブジェクトとして抽象化するＡＣＰＩと、
前記ＡＣＰＩを介して前記ＨＷの構成を認識するＯＳと、
前記ＡＣＰＩと前記ＨＷとの間に設けられ、前記ＡＣＰＩツリー上のオブジェクトと同じ配列で前記ＨＷ毎の情報を格納する場所を並べて構成されるメモリ領域を有する仮想ＨＷと、
事前に前記ＨＷを参照して前記ＨＷ毎の情報を前記仮想ＨＷのメモリ領域内の当該ＨＷに対応する場所に格納するＢＩＯＳとを有し、
前記ＡＣＰＩが前記ＨＷを直接参照せずに前記仮想ＨＷのメモリ領域を参照して前記ＨＷ毎の情報を得ることにより、前記ＯＳが前記ＡＣＰＩを介して前記ＨＷ毎の情報から前記ＨＷの構成を認識することを特徴とするコンピュータシステム。
【請求項２】
前記仮想ＨＷのメモリ領域は、前記ＡＣＰＩツリー上に割り当てられるＡＣＰＩ番号順に前記ＨＷの情報を格納する場所を並べて構成されることを特徴とする請求項１に記載のコンピュータシステム。
【請求項３】
前記仮想ＨＷは、デバッグ用メモリ領域をさらに有し、
前記ＡＣＰＩが前記仮想ＨＷのメモリ領域を参照する際、前記ＡＣＰＩ側のデバッグに

必要なデータを前記デバッグ用メモリ領域に格納することを特徴とする請求項１又は２に記載のコンピュータシステム。

【請求項４】

前記ＢＩＯＳは、前記ＨＷの初期化過程で当該ＨＷを参照して前記ＨＷ毎の情報を前記仮想ＨＷのメモリ領域内の当該ＨＷに対応する場所に格納することを特徴とする請求項１から３のいずれか１項に記載のコンピュータシステム。

【請求項５】

前記ＨＷの構成が前記ＯＳの運用中に動的に変更になった場合、前記ＢＩＯＳが、前記ＨＷの構成変更を前記仮想ＨＷのメモリ領域内の当該ＨＷに対応する場所の格納された前記ＨＷ毎の情報に反映させることにより、前記ＯＳが前記仮想ＨＷを介して前記ＨＷの構成変更を認識することを特徴とする請求項１から４のいずれか１項に記載のコンピュータシステム。

【請求項６】

ＡＣＰＩが、ＨＷ（ハードウェア）を仮想空間のＡＣＰＩツリー上に配列したオブジェクトとして抽象化し、

前記ＡＣＰＩと前記ＨＷとの間に、前記ＡＣＰＩツリー上のオブジェクトと同じ配列で前記ＨＷ毎の情報を格納する場所を並べて構成されるメモリ領域を有する仮想ＨＷを設け、

ＢＩＯＳが、事前に前記ＨＷを参照して前記ＨＷ毎の情報を前記仮想ＨＷのメモリ領域内の当該ＨＷに対応する場所に格納し、

前記ＡＣＰＩが前記ＨＷを直接参照せずに前記仮想ＨＷのメモリ領域を参照して前記ＨＷ毎の情報を得ることにより、ＯＳが前記ＡＣＰＩを介して前記ＨＷ毎の情報から前記ＨＷの構成を認識することを特徴とするコンピュータシステムのＨＷ抽象化方法。

【請求項７】

前記仮想ＨＷのメモリ領域は、前記ＡＣＰＩツリー上に割り当てられるＡＣＰＩ番号順に前記ＨＷの情報を格納する場所を並べて構成されることを特徴とする請求項６に記載のコンピュータシステムのＨＷ抽象化方法。

【請求項８】

前記仮想ＨＷは、デバッグ用メモリ領域をさらに有し、

前記ＡＣＰＩが前記仮想ＨＷのメモリ領域を参照する際、前記ＡＣＰＩ側のデバッグに必要なデータを前記デバッグ用メモリ領域に格納することを特徴とする請求項６又は７に記載のコンピュータシステムのＨＷ抽象化方法。

【請求項９】

前記ＢＩＯＳが、前記ＨＷの初期化過程で当該ＨＷを参照して前記ＨＷ毎の情報を前記仮想ＨＷのメモリ領域内の当該ＨＷに対応する場所に格納することを特徴とする請求項６から８のいずれか１項に記載のコンピュータシステムのＨＷ抽象化方法。

【請求項１０】

前記ＨＷの構成が前記ＯＳの運用中に動的に変更になった場合、前記ＢＩＯＳが、前記ＨＷの構成変更を前記仮想ＨＷのメモリ領域内の当該ＨＷに対応する場所の格納された前記ＨＷ毎の情報に反映させることにより、前記ＯＳが前記仮想ＨＷを介して前記ＨＷの構成変更を認識することを特徴とする請求項６から９のいずれか１項に記載のコンピュータシステムのＨＷ抽象化方法。

【発明の詳細な説明】

【技術分野】

【０００１】

本発明は、コンピュータシステム及びそのＨＷ（ハードウェア）抽象化方法に関する。

【背景技術】

【０００２】

コンピュータを構成するＣＰＵ（Central Processing Unit）、メモリ等のＨＷを抽象化するための標準技術として、ＡＣＰＩ（Advanced Configuration and Power Interface

）が知られている。ＡＣＰＩによってHWを抽象化することで、Ｌｉｎｕｘ、Ｗｉｎｄｏｗｓ（登録商標）などのＯＳ（Operating System）は、どのベンダのプラットホームでもブート（Ｂｏｏｔ）することが可能である。

【０００３】

具体的には、ＡＣＰＩツリー（ACPI Tree）というツリー構造の仮想空間を定義し、そのＡＣＰＩツリー上にシステムを構成するHWをデバイスオブジェクト（Device Object）として記述することでHWの抽象化を行う。ＡＣＰＩツリーの記述においては、ＡＳＬ（ACPI Source Language）という言語を用いる。各HWに対応する各オブジェクトには、対応するHW構成情報等を返却するメソッドを記述する。

【０００４】

一般的に、ＡＳＬでＡＣＰＩ内のメソッドを記述する際、HWから情報を読み出し、ＯＳへ返却する形態を用いる。そして、ＯＳ上のＡＣＰＩドライバ（ACPI Driver）がそのＡＣＰＩ内のメソッドを実行することで、HW構成を認識する。すなわち、ＯＳは、ＡＣＰＩを介してHW構成を認識する。具体的に、ＡＣＰＩがＯＳへHWの構成を認識させる方法は、次のとおりである。

【０００５】

（１）ＡＣＰＩ内のメソッドがHWの構成を参照し、データを取得する。

【０００６】

（２）ＡＣＰＩ内のメソッドが取得したデータに対し、メソッド毎の処理を行う。

【０００７】

（３）ＡＣＰＩ内のメソッドが作成したデータをＯＳへ返却する。

【０００８】

しかし、従来のシステムでＯＳへHWの構成を認識させる役割を担うものは、HWとＡＣＰＩだけであった。また、ＡＣＰＩツリーの記述で用いるＡＳＬは、Ｃ言語等の高級言語とは異なり、データ定義、ループ制御の記述に制限があり、複雑な制御プログラムの記述や、規模の大きな開発には向いていない。このため、この従来技術には、次のような問題点があった。

【０００９】

（１）大規模サーバの場合、ＡＳＬ開発規模が増加する。その理由は、ＡＣＰＩツリー上のHW数が増えるからである。

【００１０】

（２）大規模サーバの場合、開発効率化が低下し、品質悪化の懸念がある。その理由は、ＡＳＬにはデータ定義やループ制御の記述に制限があるからである。

【００１１】

（３）新機種開発などにおいて、HWが変更になると、ＡＳＬも書き換えなければならないため、流用開発が困難である。その理由は、ＡＣＰＩがHWを参照するコードをＡＳＬで記述しているからである。

【００１２】

（４）マルチＯＳをサポートする場合、非効率な面がある。その理由は、ＯＳ上のＡＣＰＩドライバがＡＳＬの記述を解釈できない場合があり、全てのＯＳが解釈できるようＡＳＬを記述しなければならなく、事前調査を必要とするからである。

【００１３】

（５）デバッグ上の不便さがある。その理由はＯＳ上のＡＣＰＩドライバがＡＳＬを解釈する方式のため、ＯＳ上からデバッグを行うのが一般的だが、ＯＳをブートできない場合、ＡＳＬのデバッグができないからである。

【００１４】

これに関連して、特許文献１には、ＡＣＰＩ制御メソッドの名前の衝突の回避方法が記載されている。この方法は、ＡＣＰＩの制御メソッドの名前の衝突を回避するコンピュータにより実行される方法であって、一意の識別子から構成される第１の引数を取得し、異なるＡＣＰＩ制御メソッドに対して同じ識別子の使用を防止することによって、名前の衝

突が回避されるように、第１の引数をＡＣＰＩ制御メソッドの予約名に関連付け、所定のＡＣＰＩ制御メソッドに対する一意の識別子を構成するものである。
【先行技術文献】
【特許文献】
【００１５】
【特許文献１】特開２００５－１８２７７９号公報
【発明の概要】
【発明が解決しようとする課題】
【００１６】
前述したＡＣＰＩによるＨＷ抽象化方式には、次のような課題がある。
【００１７】
（１）大規模サーバの場合、ＡＣＰＩツリー上のＨＷ数が増え、その結果ＡＳＬ開発規模が増加、かつ前述のＡＳＬ制限があるため、開発効率化が低下し、品質悪化の懸念がある。
【００１８】
（２）現状のＡＳＬがＨＷを直接考慮してそれを抽象化する方式では、新機種開発などにおいてＨＷが変更になるとＡＳＬも書き換えなければならないため、流用開発が困難である。
【００１９】
（３）ＯＳ上のＡＣＰＩドライバがＡＳＬの記述を解釈できない場合がある。マルチＯＳをサポートする場合、全てのＯＳが解釈できるようＡＳＬを記述しなければならなく、事前調査が必要など、非効率な面がある。
【００２０】
（４）ＯＳ上のＡＣＰＩドライバがＡＳＬを解釈する方式のため、ＯＳ上からデバッグを行うのが一般的であるが、ＯＳをブートできない場合、ＡＳＬのデバッグができないため、デバッグ上の不便さがある。
【００２１】
特許文献１の方法は、ＡＣＰＩ制御メソッドの名前の衝突を回避するものであり、上記のような課題を考慮したものではない。
【００２２】
本発明の目的は、以上の課題を解決し、ＡＳＬの開発規模を減らし、流用開発を容易にし、マルチＯＳサポートを容易にすることができるコンピュータシステム及びそのＨＷ抽象化方法を提供することにある。
【課題を解決するための手段】
【００２３】
上記目的を達成するため、本発明に係るコンピュータシステムは、ＨＷ（ハードウェア）を仮想空間のＡＣＰＩツリー上に配列したオブジェクトとして抽象化するＡＣＰＩと、前記ＡＣＰＩを介して前記ＨＷの構成を認識するＯＳと、前記ＡＣＰＩと前記ＨＷとの間に設けられ、前記ＡＣＰＩツリー上のオブジェクトと同じ配列で前記ＨＷ毎の情報を格納する場所を並べて構成されるメモリ領域を有する仮想ＨＷと、事前に前記ＨＷを参照して前記ＨＷ毎の情報を前記仮想ＨＷのメモリ領域内の当該ＨＷに対応する場所に格納するＢＩＯＳとを有し、前記ＡＣＰＩが前記ＨＷを直接参照せずに前記仮想ＨＷのメモリ領域を参照して前記ＨＷ毎の情報を得ることにより、前記ＯＳが前記ＡＣＰＩを介して前記ＨＷ毎の情報から前記ＨＷの構成を認識することを特徴とする。
【００２４】
本発明に係るコンピュータシステムのＨＷ抽象化方法は、ＡＣＰＩが、ＨＷ（ハードウェア）を仮想空間のＡＣＰＩツリー上に配列したオブジェクトとして抽象化し、前記ＡＣＰＩと前記ＨＷとの間に、前記ＡＣＰＩツリー上のオブジェクトと同じ配列で前記ＨＷ毎の情報を格納する場所を並べて構成されるメモリ領域を有する仮想ＨＷを設け、ＢＩＯＳが、事前に前記ＨＷを参照して前記ＨＷ毎の情報を前記仮想ＨＷのメモリ領域内の当該Ｈ

Ｗに対応する場所に格納し、前記ＡＣＰＩが前記ＨＷを直接参照せずに前記仮想ＨＷのメモリ領域を参照して前記ＨＷ毎の情報を得ることにより、ＯＳが前記ＡＣＰＩを介して前記ＨＷ毎の情報から前記ＨＷの構成を認識することを特徴とする。

【発明の効果】

【００２５】

本発明によれば、ＡＳＬの開発規模を減らし、流用開発を容易にし、マルチＯＳサポートを容易にすることができるコンピュータシステム及びそのＨＷ抽象化方法を提供することができる。

【図面の簡単な説明】

【００２６】

【図１】本発明の第１の実施例に係るコンピュータシステムの全体構成を示す図である。

【図２】図１に示すＡＣＰＩツリーの詳細とＨＷ構成格納部を示す図である。

【図３】図１に示すコンピュータシステムの動作を説明するフローチャートである。

【図４】本発明の第２の実施例に係るコンピュータシステムの全体構成を示す図である。

【図５】図４に示すコンピュータシステムの動作を説明するフローチャートである。

【発明を実施するための形態】

【００２７】

以下、本発明に係るコンピュータシステム及びそのＨＷ抽象化方法の実施の形態について、図面を参照して詳細に説明する。

【００２８】

本実施の形態は、大規模マルチＯＳサーバ等のコンピュータシステムにおいて、開発効率の為にＡＣＰＩメソッドがＨＷを直接参照せずにＨＷのデータを取得するものである。すなわち、本実施の形態では、ＡＣＰＩからはＨＷを直接参照せずに、ＡＣＰＩとＨＷの間に仮想ＨＷを設け、ＢＩＯＳ（Basic Input/Output System）が事前にＡＣＰＩが必要とするデータを仮想ＨＷ上に格納しておき、ＡＣＰＩからは仮想ＨＷを参照する方式（ＨＷ非直接参照型ＨＷ仮想化抽象化方式）を採用する。これにより、ＡＣＰＩ開発の効率化、品質確保が可能となり、マルチＯＳへの対応とＨＷ変更への対応が容易なものとなる。例えば、大規模マルチＯＳサーバにおけるＡＣＰＩによるＨＷ非直接参照型ＨＷ仮想化抽象化方法を構築することができる。

【００２９】

本実施の形態は、次の構成を採用している。

【００３０】

（１）ＡＣＰＩとＨＷの間に、新たに仮想ＨＷを設ける。仮想ＨＷは、ＡＣＰＩ側の各メソッドが必要とするＨＷのデータを格納するメモリ領域（図１の例では、ＨＷ構成格納部）を有する。

【００３１】

（２）ＢＩＯＳは、システム立上げ時のＨＷ初期化の過程で仮想ＨＷにデータを格納する。具体的には、メモリ等のデータ格納部にＡＣＰＩツリー上の各オブジェクトと同様な配列でＨＷの各要素を仮想的に定義し、ＡＣＰＩ側の各メソッドが必要とするＨＷのデータを生成し、仮想ＨＷ上に格納しておく。

【００３２】

（３）ＡＣＰＩ側の各メソッドは、ＨＷを参照せず、仮想ＨＷを参照するようＡＳＬを記述しておく。

【００３３】

（４）ＯＳ上のＡＣＰＩドライバがＡＣＰＩ側の各メソッドを実行すると、ＡＣＰＩは、仮想ＨＷから情報を取得してＯＳへ返却し、ＯＳは、仮想ＨＷを介してＨＷ構成を認識する。

【００３４】

（５）ＨＷ構成がＯＳ運用中に動的に変更になった場合は、ＢＩＯＳが仮想ＨＷにその変更を反映させることで、ＯＳへＨＷの構成変更を認識させる。

【００３５】
（６）仮想ＨＷにデバッグ用のストレージ（デバッグ用メモリ領域：図１の例では、トレース領域に対応する。）を設けておく。これにより、ＡＣＰＩ側から仮想ＨＷを参照する際、ＯＳ上のＡＣＰＩドライバから渡された値、あるいはＯＳへ返却する値等の情報をトレースデータとしてデバッグ用のストレージに格納する。
【００３６】
以下、本発明の具体的な実施例について説明する。
【実施例１】
【００３７】
まず、図１～図３を参照して、本発明の第１の実施例について説明する。
【００３８】
図１は、本実施例の全体構成を示す。図１を参照すると、本実施例に係るコンピュータシステム１０は、コンピュータアーキテクチャに基づくＨＷ及びＳＷ（ソフトウェア）で実現されるもので、その構成上、ＯＳ１１と、ＡＣＰＩ１２と、メモリ１３と、ＢＩＯＳ１４と、ＣＰＵ等のＨＷ１５とを有している。図１の例では、ＨＷ１５として、ＣＰＵ１、２と、ＭＥＭ（メモリ）１、２とがそれぞれ示されている。ＯＳ１１等の各ＳＷは、ＨＷ１５のＣＰＵにより実行されることにより、それぞれの機能（後述参照）が達成される。
【００３９】
ＯＳ１１は、ＡＣＰＩ１２側の各メソッドを実行するＡＣＰＩドライバ（ACPI Driver）１１１を備えている。ＡＣＰＩドライバ１１１は既知のものがそのまま適用されるため、その詳細については省略する。
【００４０】
ＡＣＰＩ１２は、ＡＳＬで記述されたツリー構造の仮想空間であるＡＣＰＩツリー（ACPI Tree）１２１を備えている。ＡＣＰＩツリー１２１上には、コンピュータシステム１０の各ＨＷが、それぞれに対応する各デバイスオブジェクト（以下オブジェクト）として抽象化される。各ＨＷに対応する各オブジェクトには、対応するＨＷ毎の情報等を返却するメソッドが記述される。
【００４１】
メモリ１３内には、仮想ＨＷ１３１が存在する。仮想ＨＷ１３１は、ＨＷ構成格納部１３１１と、デバッグ用メモリ領域であるトレース領域１３１２とで構成される。
【００４２】
仮想ＨＷ１３１は、メモリ１３内部に作成され、ＨＷ毎の情報であるＨＷ構成情報を格納する。ＨＷ構成情報とは、ＨＷの種類、ＨＷ毎の個数、その容量等を定義したものである。
【００４３】
ここで、開発者による設計段階で、メモリ１３中の何処に仮想ＨＷ１３１用の領域をアロケートするかが決定される。
【００４４】
仮想ＨＷ１３１は、次の２つの手段で実現する。第一は、ＨＷ構成情報を格納するアドレスを定義するプログラムである。第二は、ＨＷ１５を参照することで得たＨＷ構成情報を仮想ＨＷ１３１に格納するプログラムである。第二のＨＷ構成情報を仮想ＨＷ１３１に格納するプログラムにおいては、ＡＣＰＩ１２のＡＣＰＩツリー１２１上の各オブジェクトと同じ配列になるようＨＷ構成情報の格納順序を規定しておく。
【００４５】
この２つのプログラムは、ＢＩＯＳ１４に組み込まれる。ＢＩＯＳ１４は、これらプログラムを実行すると、ＨＷ構成情報を格納するアドレスを定義するプログラムに記述されているとおりにメモリ１３内のメモリ領域をアロケートする。また、ＢＩＯＳ１４は、ＨＷ１５を参照し、得たＨＷ構成情報を仮想ＨＷ１３１内の所定の場所に格納する。
【００４６】

さらに、開発者による設計段階で、ＡＣＰＩ１２が仮想ＨＷ１３１を参照するように設計される。ＡＣＰＩ１２は、前述したＡＳＬ（ＡＣＰＩツリーを記述する言語）と呼ばれる言語を用いて開発される。これにより、ＯＳ１１中に存在するＡＣＰＩドライバ１１１がＡＳＬで記述されたＡＣＰＩツリー１２上の各オブジェクトのメソッドを実行することにより、ＡＣＰＩ１２は、仮想ＨＷ１３１を参照することでＨＷ構成情報を得る。その後、ＡＣＰＩ１２は、得られたＨＷ構成情報をＯＳ１１へ渡すことでＯＳ１１にＨＷ構成情報を認識させる。

【００４７】

図２は、ＡＣＰＩツリー１２１の詳細と仮想ＨＷ１３１内のＨＷ構成格納部１３１１を示す。図２を参照すると、ＡＣＰＩツリー１２１は、ＣＰＵ等（図２の例では、ＣＰＵ１、Ｍｅｍｏｒｙ（メモリ）１、２等）のＨＷ１５がオブジェクトとして定義される。それぞれのオブジェクトは、＿ＵＩＤ（デバイス毎、ユニークに割り当てられるＩＤを返却するメソッド）等のメソッドを備えている。

【００４８】

図１で、ＯＳ１上のＡＣＰＩドライバ１１１がＡＳＬを実行すると、ＡＣＰＩ１２内の各メソッドは、それぞれ仮想ＨＷ１３１から情報を読み出し、ＯＳ１１へ返却する。

【００４９】

ＨＷ構成格納部１３１１は、仮想ＨＷ１３１内にあり、ＣＰＵ、メモリ（ＭＥＭ）等（図１の例では、ＣＰＵ１、２、３、４、ＭＥＭ１、２、３、４等）、ＨＷ１５毎の情報が格納されるメモリ領域である。

【００５０】

トレース領域１３１２とは、ＡＣＰＩ２がＯＳ１へ返却するデータ等、トレースしたいデータが格納されるメモリ領域である。

【００５１】

ＨＷ構成格納部１３１１内には、ＨＷ１５毎の情報（ＨＷ構成情報）を格納する場所が、ＡＣＰＩ番号順に並んでいる。ここで、ＡＣＰＩ番号の説明をする。同種のＨＷ１５には、番号が付けられているが、この番号付けは設計ルールとして、ＨＷ１５上とＡＣＰＩツリー１２１上で異なる。ＨＷ１５上で割り当てられる番号を物理ＨＷ番号、ＡＣＰＩツリー１２１上で割り当てられる番号をＡＣＰＩ番号と呼ぶ。物理ＨＷ番号は、基板上の配置に沿ったものが割り当てられる。一方、ＡＣＰＩ番号は、物理ＨＷ番号にＢＭＣ（Base Management Controller）やマスタＣＰＵ（Master CPU：基準となるＣＰＵ）番号の値を加味したものが割り当てられる。

【００５２】

本実施例では、ＨＷ構成格納部１３１１内で、ＡＣＰＩ番号順にＨＷ１５毎の情報を格納する場所が並んでいるので、ＡＣＰＩツリー１２１とＨＷ構成格納部１３１１内でＨＷ１５の番号順が等しくなる。つまり、ＨＷ１５の変更があった際も、ＡＣＰＩツリー１２１でのＨＷ１５とＨＷ構成格納部１３１１内のＨＷ１５の対応関係は変化しない。そのため、ＡＣＰＩ１２側の変更を最小限におさえられ、流用開発を容易にすることが可能となる。

【００５３】

次に、図３を参照して、本実施例の動作について詳細に説明する。なお、図１のＡ１～Ａ８は、図３のステップＡ１～Ａ８にそれぞれ対応する。

【００５４】

まず、ＢＩＯＳ１４は、システム立上げ時のＨＷ初期化の過程（ステップＡ１）で、ＨＷ構成格納部１３１１とトレース領域１３１２とで構成される仮想ＨＷ１３１を作成する（ステップＡ２）。その後、ＢＩＯＳ１４は、ＨＷ１５の種類、個数、容量等、ＨＷの情報をＨＷ５から参照し、参照した情報をＨＷ構成格納部１３１１に格納する（ステップＡ３）。

【００５５】

続いて、ＯＳ１１が立ち上がった（ステップＡ４）後、ＯＳ１１上のＡＣＰＩドライバ

１１１がＡＣＰＩ２側のメソッドを実行すると、そのメソッドは、HW構成格納部１３１１を参照し、データを得る（ステップＡ５）。その一例として、ＡＣＰＩ２側のメソッドを記述するＡＳＬで用意されているOperation Regionという関数を用いることができる。この関数は、アドレスを入力すると、そのアドレス上のデータを出力するものである。この場合、設計の段階でHW１５の各要素の種類毎に情報が格納されているHW構成格納部１３１１内のアドレスは定まっているので、欲しい情報が格納されているHW構成格納部１３１１内のアドレスをOperation Regionに指定することにより、そのアドレス上のデータを得ることができる。

【００５６】

ＡＣＰＩ１２内のそれぞれのメソッドは、得られたデータに対し、メソッド内の処理を実行し、ＯＳ１１へ返却するデータを生成する（ステップＡ６）。また、ＡＣＰＩ１２は、デバッグのため、そのデータを仮想HW１３１内のトレース領域１３１２に格納する（ステップＡ７）。そして、ＡＣＰＩ１２は、トレース領域１３１２に格納したものと同じデータをＯＳ１１へ返却する（ステップＡ８）。

【００５７】

従って、本実施例では、次のような効果を奏することができる。

【００５８】

第１の効果は、ＡＳＬの開発規模を減らせることにある。その理由は、従来のHW参照の処理と比較し、仮想HW参照の処理は単純であるためである。

【００５９】

第２の効果は、流用開発が容易になることである。その理由は、ＡＣＰＩ１２が仮想HW１３１を参照しており、HW変更があっても、ＡＳＬの書き換えが軽微なものとなるためである。

【００６０】

第３の効果は、マルチＯＳサポートが容易になることにある。その理由は、ＡＣＰＩ開発規模が減り、ＡＣＰＩドライバが解釈可能な標準ＡＳＬのみで記述が可能となるためである。

【００６１】

第４の効果は、ＡＣＰＩ側のデバッグをとることが可能となることである。その理由は、仮想HWストレージとしてメモリを定義することで、データを記憶させることができるためである。

【実施例２】

【００６２】

次に、図４及び図５を参照して、本発明の第２の実施例について説明する。

【００６３】

図４は、本実施例の全体構成を示す。図４を参照すると、本実施例に係るコンピュータシステム１０は、HW１５にＣＰＵ３、ＣＰＵ４が追加されている点で図１と異なる。ランタイム（Runtime）中にＣＰＵソケットなどを追加する処理を動的構成制御という。本発明は、動的構成制御に対しても機能する。

【００６４】

ここで、図１のHW５に動的構成制御によってＣＰＵ３、ＣＰＵ４が追加され、図１の構成が図４の構成になったとする。

【００６５】

図５は、本実施例による動的制御の動作ステップを示す。なお、図４のＡ１～Ａ８は、図５のステップＡ１～Ａ８にそれぞれ対応する。図５を参照すると、図３の動的構成制御でない場合の動作ステップと比較し、ステップA1～Ａ８に加え、ステップＡ４、Ａ５の間に、ステップＡ１１、Ａ１２が追加されている。ステップA1～Ａ８の動作は、図３と同じであるため、以下にステップＡ１１、Ａ１２のみ説明する。

【００６６】

まず、ＯＳ１が立ち上がった（ステップＡ４）後、ステップＡ１１では、ＣＰＵソケッ

トなどがＨＷ５に追加される（動的構成制御）。

【００６７】

次に、ステップＡ１２では、ＢＩＯＳ１４がＨＷ１５を参照し、ＨＷ構成格納部１３１１へ参照したデータを格納する。ステップＡ１２は、ステップＡ３と同じである。つまり、動的構成制御（ステップＡ１１）が終わった後、動的構成制御前に行ったステップ（仮想ＨＷ１３１、ＡＣＰＩ１２、ＯＳ１２の順にＨＷ構成情報を渡す）を反復すれば、動的構成制御に対しても本発明を機能させる（ＯＳ１１にＨＷ構成を認識させる）ことができる。

【００６８】

なお、上記の実施形態及び実施例に係るコンピュータシステム及びそのＨＷ抽象化方法は、ハードウェア、ソフトウェア、又はこれらの組合せにより実現することができる。この場合の構成は特に限定されるものではなく、上述した各構成要素の機能を実現可能であれば、いずれのものでも適用可能である。例えば、上述した各構成要素の機能毎に回路や部品等を独立させて個別に構成したものでも、複数の機能を１つの回路や部品等に組み入れて一体的に構成したものでも、いずれでもよい。

【００６９】

上記の実施形態及び実施例の一部又は全部は、以下の付記のようにも記載されうるが、以下には限定されない。

【００７０】

（付記１）ＨＷ（ハードウェア）を仮想空間のＡＣＰＩツリー上に配列したオブジェクトとして抽象化するＡＣＰＩと、ＡＣＰＩを介してＨＷの構成を認識するＯＳと、ＡＣＰＩとＨＷとの間に設けられ、ＡＣＰＩツリー上のオブジェクトと同じ配列でＨＷ毎の情報を格納する場所を並べて構成されるメモリ領域を有する仮想ＨＷと、事前にＨＷを参照してＨＷ毎の情報を仮想ＨＷのメモリ領域内の当該ＨＷに対応する場所に格納するＢＩＯＳとを有し、ＡＣＰＩがＨＷを直接参照せずに仮想ＨＷのメモリ領域を参照してＨＷ毎の情報を得ることにより、ＯＳがＡＣＰＩを介してＨＷ毎の情報からＨＷの構成を認識することを特徴とするコンピュータシステム。

【００７１】

（付記２）仮想ＨＷのメモリ領域は、ＡＣＰＩツリー上に割り当てられるＡＣＰＩ番号順にＨＷの情報を格納する場所を並べて構成されることを特徴とする付記１に記載のコンピュータシステム。

【００７２】

（付記３）仮想ＨＷは、デバッグ用メモリ領域をさらに有し、ＡＣＰＩが仮想ＨＷのメモリ領域を参照する際、ＡＣＰＩ側のデバッグに必要なデータをデバッグ用メモリ領域に格納することを特徴とする付記１又は２に記載のコンピュータシステム。

【００７３】

（付記４）ＢＩＯＳは、ＨＷの初期化過程で当該ＨＷを参照してＨＷ毎の情報を仮想ＨＷのメモリ領域内の当該ＨＷに対応する場所に格納することを特徴とする付記１から３のいずれか１項に記載のコンピュータシステム。

【００７４】

（付記５）ＨＷの構成がＯＳの運用中に動的に変更になった場合、ＢＩＯＳが、ＨＷの構成変更を仮想ＨＷのメモリ領域内の当該ＨＷに対応する場所の格納されたＨＷ毎の情報に反映させることにより、ＯＳが仮想ＨＷを介してＨＷの構成変更を認識することを特徴とする付記１から４のいずれか１項に記載のコンピュータシステム。

【００７５】

（付記６）ＡＣＰＩが、ＨＷ（ハードウェア）を仮想空間のＡＣＰＩツリー上に配列したオブジェクトとして抽象化し、ＡＣＰＩとＨＷとの間に、ＡＣＰＩツリー上のオブジェクトと同じ配列でＨＷ毎の情報を格納する場所を並べて構成されるメモリ領域を有する仮想ＨＷを設け、ＢＩＯＳが、事前にＨＷを参照してＨＷ毎の情報を仮想ＨＷのメモリ領域内の当該ＨＷに対応する場所に格納し、ＡＣＰＩがＨＷを直接参照せずに仮想ＨＷのメモ

リ領域を参照してＨＷ毎の情報を得ることにより、ＯＳがＡＣＰＩを介してＨＷ毎の情報からＨＷの構成を認識することを特徴とするコンピュータシステムのＨＷ抽象化方法。

【００７６】

（付記７）仮想ＨＷのメモリ領域は、ＡＣＰＩツリー上に割り当てられるＡＣＰＩ番号順にＨＷの情報を格納する場所を並べて構成されることを特徴とする付記６に記載のコンピュータシステムのＨＷ抽象化方法。

【００７７】

（付記８）仮想ＨＷは、デバッグ用メモリ領域をさらに有し、ＡＣＰＩが仮想ＨＷのメモリ領域を参照する際、ＡＣＰＩ側のデバッグに必要なデータをデバッグ用メモリ領域に格納することを特徴とする付記６又は７に記載のコンピュータシステムのＨＷ抽象化方法。

【００７８】

（付記９）ＢＩＯＳが、ＨＷの初期化過程で当該ＨＷを参照してＨＷ毎の情報を仮想ＨＷのメモリ領域内の当該ＨＷに対応する場所に格納することを特徴とする付記６から８のいずれか１項に記載のコンピュータシステムのＨＷ抽象化方法。

【００７９】

（付記１０）ＨＷの構成がＯＳの運用中に動的に変更になった場合、ＢＩＯＳが、ＨＷの構成変更を仮想ＨＷのメモリ領域内の当該ＨＷに対応する場所の格納されたＨＷ毎の情報に反映させることにより、ＯＳが仮想ＨＷを介してＨＷの構成変更を認識することを特徴とする付記６から９のいずれか１項に記載のコンピュータシステムのＨＷ抽象化方法。

【００８０】

以上、実施の形態を参照して本願発明を説明したが、本願発明は上記実施の形態に限定されるものではない。本願発明の構成や詳細には、本願発明のスコープ内で当業者が理解し得る様々な変更をすることができる。

【産業上の利用可能性】

【００８１】

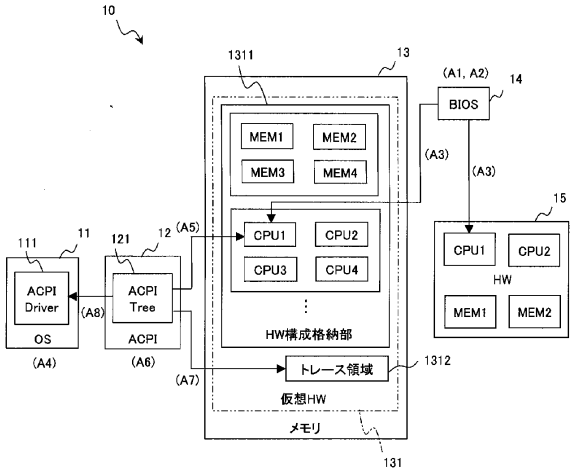
以上説明したように、本発明は、ＡＣＰＩによるＨＷ抽象化方式を用いた大規模マルチＯＳサーバなどのコンピュータシステムの用途に利用可能である。

【符号の説明】

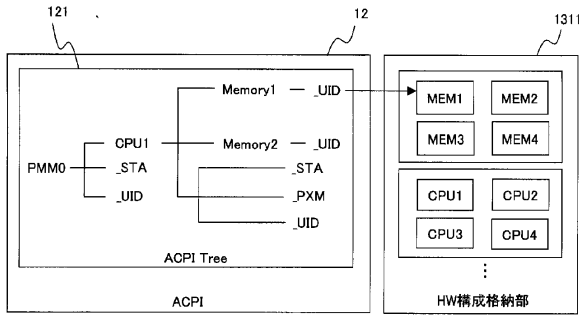
【００８２】

１０　コンピュータシステム
１１　ＯＳ
１２　ＡＣＰＩ
１３　メモリ
１４　ＢＩＯＳ
１５　ＨＷ（ハードウェア）
１１１　ＡＣＰＩドライバ
１２１　ＡＣＰＩツリー
１３１　仮想メモリ
１３１１　ＨＷ構成格納部
１３２２　トレース領域

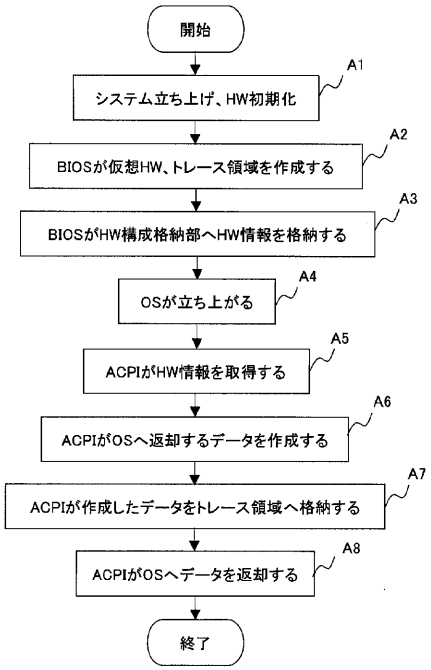
【図１】


10
1311
13
(A1, A2)
BIOS
14
MEM1
MEM2
MEM3
MEM4
(A3)
(A3)
15
CPU1
CPU2
CPU3
CPU4
HW
MEM1
MEM2
111
11
121
12
(A5)
ACPI Driver
(A8)
ACPI Tree
OS
ACPI
(A4)
(A6)
HW構成格納部
トレース領域
1312
(A7)
仮想HW
メモリ
131

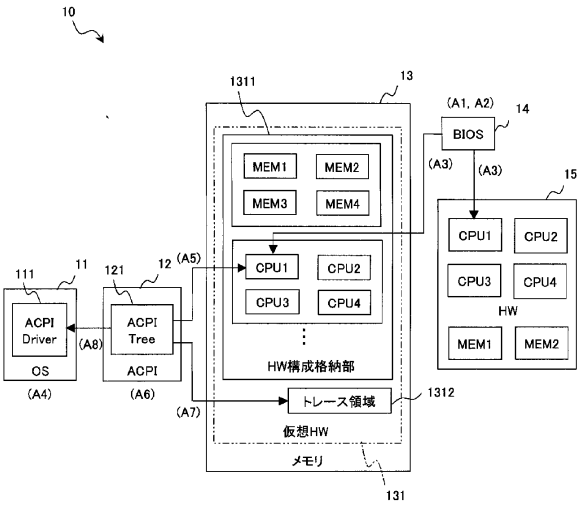
【図２】


121
12
1311
Memory1
_UID
MEM1
MEM2
MEM3
MEM4
PMM0
CPU1
_STA
_UID
Memory2
_UID
_STA
_PXM
_UID
CPU1
CPU2
CPU3
CPU4
ACPI Tree
ACPI
HW構成格納部

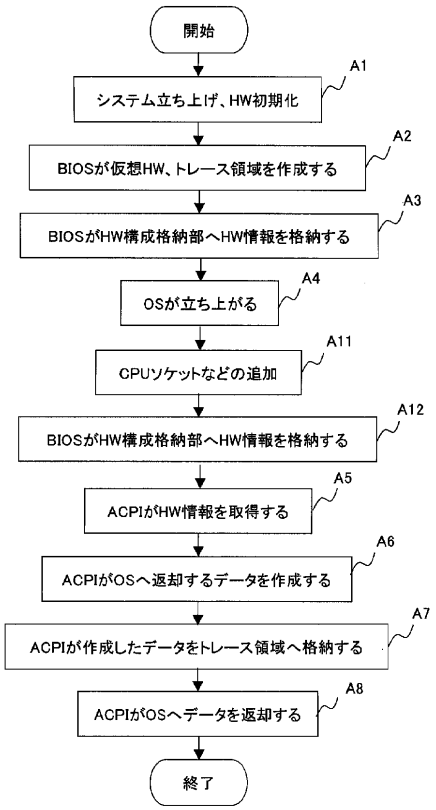
【図３】


開始
A1
システム立ち上げ、HW初期化
A2
BIOSが仮想HW、トレース領域を作成する
A3
BIOSがHW構成格納部へHW情報を格納する
A4
OSが立ち上がる
A5
ACPIがHW情報を取得する
A6
ACPIがOSへ返却するデータを作成する
A7
ACPIが作成したデータをトレース領域へ格納する
A8
ACPIがOSへデータを返却する
終了

【図４】


10
1311
13
(A1, A2)
BIOS
14
MEM1
MEM2
MEM3
MEM4
(A3)
(A3)
15
CPU1
CPU2
CPU3
CPU4
HW
MEM1
MEM2
111
11
121
12
(A5)
ACPI Driver
(A8)
ACPI Tree
OS
ACPI
(A4)
(A6)
HW構成格納部
トレース領域
1312
(A7)
仮想HW
メモリ
131

【図５】


開始
A1
システム立ち上げ、HW初期化
A2
BIOSが仮想HW、トレース領域を作成する
A3
BIOSがHW構成格納部へHW情報を格納する
A4
OSが立ち上がる
A11
CPUソケットなどの追加
A12
BIOSがHW構成格納部へHW情報を格納する
A5
ACPIがHW情報を取得する
A6
ACPIがOSへ返却するデータを作成する
A7
ACPIが作成したデータをトレース領域へ格納する
A8
ACPIがOSへデータを返却する
終了

フロントページの続き

(56)参考文献 特開２００４－０７０９５３（ＪＰ，Ａ）
特開２００４－０７０９５２（ＪＰ，Ａ）
特開２００４－０７０９４５（ＪＰ，Ａ）
特表２００９－５０８１８３（ＪＰ，Ａ）
特開２００６－３０９７５４（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　　９／４４５