仮想マシンと物理マシンを一元管理するための仮想 AMT

大 薗 弘 記† 光 来 健 一††

1. はじめに

PC があらゆる部署で使われるようになり，組織の管理者が管理しなければならない PC の数は膨大になってきている．管理者は障害が発生するたびに PC の設置場所まで行って修復作業を行わなければならない．このような管理者の負担を軽減するために，最近の PC には Active Management Technology（AMT）が搭載されるようになってきている．AMT を用いることで，管理者は PC をリモートから一元的に管理することができるようになる．しかし，近年，サーバ上で仮想マシン（VM）を動作させてその画面だけを PC に表示するという仮想デスクトップが普及してきており，組織内には物理マシンと VM が混在している．AMT は物理マシンの管理を行うことしかできないため，VM の管理は別に行う必要があった．

本研究では，物理マシンと VM を一元的に管理できるようにするために VM に対して仮想的な AMT を提供する仮想 AMT（vAMT）を提案する．

2. AMT のよる管理

AMT は，インテル社が提供する vPro の管理機能の核となる技術であり，PC をハードウェアレベルで管理することができる．AMT では PC の電源がオフの状態でもリモートからハードウェアとソフトウェアに関する情報の取得を可能にする．また，AMT は PC 上の OS に障害が発生した場合にリモートから回復させることができる．管理者は AMT を用いて PC の電源をリセットすることができるため，OS がクラッシュしてしまった時に再起動させることができる．さらに，AMT が通信の制御を行うことで，ウィルス等に感染した PC をネットワークから切り離すことができる．これにより，安全に修復作業を行ってから再度ネットワークに接続することができる．

AMT を用いることでリモートからの PC の管理が可能になるが，近年，普及してきている仮想デスクトップの管理に用いることはできない．仮想デスクトップはサーバの VM 上でシステムを動作させ，その画面だけを PC 上で表示する．現在，組織内では PC と仮想デスクトップが混在しているため，組織内の全マシンの管理を行う場合，物理マシンと VM の両方を管理する必要がある．しかし，AMT は物理マシンの管理を行うための技術であるため，VM の管理には別の管理ツールを用いる必要がある．そのため，管理者は少なくとも 2 種類の管理ツールを使い分けなければならず，管理に手間がかかる．

3. vAMT

本研究では，仮想的な AMT である vAMT を VM に対して提供する．vAMT は物理マシンを管理する AMT と同様のインタフェースで VM を管理することを可能にする．リモートの管理ツールは WS-Mamagement[1] というプロトコルを用いて vAMT にアクセスを行う．vAMT から情報を取得したり，管理を実行したりするために，システム管理の標準となっている Common Information Model (CIM) を AMT 用に拡張したインタフェースを用いる．AMT と vAMT を用いることで，物理マシンと VM の違いを意識することなく，既存の管理ツールを用いて一元的な管理を行うことができるようになる．

3.1 システム構成

vAMT は**図 1** のように，WS-Management サーバ，CIM オブジェクトマネージャ（CIMOM），リポジトリ，CIM プロバイダによって構成される．WS-Management サーバは送られてきたリクエストを CIM のリクエストに変換して CIMOM に渡す．CIMOM は CIM プロバイダの登録情報が格納されているリポジトリを参照して，リクエストを適切な CIM プロバイダに送る．CIM プロバイダでは，VM の仮想ハードウェアにアクセスすることで VM の情報を取得したり，管理を実行したりする．この実装には OpenPegasus を用いた．

† 九州工業大学
†† 九州工業大学，CREST

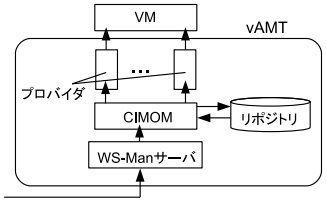

図 1 vAMT の構成

```
class Processor {
    [Key] uint32 Number;
    uint32 Enable([IN] boolean Enabled);
};
```

図 2 仮想 CPU の MOF の例

3.2 CIM プロバイダの作成

vAMT の CIM プロバイダの作成には，CIMPLE というツールを利用した．CIMPLE を用いることで，CIM が定義されている Managed Object Format (MOF) から CIM プロバイダの雛形を生成することができる．MOF には CIM のクラスが定義されており，管理対象の情報を格納するためのプロパティや管理対象に対して処理を行うためのメソッドから成る．**図 2** は仮想 CPU を管理するための MOF の例である．CPU 情報として CPU 番号（Number）が定義されており，仮想 CPU を区別することができる．また，引数の値によって仮想 CPU の有効化・無効化を行うための Enable メソッドも定義されている．

CIMPLE は C++のクラスとして CIM プロバイダを生成し，メンバ関数として enum_instances や get_instance などが定義されている．例えば，enum_instances にはその CIM クラスのすべてのインスタンスを返すように記述する．

3.3 CIM プロバイダの例

管理ツールは CIM_SoftwareIdentity を用いて AMT のバージョンを取得している．この CIM クラスには様々な情報を返すインスタンスが存在するが，その中の 1 つのインスタンスが AMT の情報を保持している．そこで，CIM プロバイダの get_instance メソッドにおいて，リクエストされた InstanceID プロパティの値が “AMT” の場合に vAMT のバージョンを返すようにした．

AMT を用いて電源のオン・オフを行うには，3 つの CIM プロバイダが必要となる．まず，VM の電源状態を取得する．次に電源操作を行う VM のシステム情報を取得し，最後に取得した情報を持つ VM の電源の操作を行う．VM の情報を取得したり，電源を操作したりするのには libvirt というライブラリを用いた．libvirt は様々な仮想化ソフトウェア上で動作している VM を統一的に扱うことを可能にしている．VM の電源状態を調べるには virDomainIsActive 関数を使用した．この関数は VM を指定して呼び出すことで，電源がオンの時に 1，オフの時に 0 を返す．また，VM の電源を入れる時に virDomainCreate 関数，電源を切る時に virDomainShutdown 関数を使用した．

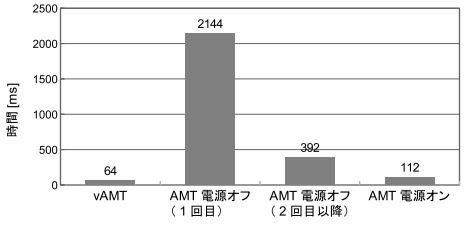

図 3 情報取得にかかる時間

4. 実　　験

管理ツールの WinRM を用いて，vAMT のバージョンを取得する get コマンドを vAMT と AMT に対して実行し，その処理時間を比較した．AMT の場合は電源がオンの状態とオフの状態で測定した．**図 3** に測定結果を示す．vAMT の処理時間は AMT よりも短いことが分かった．これは vAMT を搭載している PC 本体の CPU に比べて AMT の性能が低いためと考えられる．一方，AMT は電源がオフの場合，1 回目は非常に長い時間がかかるが，2 回目以降や電源がオンの場合には処理時間は改善されている．

5. ま と め

本研究では，VM を管理するための仮想的な AMT である vAMT を提案した．AMT と同様のインタフェースを vAMT にも持たせることによって，既存の管理ツールを用いて物理マシンと VM を一元的に管理することができる．OpenPegasus を用いて vAMT のシステムを構築し，VM を管理するためのいくつかの CIM プロバイダを作成した．今後の課題は，AMT を用いて管理を行うために必要となるすべての CIM プロバイダを実装し，既存の管理ツールを用いて様々な VM の管理を行えるようにすることである．

参 考 文 献

1) DMTF. Web Services for Management (WS-Management) Specification. 2010.