

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 園 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、ウイルス等に感染した 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ない。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-Management¹⁾ 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

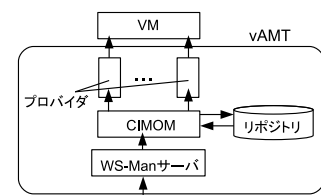


図1 vAMTの構成

```
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

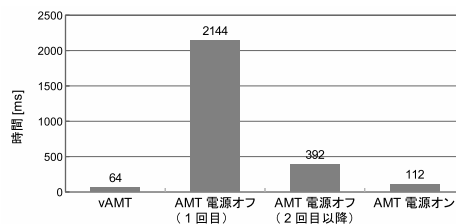


図3 情報取得にかかる時間

作したりするには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。

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の性能が低いとみられる。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.