仮想化技術を用いたサーバ統合環境における リソース浪費の検出と遮断

岡本 慶大† 河合 栄治† 砂原 秀樹†‡

†奈良先端科学技術大学院大学 情報科学研究科
‡慶応義塾大学大学院 メディアデザイン研究科

1 はじめに

サーバ統合の手段として，仮想計算機 (VM) 技術の利用が加速している．VM を用いたサーバ統合環境では，統計多重効果によって計算リソースの利用効率が向上し，管理運用の一層の効率化が望める．また，時間的な負荷の偏りを利用して多重化することができれば，より利用効率が向上し，電力消費の削減も見込める．

しかし，これら技術によって共有されるリソースは攻撃の対象にもなり得る．悪意を持った VM が共有リソースを浪費・占有することで，同じ共有リソースを利用している他の VM に十分なリソースが割り当てられず，VM 上で実行されているサービスレベルに影響を及ぼす可能性がある．

そこで本研究では，悪意を持った VM がリソースを浪費する攻撃を行った場合も，他の VM への影響を最小化するシステムの提案と実装を行う．

2 VM 環境におけるリソース浪費型 DoS 攻撃

VM 環境では複数の VM によってリソースが共有される．従来 DoS 攻撃はネットワーク越しに大量のパケットを送出する，もしくは OS の脆弱性を突くパケットを送出する手法が一般的であったが，VM 環境では以下のような攻撃が成立すると考えられる．

- CPU 処理サイクルの浪費・競合
- 過度なメモリの浪費・競合
- 過度なハードディスクの浪費・競合
- 過度なネットワークの浪費・競合

3 リソース浪費検出システムの提案

以上のような問題点を解消すべく，VMM から VM の挙動を観測し，リソース浪費が認められた場合は該当する VM へのリソース配分を絞る手法のシステムを提案する．本研究の想定条件は以下の通りである．

- HostOS 上で VMM が管理する各種リソースの状態取得が可能である
- リソースが HostOS から制御可能である
- データセンター等，HostOS と GuestOS の管理者が異なる

GuestOS への実装を必要とした場合，GuestOS 毎の異なる実装が必要となったり，GuestOS 上の他のプログラムとの競合を起こしてしまう場合がある．そこで本研究では，GuestOS が利用している物理 CPU のパフォーマンスカウンタを監視することで，GuestOS への実装無しに VMM から VM の状態監視を行う手法を利用した．

監視には Xenperf [1] を利用した．Xenperf は Domain0 から全物理 CPU のパフォーマンスカウンタ状態を取得できる Xen 専用のツールである．Xen の Domain をそれぞれ物理 CPU1 つと紐付けをし，Xenperf により一定間隔でカウンタ情報を取得しながら，各種の攻撃を行い，攻撃毎のカウンタの変化を追った．

4 設計と実装

実装はリソース浪費を定義する Policy と，実際のリソース制御を行う Judicator の 2 つで行った．Judicator と Policy の動作モデルを図 1 に示す．

4.1 Policy

Policy は Xenperf によって得られた情報を元に，リソース浪費を定義する．具体的には，Xenperf で

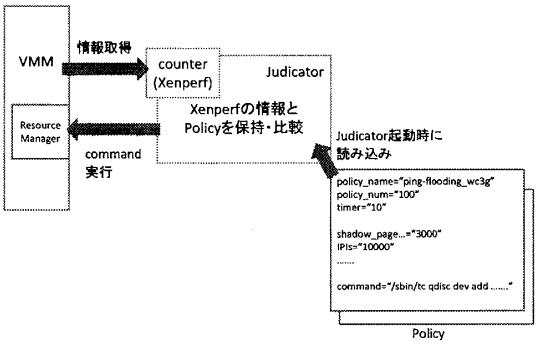

図 1　Judicator と Policy の動作

得られた各カウンタの値が，一定時間にどれだけ以上増加した場合にリソース浪費とみなすか，の閾値を記述する．また，リソース浪費と判断された場合に実行する処理の記述も Policy に行う．

4.2　Judicator

定義された Policy に基づいてリソース浪費の判断を行うプログラムが Judicator である．Judicator は，一定間隔でパフォーマンスカウンタの情報を取得し，Policy との比較を行う．Policy に設定された閾値を超えた状態が一定時間続くと，Policy に記述されたコマンドを実行する．

5　実験

実験は Intel Xeon 5160 の Dual CPU (合計 4 コア)，メモリ 4GB，VMM は Xen 3.3.0 の環境を使用した．3 つの VM にそれぞれ 1GB，Domain0 には 880MB を割り当て，各 VM に物理 CPU を 1 つずつ紐付けした．OS は Domain0/HVM Domain 共に Linux 2.6.18 (x86_64) である．

5.1　攻撃検出

Policy の例として，VM から Ping Flooding を行った場合のモニタリングを行ったものを採用した．Ping Flooding が行われている場合，非常に大量のパケットが送受信されるので，通常時に比べ，割り込み，Context-Switch が大量に発生すること，逆にメモリなどへの操作は少なく，Shadow Page Table 関係の値にはあまり変化が見られないことが特徴として挙げられる．

攻撃検出後は，tc コマンドによって帯域制限を行った．その結果，当該 VM の CPU 占有率，パケット送出レートを大幅に抑えることができた．当該 VM から Ping Flooding によって攻撃を受けていた VM の CPU 占有率も低下したため，この攻撃に対しての Policy の適用は成功したと結論づけられる．

表 1　Judicator によるオーバヘッド

スリープ時間 (sec)	CPU 占有率 (%)
0.1	0.43
0.5	0.09
1.0	0.06
5.0	0.03

5.2　オーバーヘッド

現段階の実装では，一定時間毎に Xenperf で読み込んだ各 CPU 毎のパフォーマンスカウンタの情報を配列に格納し，同じく配列に格納した Policy の情報と比較を行っている．情報取得頻度は nanosleep() によって制御を行う．スリープの時間と Judicator プロセスの CPU 占有率の関係を表 1 に示す．

表から読み取れるとおり，オーバヘッドは比較的低く抑えられていると結論づけられる．ただし，今回の実験は単一の Policy を使用したため，今後複数の Policy を同時に適用する場合は比較演算の回数が増大することが予想される．

6　まとめ

本研究では，以上のような問題点を解消すべく，VMM から VM の挙動を観測し，リソース浪費が認められた場合は該当する VM へのリソース配分を絞ることで，リソース浪費による他の VM のサービスレベル悪化を最小限に抑えるシステムを提案した．テストケースとして Ping Flooding を用いた VM 間での攻撃を想定，提案システムで検出が行われ，仮想ネットワークアダプタに対して帯域制限を行うことで，攻撃の影響を抑えられたことを確認できた．今後の展開として，Xen 以外でのプラットフォームへの適用，Policy 生成の自動化やコストの削減などが挙げられる．

参考文献

[1] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *the 19th ACM Symposium on Operating Systems Principles (SOSP '03)*, December 2003.