仮想計算機におけるアウトソーシングのためのゲスト–ホスト間RPC

齊藤剛[†1] 新城靖[†2] 榮樂英樹[†2]
佐藤聡[†2] 中井央[†3] 板野肯三[†2]

1. はじめに

現在，準仮想化においては主にディスクアクセスやネットワークアクセスなどのI/Oについて，専用のデバイスドライバを組み込むことで，VMMを通してホストOSに要求を送るという方法が取られている．この方法では，ホストOSは抽象度の低いデバイスのレベルでしかゲストOSの要求を知ることができない．

本研究では，ゲストOSからホストOSへ要求を送る方法として，アウトソーシング[1)]と呼ばれるより高いレベルでの準仮想化を行う．ゲストOSからの要求を今までのデバイスのレベルよりも抽象度の高い段階でホストOSに伝えることで，ホスト側でゲスト側の要求をより柔軟に処理することを可能とする．例えば，ゲスト側の通信要求をホスト側でSSLなどにのせて安全な通信路を通して送信することを可能にする．本稿では，ゲスト–ホスト間RPCとそれを用いたソケットアウトソーシングについて述べる．

2. ゲスト–ホスト間RPCとイベントキュー

ソケットオブジェクトにおいてアウトソーシングを実現するために，仮想計算機のゲスト–ホスト間でRPCを行う機構を提供する．これはホストOS側のVMMにサーバを追加し，その機能をゲスト側で動作するクライアントから呼び出すための枠組みで，VM RPCと呼ぶ．

VM RPCのインターフェイスを**表1**, **表2**に示す．VM RPCでは，ホストOS側のサーバを機能毎にモジュールとして追加することができる．ホストモジュールはカーネル空間で動作するものとユーザ空間で動作するもの両方をサポートし，`register_hm()`を呼ぶことで登録することができる．登録したホストモジュールはゲストOS側で名前とバージョン番号を指定して`bind_module()`を呼ぶことでバインドし，ハンドルを取得できる．このハンドルに対して呼び出したい機能番号と引数を指定して`perform_rpc()`を呼ぶことで，ホストOS側サーバの機能を呼び出すことができる．`perform_rpc()`は呼び出すとブロックし，ホスト側で処理が終わるとその結果が返り値となるため，ゲストOS側からはホストOSの機能を通常の関数呼び出しのように扱うことができる．またゲストOSとホストOSの間で，`make_ev_queue()`によって相互に出し入れ可能なイベントキューを作成できる．このキューは共有メモリの上に作成され，ゲスト-ホスト間の状態遷移なしに読み書きすることができる．

†1 筑波大学第三学群情報学類
†2 筑波大学システム情報工学研究科
†3 筑波大学図書館情報メディア研究科

3. アウトソーシングによるネットワークアクセス

ソケットアウトソーシングによるゲストOSからのネットワーク入出力について述べる．一般的な準仮想化の場合，VMMの機能を利用しホストOSに処理要求を伝えるだけの特別なデバイスドライバの組み込みを行う．これにより**図1**の左側のように，ゲストOSからホストOSのバックエンドドライバに直接接続することで余計な処理を省いて高速化できるという利点がある．

これに対してソケットアウトソーシングでは，図1の右のようにソケットオブジェクト，すなわちプロトコルスタックでの準仮想化の考え方を導入する．ゲストOS上の置き換えられたプロトコルスタックで

表1 VM RPCのホスト側インターフェイス

名前	説明
register_hm	ホストモジュールを登録する
unregister_hm	ホストモジュールを登録解除する
make_ev_queue	イベントキューを作る
del_ev_queue	イベントキューを破棄する
put_event	イベントキューにイベントを入れる
get_event	イベントキューからイベントを取り出す

表2 VM RPCのゲスト側インターフェイス

名前	説明
bind_module	ホストモジュールをバインドする
unbind_module	ホストモジュールを解放する
perform_rpc	ホストモジュールの関数を呼ぶ
put_event	イベントキューにイベントを入れる
get_event	イベントキューからイベントを取り出す

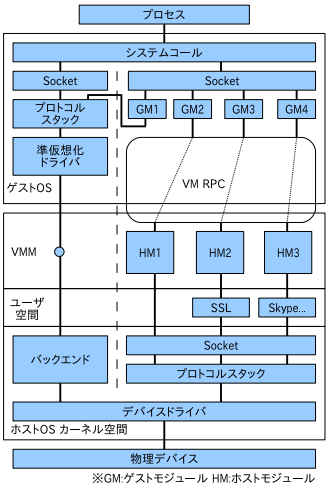

図 1 ネットワークアクセスの流れ

は実際の処理を行わず，かわりに `bind` や `connect`，`getsockopt` などのソケットインターフェイスを備えたホストモジュールを導入し，これを通して VM RPC によってホスト OS 上のプロトコルスタックに通信処理を行わせる．

ソケットレベルでのアウトソーシングでは，ホストモジュールはゲストから通信に関する情報を VM RPC により高レベルで受け取るため，ゲストの要求どおりに通常のソケットの API により外部と通信するだけでなく，より柔軟な通信の制御を容易に行えるという利点がある．例として，ゲスト OS からの特定の通信に関してはホスト OS 側で SSL に乗せることで安全に通信を行うことが考えられる．アウトソーシングによって接続先など通信に関する様々なパラメータを容易に知ることが出来るので，それらの組み合わせが事前に記述しておいたルールにマッチしたときは重要な通信と判断し，強制的に暗号化して機密を確保することができる．暗号化の他にも，通信をそのままネットワークに流さずに，例えば Skype のオーバレイネットワークに乗せるなど，様々な操作を行うことが考えられる．

本研究ではゲストのソケットオブジェクトとして，要求された操作やパラメータに応じてゲスト側のモジュールとホスト側のモジュールの対応付けの方法を切り替えるものを用意する．このオブジェクトを利用して，実際の処理が確定したときにモジュールの対応付けを行う動的特化の手法を用いることで，効率的に処理を行うことができるようにする．例えば，通信相手がゲスト OS 内のプロセスだとわかった段階でソケットオブジェクトをゲスト OS 内で通信を完結させるゲストモジュールと対応付けることで，ホスト OS 側で実際に使われないソケットオブジェクトを作ったりするなどの余計な処理を省くことができる．

4. 関 連 研 究

Linux KVM では低レベルでの準仮想化を支援するために，Virtio[2] と呼ばれる仕組みを提供している．これは，ゲストの IO をエミュレートなしでホストに伝えるキューであり，これを用いてドライバを書くことでドライバレベルでの準仮想化が実現できる．これに対し，本研究では VM RPC による手続き呼び出しを利用して，より抽象度の高いソケットオブジェクトのレベルでの準仮想化を実現する．

ゲスト OS からの通信をホスト OS 上で操作する方法として，VMM 内にプロトコルスタックを持たせてゲスト OS からの IP パケットを解析する方法[3] がある．これによってゲスト OS に手を加えることなくゲストの通信を操作することができる．これに対して本研究では，ゲスト OS を変更してプロトコルスタックを省くことで，パケット解析を不要にしている．

通信を外部から暗号化する方法として，カーネルレベルで SSL をサポートする方法[4] がある．これによって，アプリケーションに意識させることなく安全な通信を行うことができるようになる．本研究では，ホスト OS のユーザプロセスで SSL による暗号化を行う．

5. お わ り に

本論文では，モジュールにより機能追加可能なゲスト-ホスト間 RPC と，それを用いたソケットアウトソーシングについて述べた．アウトソーシングにより，ホスト OS の機能を使ってゲスト OS の機能を拡張することができる．今後は，通信の暗号化などのモジュールを実装し，性能を評価する．

参 考 文 献

1) Eiraku, H., Shinjo, Y., Pu, C., Koh, Y. and Kato, K.: Fast Networking with Socket-Outsourcing in Hosted Virtual Machine Environments, *24th ACM Symposium on Applied Computing (SAC2009)* (2009). To appear.
2) Russell, R.: Virtio: Towards a de-facto standard for virtual I/O devices, *SIGOPS Oper. Syst. Rev.*, Vol.42, No.5, pp.95–103 (2008).
3) 榮樂英樹， 新城靖, 加藤和彦：ユーザレベル OS のためのユーザレベルネットワーク機能，第 3 回情報科学技術フォーラム (FIT2004), 9 月 (2004).
4) 光来健一， 千葉滋：インターネットにおけるパーソナルネットワークの構築，情報処理学会研究報告. システムソフトウェアとオペレーティング・システム， Vol.2001, No.78, pp.83–90 (2001).