# 協調型仮想計算機モニタのためのシリアルポートを用いたOS間通信

Inter-OS Communication through Serial Port for Cooperative Virtual Machine Monitors

基盤ソフトウェア学講座　0312011147　武藤 寛

指導教員： 杉野 栄二　瀬川 典久　澤本 潤

## 1. はじめに

仮想計算機モニタ (VMM, Virtual Machine Monitor) は, OS 上で別の OS を動作させるソフトウェアである．ここで, 前者をホスト OS, 後者をゲスト OS と呼ぶ．従来の VMM ではホスト OS とゲスト OS は相互に干渉せず動作する．これに対して，協調型仮想計算機モニタ (CoVMM, Cooperative VMM)[1] はゲスト OS とホスト OS が協調して動作する VMM である．

協調型仮想計算機モニタの実装には，ゲスト OS におけるカーネル・プログラミングが必要である．これまでにゲスト OS として Linux を用いた実装がある．ゲスト OS に他の OS が使えるならば，より多様な協調利用が期待できる．そこで本研究は，ゲスト OS として Windows を想定した協調型仮想計算機モニタの実装を目的とする．この目的のために，シリアルポートを用いて OS 間通信機能を実現し，その通信性能を評価する．シリアルポートによる通信機能は多くの OS に備わっており，OS 間通信機能の協調型仮想計算機モニタの実現にとって根幹となる．

## 2. 協調型仮想計算機モニタにおける OS 間通信

従来型 VMM では，ホスト OS からゲスト OS のプログラムを利用する際には，ネットワーク通信が必要であった．個人で複数の VM を管理している場合，ゲスト OS ごとにユーザアカウントが存在し，管理が難しくなる，

協調型仮想計算機モニタでは，ネットワーク処理を介さずに OS 間で通信を行い, ユーザ認証の問題を回避する．それにより，ユーザはホスト OS のユーザ権限でゲスト OS の機能を利用することが実現する．また，ホスト・ゲスト OS 間で相互にシステムコールやライブラリ等の機能を利用することも可能となる．そのため，協調型仮想計算機モニタにおける OS 間通信では，ネットワーク処理を介さない安全な通信が重要である．

## 3. 先行研究

白石氏らは Linux KVM(Kernel based Virtual Machine) をベースとして，ホスト OS, ゲスト OS を共に Linux を用いて協調型仮想計算機モニタを実装した．OS 間通信の実装には VMRPC[2]，とソケットアウトソーシングを用いた．

VMRPC とはゲスト OS からホスト OS の機能を RPC(Remote Procedure Call) 形式で呼び出す機構である．VMRPC の引数はホスト・ゲスト OS 間の共有メモリを通じて受け渡される．

ソケットアウトソーシングとはソケット層のモジュールを置き換えることで，ホスト OS の機能を利用する技術である．ゲスト OS はホスト OS の UNIX Domain Socket を利用することで, ホスト OS のファイル空間の参照を可能にした．Unix Domain Socket で作成した通信路を用いることで，ネットワーク処理を不要にし，安全な OS 間通信を実現した．

本研究ではホスト OS として Linux，ゲスト OS として Windows を想定する．ゲスト OS が Windows の場合，VMRPC を実装するにはカーネル内のモジュールが必要となる．また，Windows では文献 2 で利用されている UNIX Domain Socket を利用することは難しい．

## 4. 設計

本研究では安全な OS 間通信が必要であるが，ゲスト OS としてクローズド OS を想定しており，ゲスト OS のカーネルに対して改変を加えることができない．性質の異なる OS 間で使えるネットワーク処理を介さない通信路として，シリアルポートがある．シリアルポートは古い通信規格ではあるが，多くの OS が対応している．そこで，本研究ではシリアルポートを用いて OS 間通信の実現を提案する．シリアルポートは多くの OS でデバイスドライバが用意されているので，カーネル・プログラミングが不要となり，より簡単に OS 間通信が実現できる．

図 1 に全体の動作の流れを示す．

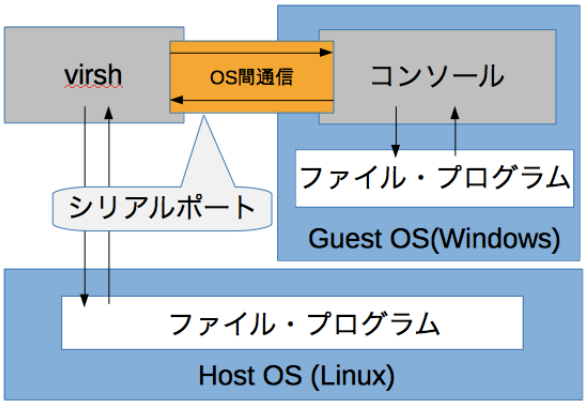


図 1 **動作の流れ**

## 5. 実装

VMM には KVM を用いた．KVM には VM を管理機能を持つ virsh が提供されており，コマンドによる VM の管理操作ができる．virsh は libvirt[3] API を利用して実装されている．libvirt API は VM の管理操作のために抽象化された C 言語の API を提供する libvirt ライブラリであり，それを利用して機能の追加も可能である．そこで，本研究では virsh にホスト OS からゲスト OS に対してデータのコピーを行う独自のコマンド com-dd を実装した．

以下が com-dd の仕様である．第一引数に接続するゲスト OS．第二引数にホスト OS のコピーの元のファイル．第三引数にゲスト OS のコピー先のパスを指定する．

```
com-dd <Guest Domain> <In File> <Out Path>
```

ホスト OS 側では，引数で指定されたデータを読み取り，libvirt API を用いてゲスト OS のシリアルポートに対して送信処理を行う．また，送信処理の際に，開始から終了までの時間を計測して転送速度を計測する．

ゲスト OS 側では，Win32 API を用いてホスト OS からの送受信処理とリクエストの実行を行うコンソールアプリケーションを実装した．

## 6. 実験

com-dd コマンドを使用して転送速度を計測した．以下に実験環境を示す．

- ホスト OS
  - OS：Linux 3.17.7
  - CPU：Intel Core i7-2700K 3.50GHz
  - メモリ：12GB
- ゲスト OS
  - OS：Windows 7 Professional
  - メモリ：4GB

転送用に 1M バイトのダミーファイルを作成し，ゲスト OS の NULL デバイスに転送した．これを 10 回繰り返し，平均を求めた．平均速度は 51770 バイト/秒であった．

先行研究の OS 間通信の転送速度は 1.1G バイト/秒であり，本 OS 間通信は極めて低速である．シリアルポートのデバイスドライバでは，通信バッファのサイズに制限があり，それに合わせて送受信処理を行うので，処理のループが増えてしまった．それが転送速度に影響していると考えられる．また，先行研究のソケットアウトソーシングは VMM によるエミュレーションを行わずに実機が処理を行うことで I/O を高速化する手法であるため大きく差が開いた．

## 7. 終わりに

本研究ではシリアルポートを用いた OS 間通信の実現を提案した．OS 間通信を用いるアプリケーションとして，virsh に対して com-dd コマンドを実装した．また，実装した OS 間通信の転送速度を計測する実験を行った．本 OS 間通信を高速化するためには送受信処理の改良が課題である．

## 参考文献

1) 白石 光隆, 新城 靖, 齊藤 剛, 豊岡 拓, 五明 将幸: 協調型仮想計算機モニタとその Linux KVM における実装, 情報処理学会論文誌 コンピューティングシステム Vol. 3 No. 2 147− 162 (June 2010).

2) Hideki Eiraku, Yasushi Shinjo, Calton Pu, Younggyun Koh, Kazuhiko Kato: ”Fast Networking with Socket-Outsourcing in Hosted Virtual Machine Environments”, ACM Symposium on Applied Computing (SAC2009), pp.310-317 (2009).

3) libvirt:http://libvirt.org/index.html