2K-02

協調型仮想計算機モニタのための シリアル接続を用いたOS間通信

武藤寛† 杉野栄二† 瀬川典久† 澤本 潤† 新城 靖‡
岩手県立大学† 筑波大学‡

1. はじめに

Type II VMM (Virtual Machine Monitor) を利用して，起動中の OS (Operating System) とは別の OS のアプリケーションを動作させることができる．従来の VMM では実機 (以下，ホスト OS と呼ぶ)と VM (Virtual Machine．以下，VM 上で実行する OS をゲスト OS と呼ぶ) は相互干渉せずに動作することを目的とする．これに対し，協調型仮想計算機モニタ(Cooperative VMM)は独立して動作するのではなく，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，安全な OS 間通信が必要であるが，それにはゲスト OS におけるカーネル・プログラミングが必要となる．

そこで本研究では多くの OS でサポートされるシリアルポートを用いてゲスト OS におけるカーネル・プログラミングを不要にし，ホスト OS からゲスト OS をより簡単に利用可能にする．

2. 協調型仮想計算機モニタ

従来型 VMM は隔離された VM の管理により，利用者が複数の場合でも相互干渉することなく VM を利用できた．しかし，個人の利用者が全ての VM を管理する場合では隔離された管理が不要である．また，ホスト OS からゲスト OS のプログラムを利用する際には，ユーザ認証を求められる．複数の VM を管理している場合，VM ごとにユーザアカウントが必要となり管理が難しくなる．

協調型仮想計算機モニタでは，その隔離性を緩和して，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可能にすることで問題を解決する．そのためには複数の OS の機能を同時に利用するようなホスト・ゲスト OS 間で協調した動作が必要になる．

Inter-OS Communication through Serial Port for Cooperative Virtual Machine Monitors
Hiroshi Muto† Eiji Sugino† Norihisa Segawa† Jun Sawamoto†
Iwate Prefectural University†
Yasushi Shinjo‡
University of Tsukuba‡

3. 先行研究

先行研究[1]では 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．また，ホスト・ゲスト OS の協調動作の例として，ホスト OS とゲスト OS のプログラムをコマンドレベルで利用可能にする多重 OS シェルを実装した．

ホスト・ゲスト OS 間でネットワーク処理を介さずに行う OS 間通信を実装した．実装には VMRPC[2]，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．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．

4. 本研究の目的

VMM を利用する目的として，ホスト OS ではサポートされてないアプリケーションを利用したい場合に，そのアプリケーションをサポートしている OS を用意して VM 上で動作させることが挙げられる．協調型仮想計算機モニタも同様であり，OS の組み合わせに限定されずに利用できることが望ましい．本研究ではホスト OS を Linux，ゲスト OS を Windows とした場合の協調型仮想計算機モニタの実装を目的とした．

先行研究の 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．ゲスト OS が Windows の場合，カーネル内のモジュールが必要となる．また，文献 2 で利用されている Unix Domain Socket は，Windows では使用することが難しい．

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．また、シリアルポートは多くの 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 間通信を実現することを提案する．本 OS 間通信は，表 1 に示す API を提供す

Copyright ©2015 Information Processing Society of Japan.
All Rights Reserved.

る．この API は，各ゲスト OS でシリアルポートをアクセスすることで実装する．

virDomainOpenConsole()	ゲスト OS コンソールに接続
virStreamNew()	ストリームを作成
virStreamSend()	ストリームへデータ送信
virStreamRecv()	ストリームからデータ受信
virStreamFree()	ストリームを解放

表 1 OS 間通信の libvirt API

5. 実装

VMM には KVM を用いた．KVM には コマンドにより VM を管理する virsh が提供されている．virsh は C 言語のライブラリである libvirt[3] API による実装されている．この API を用いることによりユーザレベルで機能を拡張できる．本研究ではそれを利用して virsh に独自のコマンド com-dd，com-run を実装した．

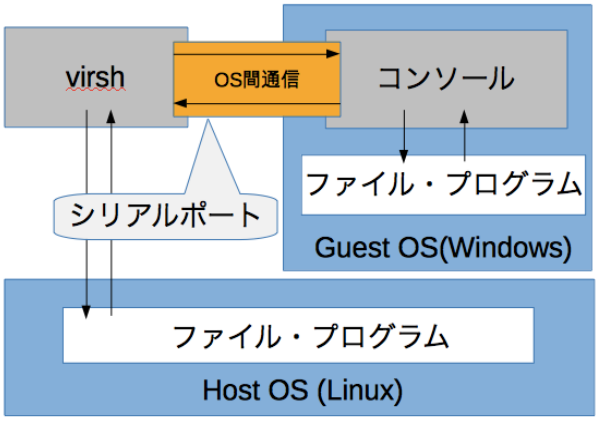

図 1 動作の流れ

図 1 に全体の動作の流れを表す．本研究ではホスト OS で動作する virsh とゲスト OS のコンソール間で行う OS 間通信にシリアルポートを用いている．これを実現するために，ゲスト OS 側ではホスト OS のリクエストに対して処理を行うコンソールアプリケーションを実装した．virsh では表 1 に示した libvirt API の virDomainOpenConsole()を用いてゲスト OS のシリアルポートにアクセスする．次に，virStreamNew()によりストリームを作成する．このストリームを通して OS 間でデータの送受信を行う．

ゲストコンソールでは Win32API によりシリアルポートの送受信処理とリクエストに応じた機能を実装した．

以下に実装したコマンドの仕様を示す．com-dd はホスト OS からゲスト OS に対してデータのコピーを行うコマンドである．

```
com-dd <Guest Domain> <Host Src> <Guset Dst>
```

そして，com-run はゲスト側のプログラムを実行するコマンドである．

```
com-run <Guest Domain> <Program> <Args>
```

これらの実装にあたり，ホスト OS では virsh に 270 行のソースコードを追加して，ゲスト OS ではコンソールアプリケーションを 260 行のソースコードにより作成した．

6. 実験

com-run のホスト OS の実行要求からゲスト OS でプログラムが実行されるまでの遅延時間を計測した．以下に実験環境を示す．

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

com-run を 10 回繰り返し，その平均遅延時間を求めた．平均遅延時間は 0.76 秒であった．

この遅延時間は実用に耐えうる程度であると考える．また，シリアルポートを用いることで，ネットワーク処理のセキュリティ問題を回避でき，安全にプログラムの実行が可能である．

7. 終わりに

本研究ではシリアルポートを用いた OS 間通信を提案した．OS 間通信を行うアプリケーションとして，virsh に対して com-dd と com-run コマンドを実装した．これらは libvirt API を用いることでユーザレベルのプログラミングで実現できる．

シリアルポートを用いる利点として多くの OS がサポートしている点がある．今後は Windows 以外の OS として Linux をゲスト OS として実装を行い，提案手法のポータビリティについて評価を行う．

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 (June2010).

[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

Copyright ©2015 Information Processing Society of Japan.
All Rights Reserved.