シャドウデバイスを用いた帯域外リモート管理に対応したVMマイグレーション

鵜木 智矢[1]　二神 翔太[1]　光来 健一[1]

概要：IaaS 型クラウドが提供する帯域外リモート管理にはクラウドの管理者への情報漏洩の危険性があるため，その対策として VSBypass と呼ばれるシステムが提案されている．VSBypass では，信頼できない管理者が管理権限を持つ仮想化システムの外側でシャドウデバイスを動作させ，仮想マシン（VM）に対する帯域外リモート管理の入出力を安全に処理する．しかし，VM をマイグレーションするとシャドウデバイスの状態が失われ，シャドウデバイスを用いた帯域外リモート管理が行えなくなる．そこで本稿では，シャドウデバイスの状態を VM と同時に移送先に転送することにより，VM マイグレーション後もシャドウデバイスを用いた帯域外リモート管理を可能にするシステム USShadow を提案する．USShadow では，仮想化システム内の移送マネージャが疑似デバイスを通してシャドウデバイスと高速に通信し，その状態を透過的に保存・復元する．また，シャドウデバイスにおいて状態を暗号化・復号化することにより，マイグレーション中の情報漏洩を防ぐ．我々は，仮想シリアルコンソールと VNC を用いた帯域外リモート管理に対応した USShadow を Xen に実装し，マイグレーションのオーバヘッド増加が小さいことを確認した．

1. はじめに

IaaS 型クラウドでは，ユーザは自らの用途に合わせて必要な数の仮想マシン（VM）を用意し，システムを構築することができる．ユーザは SSH や VNC といったリモート管理ソフトウェアにより，クラウドから提供される VM の管理を行う．その際に，VM に直接接続してアクセスを行う管理手法の他に，クラウドが提供する帯域外リモート管理と呼ばれる管理手法を利用することもできる．帯域外リモート管理では，ユーザが VM の仮想シリアルデバイスや仮想キーボード，仮想マウス，仮想ビデオカードといった仮想デバイスに直接アクセスすることで VM の管理を行うことができる．この管理手法は VM 内のリモート管理の設定に依存しないため，VM 内でネットワーク障害が発生した場合においてもユーザは VM の管理を行うことができる．

帯域外リモート管理においてユーザがアクセスする仮想デバイスはクラウドの管理者によって管理されている．その一方で，クラウドの管理者が全て信頼できるとは限らず，中には悪意あるクラウドの管理者も存在する．実際に，管理者の約 35 %が機密情報に無断でアクセスしたことがあり [1]，サイバー犯罪の約 28 %が内部犯行である [2] という報告もある．クラウドの管理者が悪意を持っていた場合，ユーザが仮想デバイスを介して VM に行う入出力の情報を容易に盗聴することができる．

そこで，仮想デバイスからクラウドの管理者への情報漏洩を防ぐために VSBypass [3] と呼ばれるシステムが提案されている．VSBypass では，ネストした仮想化を用いて，従来の信頼できない仮想化システムを VM の中で動作させる．そして，帯域外リモート管理に用いられる仮想デバイスへのアクセスを横取りして仮想化システムの外側で動作するシャドウデバイスで処理し，安全に帯域外リモート管理を行う．このように，帯域外リモート管理の入出力はすべて仮想化システムの外側で処理されるため，VSBypass は仮想デバイスからクラウドの管理者への情報漏洩を防ぐことができる．

しかし，VSBypass では VM マイグレーションの際にシャドウデバイスの状態が失われるため，マイグレーション後にシャドウデバイスを用いた帯域外リモート管理を行うことができない．シャドウデバイスは仮想化システムの外側で動作するため，仮想化システム内でマイグレーションを行う移送マネージャがシャドウデバイスの状態を保存・復元できないためである．また，ユーザ VM 内の VRAM にもアクセスすることができなくなり，VNC を用いたリモート管理で必要な画面更新の検出も行えなくなる．

そこで本稿では，シャドウデバイスの状態をユーザの VM

[1] 九州工業大学
Kyushu Institute of Technology

と合わせて転送できるようにすることで，マイグレーション後もシャドウデバイスを用いた帯域外リモート管理を行えるシステム USShadow を提案する．USShadow では，マイグレーションの際に仮想化システム内の移送マネージャが仮想化システム外部のシャドウデバイスから状態を取得し転送する．そして，マイグレーション後には移送先ホスト内の移送マネージャがシャドウデバイスに対して，転送されてきた状態の復元を行う．移送マネージャへの改変を不要にするために，仮想化システム内で動作させる疑似デバイス経由で透過的にシャドウデバイスの状態の保存・復元を行う．疑似デバイスはウルトラコール [4] と呼ばれる機構を用いてシャドウデバイスと通信を行う．シャドウデバイスの状態からの情報漏洩を防ぐために，USShadow では移送元のシャドウデバイスにおいて状態の暗号化を行い，移送先のシャドウデバイスで状態の復元を行う．

我々は USShadow を Xen 4.8.0 をベースとして開発された VSBypass に実装した．VM 内で動作させる仮想化システムが Xen の場合，シャドウデバイスとして仮想シリアルデバイスと仮想キーボード，仮想マウス，仮想ビデオカードに対応しており，KVM の場合は仮想ビデオカード以外に対応している．我々は，USShadow を用いてマイグレーション性能を測定する実験を行い，マイグレーションにかかる時間とダウンタイムの増加はネストした仮想化を用いる従来システムおよび VSBypass に比べて小さいことが分かった．

以下，2 章で VSBypass とその VM マイグレーションにおける問題について述べる．3 章で VM マイグレーション後にシャドウデバイスを用いた帯域外リモート管理を可能とするシステム USShadow を提案する．4 章で USShadow の実装について述べ，5 章で USShadow の性能を調べた実験について述べる．6 章で関連研究に触れ，7 章で本稿をまとめる．

2. シャドウデバイスを用いた帯域外リモート管理

2.1 VSBypass

クラウドにおいて安全な帯域外リモート管理を実現するために VSBypass [3] が提案されている．VSBypass は仮想化システムの外側でシャドウデバイスを動作させ，シャドウデバイス経由で帯域外リモート管理を行うことを可能にする．ユーザは仮想化システム内部の仮想デバイスを経由せずに仮想シリアルコンソールや VNC を用いた帯域外リモート管理を行えるため，仮想デバイスからの情報漏洩を防ぐことができる．VSBypass では，仮想化システム全体を信頼できないものとし，その外部だけを信頼できると仮定している．このようなシステムを実現するために，VSBypass では**ネストした仮想化**を用いて従来の仮想化システム全体を VM 内で動作させる．また，**強制パススルー**と呼ばれる機構を提供して，仮想化システム内部の VM が外部のシャドウデバイスを用いて入出力を行えるようにする．強制パススルー機構は VM が仮想デバイスに対して実行した入出力命令を横取りし，シャドウデバイスで透過的に処理する．シャドウデバイスとして，仮想シリアルデバイスと仮想キーボード，仮想マウス，仮想ビデオカードに対応している．

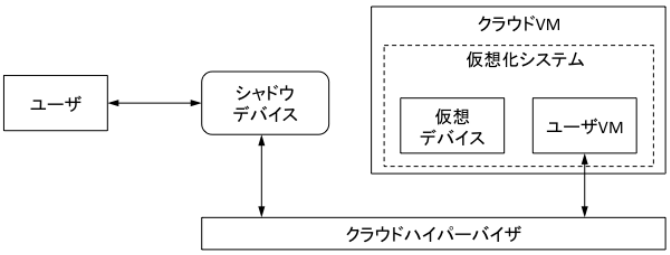


図 1 VSBypass のシステム構成

VSBypass のシステム構成を図 1 に示す．従来の仮想化システムはクラウド VM と呼ばれる VM 内で動作し，クラウド VM の中ではクラウド利用者の VM（ユーザ VM）が複数動作する．また，ユーザ VM 用の仮想デバイスもクラウド VM 内で動作する．クラウド VM の外側ではシャドウデバイスが仮想デバイスに 1 対 1 に対応して動作する．ユーザは SSH や VNC などを用いてネットワーク経由でシャドウデバイスにアクセスすることにより，安全な帯域外リモート管理を行う．クラウド VM とシャドウデバイスの下ではハイパーバイザが動作し，このハイパーバイザをクラウドハイパーバイザと呼ぶ．

VSBypass では，ユーザ VM が仮想デバイスに対して入出力命令を実行すると，その命令実行をクラウドハイパーバイザが横取りする．横取りした入出力命令はシャドウデバイスに転送して処理する．入力命令の場合には，リモートユーザがシャドウデバイスに送信した入力をユーザ VM に返す．出力命令の場合は，ユーザ VM の出力をシャドウデバイスに書き込み，リモートユーザはそれを取得する．一方，シャドウデバイスで仮想割り込みが発生した時，シャドウデバイスはユーザ VM に仮想割り込みを転送する．その際に信頼できない仮想化システムを経由するが，仮想割り込みには機密情報が含まれていないため情報漏洩の恐れはない．

シャドウビデオカードは他のシャドウデバイスと異なり，ユーザ VM のメモリを共有することにより VRAM へのアクセスを行う，VM は VRAM に対してメモリマップト I/O によるアクセスだけでなく，直接アクセスも行う．この直接アクセスはクラウドハイパーバイザが横取りできないため，クラウドハイパーバイザの機能を用いて VRAM への書き込みの検知を行う．これにより，シャドウビデオカードは VRAM の変更箇所を効率よく判定してリモートユーザに送信することができる．

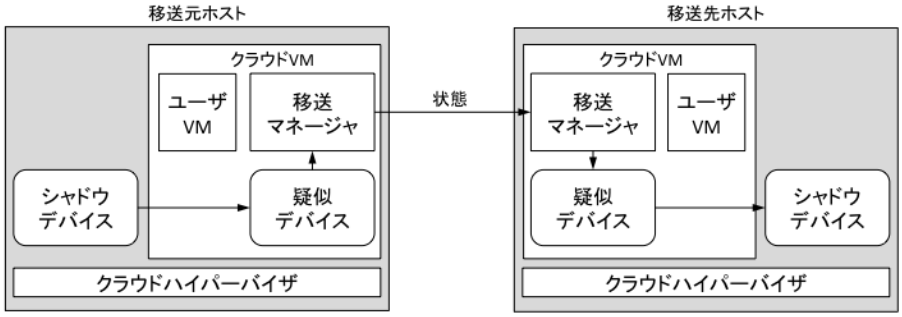


図 2　USShadow での VM マイグレーション

2.2　VSBypass における VM マイグレーション

VSBypass における VM マイグレーションは，移送元ホストと移送先ホストの仮想化システム内でそれぞれ動作する移送マネージャの間で行われる．VM マイグレーションの流れは以下のようになる．まず，移送元ホストの移送マネージャがユーザ VM のメモリを移送先ホストの移送マネージャに送信する．VM マイグレーションを行っている間にも VM の内容は更新されているため，更新されたメモリを再送する．再送するメモリ量が十分小さくなったところでユーザ VM を停止し，CPU や仮想デバイスの状態を移送先ホストに送信する．移送先ホストの移送マネージャでは，受信した CPU や仮想デバイスの状態を復元してユーザ VM の実行を再開する．

しかし，VSBypass では VM マイグレーションを行うと，シャドウデバイスを用いた帯域外リモート管理が行えなくなる．これは，シャドウデバイスが仮想化システムの外側で動作しており，仮想化システムの内部で動作する従来の仮想デバイスのように状態を保存したり復元したりすることができないためである．そのため，シャドウデバイスの状態を移送先ホストに転送することができず，シャドウデバイスの状態が失われてしまう．その結果，ユーザ VM やリモートユーザからシャドウデバイスに正常にアクセスできなくなる可能性がある．

また，VM マイグレーション後はユーザ VM の VRAM をシャドウビデオカードと再共有する必要がある．VSBypass では VRAM を共有するために，ユーザ VM が起動時にメモリマップト I/O 経由で VRAM にアクセスした際に，クラウドハイパーバイザが VRAM のメモリ情報を取得していた．しかし，VM マイグレーション後はメモリマップト I/O 経由で VRAM へのアクセスが行われないため，マイグレーション後に VRAM の情報を取得することができない．

3.　USShadow

本稿では，VM マイグレーション後もシャドウデバイスを用いた帯域外リモート管理を可能にするシステム USShadow を提案する．USShadow では，VM マイグレーションの際に仮想化システム内の移送マネージャが仮想化システムの外側にあるシャドウデバイスの状態を転送する．USShadow における VM マイグレーションの流れを図 2 に示す．移送元ホストの移送マネージャは仮想化システム内の疑似デバイス経由でシャドウデバイスの状態を取得し，取得した状態を移送先ホストの移送マネージャに送信する．その後，移送元ホストのシャドウデバイスを停止させる．一方，移送先ホストでは VM マイグレーションを開始した際にシャドウデバイスを起動する．そして，移送元ホストからシャドウデバイスの状態を受信した後，疑似デバイス経由で起動したシャドウデバイスに受信した状態を送る．このように，シャドウデバイスの状態を保存・復元することで，マイグレーション後に正常にシャドウデバイスを用いることが可能になる．

仮想化システム内で動作させる疑似デバイスは移送マネージャを改変せずに済ませるために用いられ，透過的にシャドウデバイスの状態の保存・復元を行う．移送マネージャが通常の仮想デバイスと同様に疑似デバイスの状態を保存しようすると，疑似デバイスはシャドウデバイスと通信を行い，シャドウデバイスの状態を取得する．このようにすることで，移送マネージャに対してシャドウデバイスの状態を疑似デバイスの状態であるかのように見せかけることができる．同様に，移送マネージャが疑似デバイスの状態を復元しようとした際には，疑似デバイスがシャドウデバイスと通信を行いシャドウデバイスの状態を復元する．

従来，マイグレーションの際に行われる仮想デバイスの状態の保存・復元にはプロセス間通信が用いられていた．これは，移送マネージャと仮想デバイスが同一の OS 上で動作しているため可能であった．しかし，シャドウデバイスは仮想化システムの外側で動作しているため，疑似デバイスとの間でプロセス間通信を行うことはできない．そこで，仮想ネットワークを用いた通信が考えられるが，ネットワーク通信を行うためにはシャドウデバイスに対してネットワーク経由でのアクセスを許可する必要がある．そのため，信頼できない仮想化システムからの攻撃を受ける危険性が増大する．また，仮想ネットワークを用いた通信を行うにはシステム内の多くのコンポーネントを経由する必要があるため，オーバヘッドが大きい．

そこで，USShadow では図 3 のように疑似デバイスがクラウドハイパーバイザを直接呼び出すことによりシャドウデバイスとの通信を行う．そのために，USShadow はウルトラコール [4] と呼ばれる機構を用いる．ウルトラコールは，CPU の仮想化支援機構を利用することで仮想化システムをバイパスしてクラウドハイパーバイザに直接制御を移す機構である．仮想化システムへの改変が不要であるため，既存の仮想化システムを用いることができる．その後，クラウドハイパーバイザがシャドウデバイスを呼び出し，シャドウデバイスが疑似デバイスのメモリに直接アクセスしてデータのやりとりを行う．疑似デバイスのメモリにアクセスするために，クラウドハイパーバイザは疑似デバイスのバッファの仮想アドレスをクラウド VM の物理アドレスに変換する．シャドウデバイスはクラウドハイパーバイザの機能を用いてそのメモリを共有してアクセスする．

疑似デバイスとの間でシャドウデバイスの状態をやり取りする際には，シャドウデバイス自身が状態の暗号化・復号化を行う．これにより，移送マネージャにおけるシャドウデバイスの状態からの情報漏洩を防ぐ．シャドウデバイスの状態にはリモートユーザにより入力されたパスワードやユーザ VM から出力された機密情報などが含まれる可能性がある．シャドウデバイスは信頼できない仮想化システムの外側で動作するため，シャドウデバイス自身で暗号化・復号化を行えば情報が漏洩することはない．移送元ホストの移送マネージャは暗号化されたシャドウデバイスの状態を取得し，それを移送先ホストに送信する．移送先ホストの移送マネージャは暗号化された状態をシャドウデバイスに送信し，シャドウデバイスにおいてその状態が復号される．

USShadow は，シャドウデバイスの状態を復元した後にユーザ VM とシャドウデバイスの間で VRAM の再共有を行うことを可能にする．クラウドハイパーバイザは VM マイグレーション後に移送先ホストでユーザ VM が再開されたのを検知して，ユーザ VM の拡張ページテーブル（EPT）の取得を行う．その EPT を用いてユーザ VM の VRAM の物理アドレスをクラウド VM の物理アドレスに変換する．これにより，ユーザ VM により VRAM に対してメモリマップト I/O が実行されなくても VRAM のメモリ情報を取得することができる．シャドウデバイスはその物理アドレスを取得して VRAM のメモリをマッピングすることで，ユーザ VM の VRAM にアクセスすることができる．

4. 実装

我々は，USShadow を Xen 4.8.0 をベースとする VSBypass に実装した．USShadow を実装したハイパーバイザをクラウドハイパーバイザとして動作させ，その上でクラウド管理 VM とクラウド VM を動作させる．クラウド VM 内では既存の仮想化システムを動作させる．現在のところ，Xen と KVM に対応している．Xen の場合，図 4 のように仮想化システム内ではゲストハイパーバイザとゲスト管理 VM が動作し，ゲスト管理 VM 内で移送マネージャと疑似デバイスが動作する．KVM の場合，ゲストハイパーバイザが動作する Linux カーネルの上で移送マネージャと疑似デバイスが動作する．シャドウシリアルデバイスの状態を保持する VM マイグレーションは Xen と KVM の両方に対応しており，シャドウキーボード，シャドウマウス，シャドウビデオカードの状態を保持する VM マイグレーションは Xen のみに対応している．

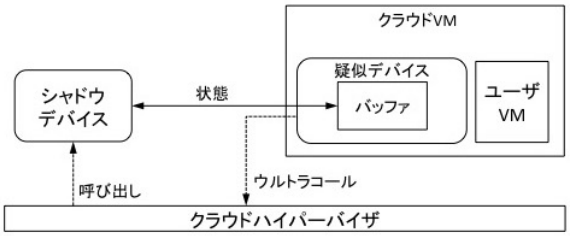


図 3　シャドウデバイスとの通信

4.1 シャドウデバイス

USShadow におけるシャドウデバイスの実装は，VSBypass と同様である．シャドウデバイスは，ユーザ VM ごとに作成されるプロキシ VM により提供される仮想デバイスである．プロキシ VM はクラウドハイパーバイザ上で動作し，ユーザ VM に強制パススルー先のシャドウデバイスを提供するためだけに用いられる．プロキシ VM にはクラウド VM と同数の仮想 CPU を割り当てる必要があるが，起動後には停止状態となり，メモリやディスクの割り当ては最低限でよい．現在の実装では，移送元ホストではユーザ VM の起動前，移送先ホストではユーザ VM の移送前に起動しておく必要がある．クラウドハイパーバイザにおいてプロキシ VM とユーザ VM を対応づけるために，ユーザ VM の EPT を利用する．

クラウドハイパーバイザは，ユーザ VM が帯域外リモート管理に用いられる仮想デバイスに対して入出力命令を実行した時だけその命令を横取りし，強制パススルーを行う．ユーザ VM が入出力命令を実行すると VM Exit が発

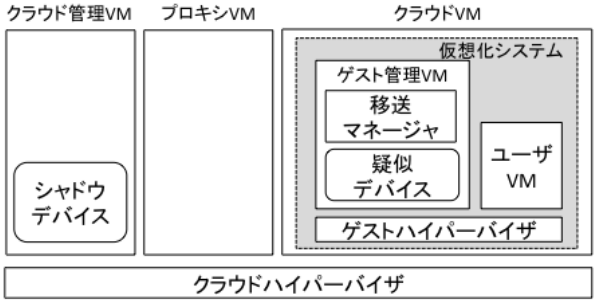


図 4　USShadow のシステム構成

生し，クラウドハイパーバイザに制御が移る．I/O マップト I/O の場合はポート番号，メモリマップト I/O の場合はアクセスされたメモリアドレスが強制パススルーの対象であった場合，クラウドハイパーバイザにおいて命令のエミュレーションを行う．そのためにシャドウデバイスに入出力要求を送り，シャドウデバイスで入出力処理が完了すると，ユーザ VM に対して VM Entry を実行する．

USShadow ではマイグレーション後も VRAM 情報を取得できるように，4.5 節のメモリマップト I/O に依存しない手法を用いる．VSBypass では，ユーザ VM が起動時にて VRAM に対して実行するメモリマップト I/O を横取りして VRAM 情報を取得していた．ユーザ VM が VRAM を直接，書き換えた場合には強制パススルーを行うことができないため，シャドウビデオカードはクラウドハイパーバイザのログダーティ機構を用いて書き換えを検出する．この機構は VRAM を書き込み禁止にし，書き込まれた際に VM Exit を発生させて記録する．

4.2 疑似デバイス

疑似デバイスはシャドウデバイスに 1 対 1 に対応した疑似的な仮想デバイスであり，仮想化システム内の QEMU に組み込んで動作させる．疑似デバイスは QEMU への登録および状態の保存・復元の処理だけを行う．そのため，疑似デバイスはシャドウデバイスの実装から容易に作成することができる．

疑似デバイスはシャドウデバイスの状態を保存するために，状態を格納するバッファをウルトラコールを用いてシャドウデバイスと共有する．シャドウデバイスとの通信の詳細については 4.3 節で述べる．このバッファにシャドウデバイスが状態を格納することで，シャドウデバイスの状態が疑似デバイスに格納される．シャドウデバイスの状態のサイズはウルトラコールの返り値で返される．移送マネージャは疑似デバイスの状態としてシャドウデバイスの状態を取得し，移送先ホストに転送する．

疑似デバイスがシャドウデバイスの状態を復元する際には，バッファを確保して移送マネージャが受信したシャドウデバイスの状態を格納する．そして，ウルトラコールを用いてそのバッファをシャドウデバイスと共有し，シャドウデバイスはバッファ内の状態を用いて復元を行う．

4.3 シャドウデバイスとの通信

疑似デバイスはシャドウデバイスとの通信を行うために，まず，ウルトラコールを用いてクラウドハイパーバイザを明示的に呼び出す．ウルトラコールはハイパーコール実行にも用いられている vmcall 命令を用いて実装した．ゲスト管理 VM 内で vmcall 命令を実行すると，ゲストハイパーバイザではなく，クラウドハイパーバイザに VM Exit が発生する．シャドウデバイスの状態の保存・復元を行うウルトラコールでは引数として，状態の保存・復元のどちらを行うか，シャドウデバイスの状態が格納される疑似デバイスのバッファの仮想アドレス，そのサイズを指定する．バッファは mmap システムコールで確保された 1 ページ分の匿名メモリであり，ページアウトされないように mlock システムコールを用いてピン留めする．

ウルトラコールで呼び出されたクラウドハイパーバイザは，シャドウデバイスが疑似デバイスのメモリにアクセスできるようにするために，バッファ用メモリのアドレス変換を行う．まず，クラウド VM の仮想 CPU からページテーブルのアドレスが格納されている CR3 レジスタの値を取得する．ゲスト管理 VM では CPU が準仮想化されているため，VM Exit を発生させた時のクラウド VM の仮想 CPU のレジスタ値はゲスト管理 VM の仮想 CPU のものと同一である．そして，そのページテーブルをたどり，疑似デバイスが動作している QEMU の仮想アドレスからクラウド VM の物理アドレスへの変換を行う．ゲスト管理 VM ではメモリも準仮想化されているため，ページテーブルにはクラウド VM の物理アドレスが直接格納されており，EPT をたどる必要はない．

クラウドハイパーバイザはシャドウデバイスが動作している QEMU に特殊な I/O リクエストを送信することにより，シャドウデバイスを呼び出す．その際に，通常の I/O リクエストと区別するために，専用の I/O ポート番号を利用する．I/O リクエストには I/O ポート番号，疑似デバイスのバッファの物理アドレス，そのサイズを格納する．シャドウデバイスでリクエスト処理が完了したら，保存・復元が行われた状態のサイズをウルトラコールの返り値として疑似デバイスに返す．

I/O リクエストを受信したシャドウデバイスは，疑似デバイスのバッファの物理アドレスを用いてメモリのマッピングを行う．そのメモリに対して読み書きを行うことで，疑似デバイスとの通信を行う．シャドウデバイスはプロキシ VM 用の QEMU 内に実装されているが，この QEMU はプロキシ VM のメモリにアクセスするインタフェースしか提供していない．そこで，クラウドハイパーバイザの機能を用いてクラウド VM のメモリにアクセスするインタフェースを追加した．

4.4 シャドウデバイスの状態の保存・復元

仮想デバイスの状態を保存・復元するために QEMU が提供している機能を用いて，シャドウデバイスの状態の保存・復元を行う．従来の QEMU は仮想デバイスの状態をファイルまたはソケットにしか保存することができなかった．この機能は，VM の状態をディスクに保存したり，マイグレーションで移送先に転送したりする際に使われるためである．そこで，USShadow の通信機構とともに利用できるようにするために，仮想デバイスの状態をメモリに保

表 1　シャドウデバイスの状態

種類	状態	サイズ（バイト）
シャドウシリアルデバイス	10 個のレジスタ	16
	送受信用 FIFO 等の内部状態	
シャドウキーボード	4 つの状態	288
	1 つの内部状態	
シャドウマウス	4 つの状態	304
	マウスの情報を格納するキュー	
シャドウビデオカード	27 個のレジスタ	1408

存する機能を追加した．これにより，疑似デバイスのバッファにシャドウデバイスの状態を書き込んだり，読み込んだりすることができる．

USShadow では，シャドウシリアルデバイス，シャドウキーボード，シャドウマウス，シャドウビデオカードの状態の保存・復元を行う．状態の保存は，デバイスレジスタの値などをバッファに保存することにより行う．その際に，デバイスレジスタの値を 16 バイト単位で AES により暗号化する．一方，状態の復元はバッファのデータを復号し，デバイスレジスタに書き込むことにより行う．それぞれのシャドウデバイスにおいて保存・復元される状態とそのサイズを表 1 に示す．

4.5　VRAM の共有

シャドウビデオカードがユーザ VM の VRAM にアクセスできるようにするために，VRAM として使用されているメモリを特定してメモリマッピングを行う．シャドウデバイスはプロキシ VM の仮想デバイスであるため，そのままではプロキシ VM の VRAM にしかアクセスできない．ユーザ VM の起動時や再開時にはゲストハイパーバイザが vmptrld 命令を実行するため，クラウドハイパーバイザに VM Exit が発生する．クラウドハイパーバイザでは，ユーザ VM の VMCS から EPT のアドレスを取得し，VRAM の各ページの物理アドレスをクラウド VM の物理アドレスへと変換する．Xen では VM のメモリサイズに関わらず，ユーザ VM の物理アドレス F1000000 から 16MB の物理メモリが VRAM として使用されている．そして，図 5 のようにユーザ VM の VRAM をシャドウビデオカードにマッピングし，プロキシ VM の VRAM の代わりに使用する．これにより，シャドウデバイスからユーザ VM の VRAM への透過的なアクセスを実現する．

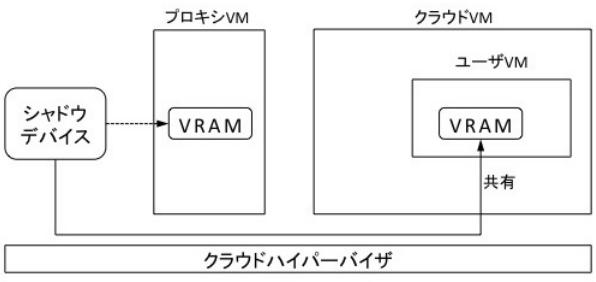


図 5　VRAM の共有

5.　実験

USShadow を用いて VM マイグレーション後の帯域外リモート管理の動作を確認し，VM マイグレーションの性能を調べる実験を行った．実験には，Intel Xeon E3-1226 v3 の CPU，8GB のメモリ，ギガビットイーサネットを搭載したマシン 2 台を移送元ホスト，移送先ホストとして用いた．クラウド VM には 2 個の仮想 CPU と 3GB のメモリ，ユーザ VM には 2 個の仮想 CPU と 256MB のメモリ，プロキシ VM には 2 個の仮想 CPU と 1GB のメモリをそれぞれ割り当てた．クラウド VM 内の仮想化システムには Xen 4.4.0 と KVM を用いた．比較として，ネストした仮想化を用いる従来システムおよび，マイグレーション時にシャドウデバイスの状態を転送しない VSBypass を用いた．

5.1　マイグレーション後の帯域外リモート管理

USShadow を用いて，VM マイグレーション後にシャドウデバイスを用いた帯域外リモート管理が可能であることの確認を行った．まず，仮想シリアルコンソールを用いた帯域外リモート管理を行い，ユーザ VM にログインした状態で VM マイグレーションを行った．VM マイグレーション後に，マイグレーション先のシャドウデバイスに接続して帯域外リモート管理を行ったところ，正常にアクセスでき，ユーザ VM にログインした状態から再開できた．次に，VNC を用いた帯域外リモート管理を行い，ユーザ VM にログインした状態で VM マイグレーションを行った．この場合も，帯域外リモート管理を用いて正常にアクセスでき，ユーザ VM にログインした状態から再開できた．

5.2　状態の保存・復元時間

USShadow において，シャドウデバイスの状態の保存・復元にかかる時間の測定を行った．まず，シャドウシリアルデバイスとシャドウキーボード，シャドウマウス，シャドウビデオカードに対してそれぞれウルトラコールを発行して状態の保存・復元を行った．測定結果を図 6 に示す．状態の保存・復元に必要な時間は，基本的に表 1 に示した状態のサイズに応じて増加したが，シャドウマウスの復元はシャドウキーボードよりも高速であった．最もサイズの大きいシャドウビデオカードでも保存・復元時間の合計は 0.93 ミリ秒であった．次に，これらのシャドウデバイスに対して連続してウルトラコールを発行し，すべての状態を保存・復元するのにかかる時間を測定した．保存・復元時間の合計は 3 ミリ秒となり，5.3 節で示すダウンタイムに対して非常に短い時間で完了することが分かった．

5.3　マイグレーション性能

USShadow を用いた VM マイグレーションの性能を測定

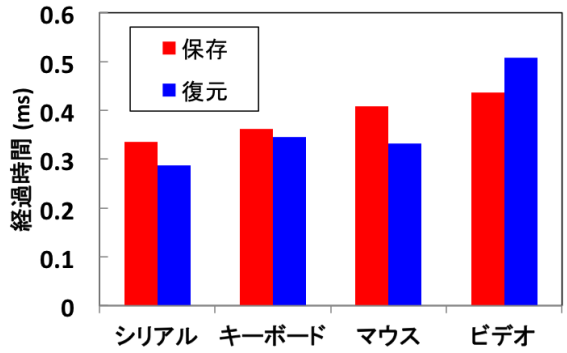


図 6　シャドウデバイスの保存・復元にかかる時間

し，ネストした仮想化を用いる従来システムと VSBypass との比較を行った．

5.3.1　仮想化システムに Xen を使用した場合

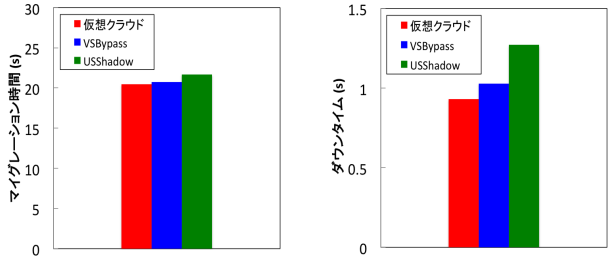
クラウド VM 内で動作させる仮想化システムとして Xen を用いた場合のマイグレーション時間の測定結果を図 7（a）に示す．マイグレーション時間は，移送元で移送マネージャの実行を開始してから完了するまでの時間を測定した．USShadow ではマイグレーション時間が VSBypass より 0.9 秒長かった．これはシャドウデバイスの状態を疑似デバイス経由で保存・復元したり，移送先に転送したりすることによるオーバヘッドと考えられる．一方，ネストした仮想化を用いる従来システムと比べると，1.2 秒程度の増加が見られた．これは，シャドウデバイスの状態を扱うオーバヘッドに加えて，VSBypass によるシャドウデバイスを用いた入出力処理のオーバヘッドと考えられる．

VM マイグレーション中のダウンタイムを図 7（b）に示す．ダウンタイムは，移送元ホストで VM が停止されてから移送先ホストで再開されるまでの時間を測定した．USShadow は，VSBypass と比べてダウンタイムが 0.2 秒増加し，従来のシステムに比べると 0.3 秒増加した．これもマイグレーション時間の増加と同じ理由によるものと考えられる．

5.3.2　仮想化システムに KVM を使用した場合

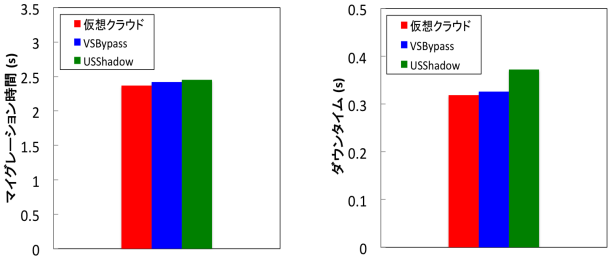
仮想化システムとして KVM を用いた場合のマイグレーション時間の測定結果を図 8（a）に示す．Xen の場合よりマイグレーション時間が大幅に短いのは，ネストした仮想化における仮想ネットワークの性能が高かったためである．USShadow ではマイグレーション時間が VSBypass より 0.02 秒，従来システムより 0.08 秒だけ長かった．Xen の場合と比べてオーバヘッドが小さい理由については，現在，調査中である．

VM マイグレーション中のダウンタイムを図 8（b）に示す．USShadow は VSBypass や従来システムと比べてダウンタイムが 0.05 秒増加した．マイグレーション時間の増加よりもダウンタイムの増加のほうが大きい原因については，現在調査中である．



（a）マイグレーション時間　　（b）ダウンタイム

図 7　Xen でのマイグレーション性能



（a）マイグレーション時間　　（b）ダウンタイム

図 8　KVM でのマイグレーション性能

5.4　CPU 使用率

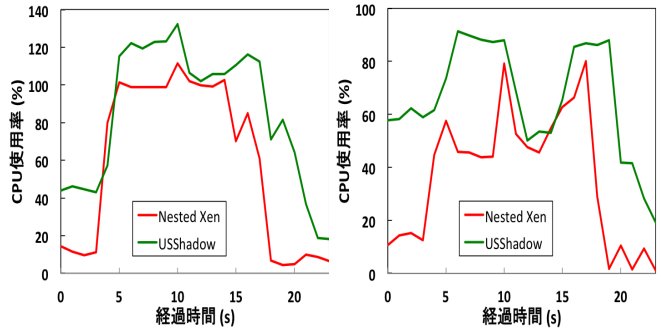
USShadow を用いたマイグレーション中のクラウド管理 VM とクラウド VM の CPU 使用率を測定し，ネストした仮想化を用いた従来システムとの比較を行った．移送元ホストでの CPU 使用率は図 9 のようになった．全体的に USShadow のほうが CPU 使用率が高いのは，クラウド管理 VM でシャドウデバイスが動作しており，クラウド VM で割り込みサーバがポーリングを行っているためだと考えられる．USShadow では 15〜20 秒付近でシャドウデバイスの保存・転送を行ったため，CPU 使用率が高い状態が続いている．一方，図 10 は移送先ホストでの CPU 使用率である．15〜20 秒付近でクラウド管理 VM の CPU 使用率が高くなっているのは，シャドウデバイスの状態の受信・復元のためと考えられる．

5.5　帯域外リモート管理の性能

VM マイグレーションの前後で，シャドウデバイスを用いた帯域外リモート管理の性能変化を調べた．仮想シリアルコンソールと VNC を用いた帯域外リモート管理の性能を測定し，VNC を用いる場合にはテキストモードとグラフィックモードの二つについて測定を行った．

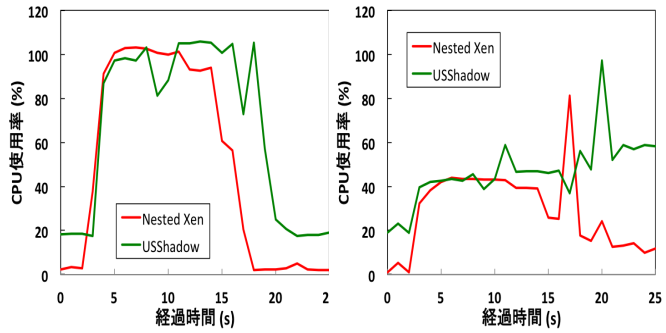
5.5.1　仮想シリアルコンソールの性能

SSH を用いて仮想シリアルコンソールでの入力に対する応答時間の測定を行った．応答時間は，SSH クライアントで文字の入力を行った時点から，その入力がユーザ VM によって処理され SSH クライアントにエコーバック表示されるまでの時間とした．測定結果を図 11（a）に示す．マイグレーション前後でのユーザ VM の応答時間の差は 0.06 ミリ秒であった．このことから，応答時間はマイグレー



（a）クラウド管理 VM　　（b）クラウド VM

図 9　移送元ホストでの CPU 使用率



（a）クラウド管理 VM　　（b）クラウド VM

図 10　移送先ホストでの CPU 使用率

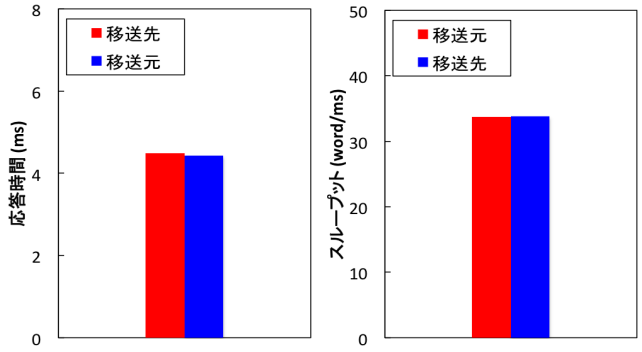
ション前後で誤差程度の違いしかないことが分かった．

次に，帯域外リモート管理を用いてログインしたユーザ VM で cat コマンドを実行し，テキストファイルの中身を表示した時のスループットを測定した．SSH クライアントでは，コマンドを実行した時からテキストファイルの表示が終わるまでの時間を測定し，スループットを算出した．測定結果を図 11（b）に示す．スループットもマイグレーション前後で 0.2 %しか差がなく，応答時間と同様にマイグレーション前後で同程度の性能であることが分かった．

5.5.2　VNC の性能

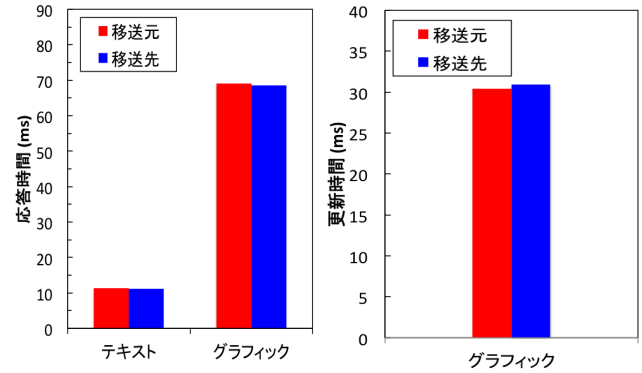
VNC を用いて帯域外リモート管理におけるキーボード入力の応答時間を測定した．応答時間は，VNC クライアントで文字を入力してから，その文字がユーザ VM に表示され，画面が書き換わることで送られてくる更新データを VNC クライアントが受け取るまでの時間とした．測定結果を図 12（a）に示す．テキストモードでは性能差は 1.4 %，グラフィックモードで性能差は 0.7 %であり，性能差は小さかった．

グラフィックモードの場合におけるスクリーンデータの更新時間の測定結果を図 12（b）に示す．更新時間は，ユーザ VM の画面を一定時間ごとに書き換えている際に，VNC クライアントが画面の更新要求を送信してから更新データを受信するまでの時間とした．ユーザ VM の画面の更新領域は，幅 300 ピクセル，高さ 600 ピクセルの矩形であり，フレームレートは 1fps とした．実験の結果，マイグレーション前後での性能差は 1.4 %であり，性能差は小さかった．



（a）応答時間　　（b）スループット

図 11　仮想シリアルコンソールでの帯域外リモート管理性能



（a）キーボード入力の応答時間　（b）スクリーンデータの更新時間

図 12　VNC での帯域外リモート管理性能

6.　関連研究

USShadow のシャドウデバイスは一種のパススルーデバイスである．パススルーデバイスを用いた VM をマイグレーションするシステムはいくつか提案されている．

ボンディングドライバを利用したマイグレーション [5] では，ボンディングドライバを用いてパススルーネットワークデバイスと準仮想化ネットワークデバイスを束ねる．マイグレーション時にパススルーデバイスをホットアンプラグすると，準仮想化デバイスに自動的に切り替わる．準仮想化デバイスを用いる VM は容易にマイグレーションすることができる．マイグレーション後にパススルーデバイスをホットプラグすると，準仮想化デバイスから自動的に切り替わる．設定の変更のみでマイグレーションに対応できる利点があるが，ネットワークデバイスにしか対応しておらず，ホットアンプラグの際にダウンタイムが発生するなどの欠点もある．

Network Plug-In Architecture (NPIA/NPA) では，マイグレーション中にネットワークドライバのプラグインを抜き差しすることで，パススルーデバイスを用いる VM のマイグレーションに対応する．ボンディングドライバを用い

る手法と同様に，マイグレーションの際には準仮想化デバイスのプラグインに切り替える．ボンディングドライバより切り替え時間が短いという利点があるが，SR-IOV NIC にしか対応しておらず，ネットワークドライバの大幅な書き換えが必要となる．

シャドウドライバを利用したマイグレーション [6] では，シャドウドライバが OS からネットワークドライバへのアクセスを記録する．パススルーデバイスを用いる VM のマイグレーション中はシャドウドライバがリクエストを処理し，移送先ホストで新たなネットワークドライバが起動したらリダイレクトを行う．しかし，ハイパーバイザと OS に大幅な変更が必要である．

CompSC [7] では，移送元ホストでパススルー NIC に行った操作を記録し，移送先ホストで再現することでデバイスレジスタの値を復元する．これにより，読むと値が消えるデバイスレジスタや，書き込めないレジスタ，書き込むとデバイスが何らかの処理を行うレジスタなどについても状態を復元することができる．加えて，CompSC ではハイパーバイザでエミュレーションを行い，統計情報等の復元も行う．USShadow ではパススルーデバイスであるシャドウデバイスは仮想デバイスであるため，状態を容易に保存・復元することができる．

SRVM [8] では，移送先ホストで SR-IOV NIC の状態を完全に復元するのではなく，正しく動作するようにだけ復元を行う．そのために，NIC が受信したパケットを書き込んだメモリを検出して，移送先ホストに転送できるようにしている．このようにして，SRVM では最小限の状態だけを復元する．

D-MORE [9] は VM マイグレーション後も従来の帯域外リモート管理を継続することができる．D-MORE では VM の仮想デバイスをマイグレーション可能なリモート管理専用 VM の上で動かす．マイグレーションの際には，ユーザ VM とリモート管理専用 VM を一緒に転送することで仮想デバイスの状態を保持する．しかし，シャドウデバイスを用いた帯域外リモート管理に D-MORE の手法を適用するには，シャドウデバイス専用の VM を仮想化システムの外部に用意する必要がある．仮想化システム内の移送マネージャがユーザ VM と一緒に仮想化システムの外側にある VM をマイグレーションできるようにすると，セキュリティ上の問題が発生する可能性がある．

ネストした仮想化における通信の高速化も提案されている．Xen-Blanket [10] は，ゲスト管理 VM とクラウド管理 VM の間の高速な通信機構を提供する．Xen-Blanket では，ゲスト管理 VM の OS 内で Blanket ドライバを動作させ，ゲストハイパーバイザへのハイパーコール呼び出しを経由して，クラウド管理 VM と通信を行う．一方，USShadow ではゲスト管理 VM からクラウドハイパーバイザを直接呼び出して，クラウド管理 VM と通信を行う．Xen-Blanket ではクラウド VM 内の仮想化システムにのみ改変を加える設計であるのに対し，USShadow では仮想化システムは変更しない設計である点が異なる．

7. まとめ

本稿では，VM マイグレーション後にシャドウデバイスを用いた帯域外リモート管理を可能にするシステム USShadow を提案した．USShadow は仮想化システム内の疑似デバイス経由でシャドウデバイスの状態を保存し，移送先で状態を復元することができる．疑似デバイスは仮想化システムの外側にあるシャドウデバイスと安全で効率のよい通信を行う．マイグレーション中の情報漏洩を防ぐために，移送元のシャドウデバイスで状態を暗号化し，移送先のシャドウデバイスでその状態を復号する．USShadow を Xen 4.8.0 に実装し，シャドウデバイスを用いた帯域外リモート管理がマイグレーション後にも継続して行えることを確認した．また，USShadow における VM マイグレーションの性能低下は小さいことが分かった．

今後の課題は，仮想化システムとして KVM を動作させた場合に，VNC を用いた帯域外リモート管理を可能にすることである．その上で，マイグレーション後に VNC を用いた帯域外リモート管理を継続させられるようにする．

参考文献

[1] CyberArk Software. Global IT Security Service, 2009.
[2] PwC: US Cybercrime: Rising Risks, Reduced Readiness 2014.
[3] S. Futagami, T. Unoki, K. Kourai: VSBypass: Secure Out-of-band Remote Management of Virtual Machines with Transparent Passthrough. In Proc. Annual Computer Security Applications Conf. 2018.
[4] S. Miyama, K.Kourai. 2017 Secure IDS Offloading with Nested Virutalization and Deep VM Introspection. In Proc. European Symp. Research in Computer Security, Part . 305-323.
[5] E. Zhai, G. D. Cummings, and Y. Dong. Live migration with passthrough device for Linux VM. In Ottawa Linux Symp. 2008.
[6] A. Kadav and M. M. Swift. Live migration of direct-access devices. SIGOPS Oper. Syst. Rev., 43:9-104, 2009.
[7] Z. Pan, Y. Dong, Y. Chen, L. Zhang, Z. Zhang: CompSC: live migration with pass-through devices. In Proc. Int. Conf. Virtual Execution Environments, pp.109-120, 2012.
[8] X. Xu, B. Davda: SRVM: Hypervisor Support for Live Migration with Passthrough SR-IOV Network Devices. In Proc. Int. Conf. Virtual Execution Environments, pp.65-77, 2016.
[9] Sho Kawahara,Kenichi Kourai: D-MORE: The Continuity of Out-of-band Remote Management across Virtual Machine Migration in Clouds, In Proc. Int. Conf. Utility and Cloud Computing, pp.176-185, 2014.
[10] D. Williams, H. Jamjoom, and H. Weatherspoon: The Xen-Blanket: Virtualize Once, Run Everywhere. In Proc. European Conf. Computer Systems, pp.113-126, 2012.