Vol.2011-ARC-195 No.16
Vol.2011-OS-117 No.16
2011/4/13

I/O エミュレーションのロギングリプレイによる仮想マシン同期機構の高速化

大村 圭[†1] 田村芳明[†1]
湯口 徹[†1] 盛合敏[†1]

仮想マシンを利用した高可用技術は，アプリケーションや OS に依存せずに利用可能な有用な手段である．しかし，広域ネットワーク (WAN) 環境で利用するためには，仮想マシン状態に加え，ストレージの複製が必要であるため，オーバヘッドが大きく仮想サーバ性能が著しく低下するという問題がある．本論文では，QEMU の I/O エミュレーションのロギングリプレイによる，仮想マシン同期機構の高速化を提案する．プライマリとバックアップが同様の I/O エミュレーションを実行することで，仮想マシンが利用するストレージの同期を実現している．提案手法を仮想マシンモニタ KVM/QEMU に実装し，実験した結果，同期のためのデータ通信量を削減し，従来技術と比較して高い性能を得ることができた．

Rapid VM Synchronization with I/O Emulation Logging-Replay

Kei Ohmura ,[†1] Yoshiaki Tamura ,[†1]
Toru Yuguchi [†1] and Satoshi Moriai[†1]

Virtual machine (VM) based high availability technologies are novel approaches that does not require specific hardware or modifications to software. However it is difficult to use in wide-area networks, because the perfomance of VM may heavyly degrade due to synchronization of VM and storage. This paper describes rapid VM synchronization with QEMU's I/O emulation logging-replay which replicates the data of storage with low-overhead. We implemented a prototype based on KVM/QEMU, and measured the performance of the system in a WAN-emulated environment. We demonstrate that our approach reduces network traffics and performs better than existing approaches.

1. はじめに

近年，企業の事業活動の多くが情報システムに依存しており，情報システムの停止が事業の停止に直結する．例えば，インターネットサービスを提供する企業の情報システムが停止した場合，企業に大きな損害を与えると共に，多くのユーザにも影響を与えてしまう．システム停止の要因は，人的要因やソフトウェアのバグ，ハードウェアメンテナンスのための計画的な停止，ハードウェア故障などあり，これらに対してシステムの可用性を高めることは，重要な課題である．

従来の高可用技術は，アプリケーションに応じた構成を組む必要が有り，サーバ上の全てのアプリケーションに対応しなければならなかったが，近年は，仮想マシンのライブマイグレーション[4] を利用したシステムの計画停止の回避や，仮想マシン複製技術[3)5)13)15)18)] によるハードウェア故障の回避 (Fault Tolerant: FT) といったように，アプリケーションや OS に依存しない，仮想マシンによる高可用技術が有用な手段として注目を浴びている．特に，データセンタ間のライブマイグレーションや FT 技術は，システムの柔軟な運用やデータセンタの停電などの大規模障害に対する手段として重要な技術である．しかし，仮想マシンを利用した高可用技術は，仮想マシン状態（主にメモリ）を複製するためのオーバヘッドが大きく，特に，データセンタ間のような 広域ネットワーク (WAN) 環境で利用するためには，仮想マシン状態に加え，ストレージの複製も必要であり，帯域と遅延の影響により，複製元の仮想サーバ性能が著しく低下するという問題がある．また，FT 技術では同期的に仮想マシンおよびストレージを複製する必要があるため，特に，遅延の影響を大きく受けてしまう．

本論文では，QEMU の I/O エミュレーションのロギングリプレイによるストレージ複製手法を提案する．プライマリの QEMU が行う I/O エミュレーションの内，特にストレージ書き込みのエミュレーションのログをバックアップに転送し，バックアップの QEMU はログをもとに，同様の I/O エミュレーションをリプレイし，ストレージ書き込みを行うことで，ストレージデータの転送を行うことなく，ストレージの複製を実現した．提案手法をオープンソースの仮想マシンモニタ KVM/QEMU を用いて実装し，先行研究[17] で開発した仮想マシン同期機構に組み込み，同期的に仮想マシンおよびストレージを複製させた際

†1 日本電信電話株式会社サイバースペース研究所
NTT Cyber Space Laboratories

Vol.2011-ARC-195 No.16
Vol.2011-OS-117 No.16
2011/4/13

の，同期用ネットワークの通信量や WAN 環境での性能について評価した．

本論文の構成は以下のとおりである．2 節では，既存の仮想マシン複製技術とその問題点について説明する．3 節で提案手法について述べ，4 節で実装について述べる．5 節で実験結果について記述する．6 節で，関連研究との比較を行い，7 節で本論文をまとめ今後の課題について述べる．

2. 既存の仮想マシン複製技術とその問題点

本節では，プライマリ・バックアップ構成のサーバ上で動作する仮想マシンおよびストレージを複製するシステムを実現するための，既存の仮想マシン複製技術について説明し，我々の開発した仮想マシン同期機構と既存のストレージ複製技術を組み合わせて WAN 環境に適用した際の問題および予備実験の結果について述べる．

2.1 既存の仮想マシン複製技術

仮想マシンを複製する手法は，主に，仮想マシンスナップショットを転送する方式と，ロックステップ方式にわけることができる．本研究は，汎用性の点から仮想マシンスナップショットを転送する方式をベースにしている．以下で，各手法の利点・欠点について述べる．

仮想マシンのスナップショットを転送する方式では，プライマリの仮想マシンの状態 (CPU，メモリ，デバイス状態など) をバックアップに転送することで，それぞれの仮想マシンの状態を同一に保つことができる．また，ストレージデータに関しては共有ストレージを利用するか，仮想マシンの状態と同様にバックアップに転送する必要がある．仮想マシンを停止せずに複製する場合や，定期的に複製を行う場合は，ダーティビットマップを利用し，前回複製時の差分転送を行うことで転送量を抑えることができる．複製に必要なデータ転送量はロックステップ方式と比較して大きくなってしまうが，実装が容易であり，利用できるプロセッサが制約されるといったことがない．また，複製のタイミングは，ライブマイグレーションのように，タイマなどを契機に定期的に差分転送を行い，ダーティページの量が一定以下になると，仮想マシンを停止して，残りのダーティページおよびデバイス状態を転送する方式や，FT 機能のように，ストレージ書き込みや，ネットワークアウトプットといったタイミングを契機に同期的に複製を開始する方式[18] や，ネットワークアウトプットをバッファリングし，同期後に出力することで整合性を保つ方式[5] がある．

一方，ロックステップ[3][13][15] 方式は，プライマリとバックアップに同じ入力を与え，各仮想マシンで同様の処理を行わせることで，仮想マシンの状態を同一に保つことができる．各仮想マシンで計算を行わせるため，複製に必要なデータ量は小さくて済むが，マルチコアへの対応が難しく，利用可能なプロセッサが限定されてしまうという問題がある．本研究では，アプリケーションやハードウェアに依存しない，汎用的な手法による実現を目指しているため，この方式は目的に合わない．

2.2 WAN 環境に適用する際の問題

本節では，仮想マシンスナップショットを転送する方式を低帯域・高遅延な WAN 環境に適用し，仮想マシンおよびストレージを複製させた際に生じる問題について述べる．

プライマリとバックアップの仮想マシンを複製するシステムでは，一般的にプライマリ・バックアップ両方の仮想マシンからアクセスできる共有ストレージを用意する必要がある．しかし，WAN を介して共有すると，遅延によるアクセス性能の低下や，ストレージを共有するための距離の制限，共有ストレージが単一障害点になる，といった様々な問題が生じる．そのため， WAN で仮想マシンを複製する際は，それぞれに固有のローカルストレージを用意し，ストレージのデータも仮想マシンと同様に転送する必要があり，オープンソースのストレージ複製ソフトウェアである DRBD[6] の利用や，ストレージに書き込むデータ（もしくは既に書き込まれたデータ）を転送することで実現できる．しかし，仮想マシンとストレージの両方を複製するためのデータ転送量は大きく，低帯域に対応するためには転送量の削減が必要である．

次に，プライマリとバックアップの仮想マシンおよびストレージを一致させる複製処理の頻度について述べる．プライマリがいつ停止しても，バックアップで即座にサービスを引き継いで矛盾なく実行するためには，特定のタイミングを契機に同期的に複製する必要がある．しかし，高遅延ネットワークでは，プライマリがバックアップからの同期完了通知を待つだけで，遅延分の時間がかかってしまうため，同期頻度が高い場合のサーバ性能低下は著しい．同期頻度を削減する手法として，I/O をバッファリングし，ある程度の I/O がバッファにたまると，同期処理を行う手法あるが，I/O のスループットに大きな影響を与えるため，サーバ性能低下の改善につながらず，本質的な問題解決にはならない．そのため，同期頻度を削減するための手法が必要である．

2.3 予 備 実 験

仮想マシンとストレージを定期的かつ同期的に複製するシステムを WAN を模擬した環境に適用した時の予備実験の結果について述べる．仮想マシンスナップショットを定期的に同期転送するシステムに，DRBD もしくは，ストレージデータも同様に転送して複製する手法を組み合わせて実験を行った．それぞれの組み合わせに応じて，DRBD，dirty-block-copy として示している．また，実験環境や実験手法の詳細は，5 節に記述する．

Vol.2011-ARC-195 No.16
Vol.2011-OS-117 No.16
2011/4/13

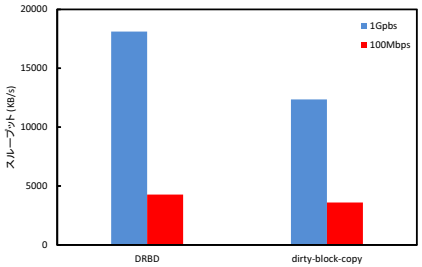

図 1 各帯域での I/O 性能
Fig. 1 I/O throughput with each bandwidth.

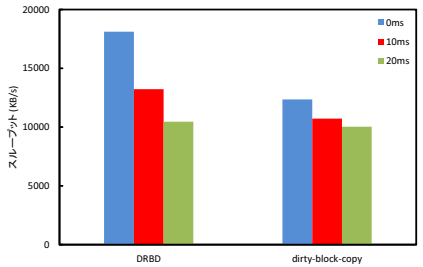

図 2 各遅延での I/O 性能
Fig. 2 I/O throughput with each bandwidth.

帯域や遅延をツールにより変化させた同期用ネットワークで，仮想マシンとストレージを同期させているゲスト OS 上でのファイル I/O 性能を測定した．図 1 に帯域によるファイル I/O 性能への影響を示す．帯域を 100Mpbs にすることで 1Gpbs の性能に対して，DRBD も dirty-block-copy も 30%以下に性能が低下していた．また，1Gpbs で同期させた場合の同期用ネットワークの平均通信量は，DRBD は 317Mbps，dirty-block-copy は 390Mbps，ピーク時には両方とも，700Mbps 前後の帯域を利用していたことから，同期用ネットワークの帯域を小さくした際に性能に与える影響が大きいことがわかる．

次に，図 2 に遅延によるファイル I/O 性能への影響を示す．遅延 0ms の場合の性能と比較して，DRBD は 10ms で 73%, 20ms で 58%，dirty-block-copy は 10ms で 87%，20ms で 81%の性能に低下していた．DRBD と，dirty-block-copy で遅延の影響による性能低下率が異なる原因として，基本性能が異なること以外に，同期プロトコルの違いが大きな要因として考えられる．dirty-block-copy では，仮想マシン同期機構と協調して動作するため，仮想マシンとストレージの複製後，同期完了通知を 1 度受信すればよいのに対し，DRBD は，仮想マシン同期機構と DRBD が，それぞれ異なる同期プロトコルでバックアップ側と同期を行うため，仮想マシンとストレージの複製のために，同期完了通知をそれぞれで受信する必要がある．そのため，遅延の影響を大きく受けてしまったと考えられる．

以上より，WAN 環境で，仮想マシンとストレージを複製するためには，転送量を増加することなく，ストレージを複製する手法を仮想マシン同期機構に組み込むことが，低帯域・高遅延なネットワークに対応するための重要な課題であることがわかる．

3. 提案手法

本節では，仮想マシンとストレージを継続的に複製するシステムにおいて，データ転送量を削減するために考案した，I/O エミュレーションのロギングリプレイによるストレージ複製手法について述べる．

3.1 I/O エミュレーションのロギングリプレイ

本研究では，プライマリの仮想マシンモニタのストレージ I/O エミュレーションをバックアップの仮想マシンモニタでも同様に実行させることで，プライマリとバックアップのストレージを複製する手法を提案する．

ゲスト OS がストレージ書き込み命令を発行すると，仮想マシンモニタに処理が依頼される．仮想マシンモニタは，ゲスト OS の書き込み命令を解析し，実際のストレージに書き出す処理を実行する．その際に，書き込む元データはゲスト OS のメモリである．そのため，仮想マシンを複製するシステムにおいて，ストレージデータを，DRBD の利用や，メモリと同様に転送する手法などで，複製しようとすることは，メモリとブロックで同一データを 2 重転送することになり，転送量の増加につながってしまうと考えられる．

そこで，本論文では，ストレージデータを 2 重転送することなく，ストレージの複製を実現するために，プライマリの仮想マシンモニタの I/O エミュレーションのログを基に，バックアップの仮想マシンモニタで同様の I/O エミュレーションをリプレイすることで，同様のストレージ書き込みを再現する手法を提案する．I/O エミュレーションをリプレイするために必要なデータは，書き込むデータ，書き込む場所（ストレージのセクタ番号など），書き込むサイズ，その他（仮想マシンモニタ固有の情報など）である．ここで，ストレージに書き込むデータをそのまま転送すると，大きな転送量が発生してしまうが，前述したとおり，書き込むデータはゲスト OS のメモリが基となっており，仮想マシンの複製により既に，バックアップに転送済であるため，転送する必要がない．そこで，本手法では，書き込むデータをそのまま転送するのではなく，書き込むデータの基となったゲスト OS のメモリページアドレスを転送している．バックアップの仮想マシンモニタはページアドレスを基に，ストレージに書き出すべきデータをゲスト OS のメモリから参照することで，同様のストレージ書き込みを実現することができる．

3.2 I/O エミュレーションのロギングリプレイが可能な複製のタイミング

プライマリとバックアップで同様のストレージ書き込みを実現するには，ダーティページを利用した書き込みを行う直前に複製を行う必要がある．以下で，その理由について述べる．

Vol.2011-ARC-195 No.16
Vol.2011-OS-117 No.16
2011/4/13

I/O エミュレーションのロギングリプレイを実現するためには，プライマリとバックアップが，ストレージ書き込みを行う際に利用するメモリが完全に一致している必要がある．プライマリの仮想マシンモニタがストレージ書き込みを行うときに利用するメモリは，前回複製時から変更が生じたダーティページと，変更のない非ダーティページの 2 つに分けることができる．非ダーティページは，既にバックアップの仮想マシンモニタに転送済みのメモリであり，また，バックアップの仮想マシンモニタ上のゲスト OS は待機中であり，メモリデータに変更が起きないため，必ずプライマリとセカンダリのメモリデータは一致する．そのため，プライマリが非ダーティページを利用したストレージ書き込み後，そのメモリに変更が生じても，バックアップにはストレージ書き込みで利用したメモリが存在するため問題が生じない．一方，ダーティページの場合は，ページアドレスのみを転送しても，プライマリとバックアップのページアドレスが指し示すメモリデータは一致しない．また，ダーティページを利用した書き込み後，そのダーティページに変更が生じた場合，そのデータは失われてしまう．そのため，ダーティページを利用した書き込みをする場合は，書き込みを行う前に必ずバックアップに転送する必要がある．これにより，プライマリでダーティページを利用した書き込みが発生しなくなるため，プライマリがストレージ書き込みを行う時に利用するメモリデータはプライマリとバックアップで一致しているので，I/O エミュレーションのロギングリプレイによるストレージ複製が可能となる．

4. 実　　装

提案手法を，オープンソースの仮想マシンモニタ KVM/QEMU[1)10)] を用いて実装した．さらに，先行研究[17)] で開発した仮想マシン同期機構に組み込むことで，同期的に仮想マシンおよびストレージを複製している．また，ストレージ書き込みもしくはネットワークアウトプットを契機に同期しているため，I/O エミュレーションのロギングリプレイを行うことが可能なタイミングとなっている．以降で，本システムの基盤である仮想マシンモニタ KVM/QEMU についての説明と，提案手法の実装について述べる．

4.1 KVM/QEMU の I/O エミュレーション

KVM (Kernel-based Virtual Machine)[10)] は，ホスト型仮想マシンモニタとして Linux に実装されており，ハードウェアによる仮想化支援機構や，I/O のエミュレーションに QEMU[1)] を利用することで，完全仮想化を実現している．

KVM/QEMU の構成と I/O 処理の流れを図 3 に示す．ゲスト OS が，I/O を発行すると，プロセッサの仮想化支援機構によりトラップされ，処理が KVM に移行する．KVM は

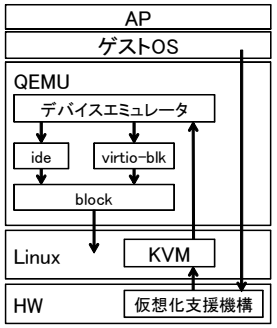

図 3 KVM/QEMU の構成と I/O 処理
Fig. 3 Architecture of KVM/QEMU and I/O processing.

I/O の解析を行い，処理を QEMU へ依頼する．QEMU はエミュレーションにより I/O をシステムコールに変換し，ホスト OS に実行を依頼し，実際のデバイスに I/O 処理が行われる．次に，QEMU の I/O エミュレーションについて，ゲスト OS がストレージ I/O を発行したときの流れについて述べる．ゲスト OS がストレージへの I/O を発行すると，利用しているデバイス (virtio-blk, ide など) に対応して， I/O エミュレーションが行われる．各デバイスドライバで処理が行われた後に，block レイヤに処理が移行し，共通処理が行われる．block レイヤでは，I/O を即座に実行せずに，リクエストをキューにため，リクエストを処理するための非同期 I/O スレッドを作成した後に，デバイスドライバに処理を戻す．非同期 I/O スレッドは，処理をスケジューリングされると，キューからリクエストを取り出し、ホスト OS に書き込みを依頼する．I/O が完了すると，利用しているデバイスドライバに応じたコールバック関数が呼び出され，後処理が行われる．

4.2 I/O エミュレーションのロギングリプレイの流れ

提案手法の実装を図 4 に示す．(1) プライマリは I/O ロギングで，ストレージ書き込みのエミュレーションログの保存を行う．また，同期のタイミングになると，同期機構に同期処理の開始を通知する．(2) 同期機構では，仮想マシン状態とログの転送を行い，(3) バックアップは，ログを基に，同様のエミュレーションをリプレイし，ストレージ書き込みを実行する．書き込み完了後，同期機構に通知し，(4) バックアップからプライマリに，仮想マシンおよびストレージの同期完了を通知し，(5) プライマリは，途中であったストレージ書き込みを実行している．

Vol.2011-ARC-195 No.16
Vol.2011-OS-117 No.16
2011/4/13

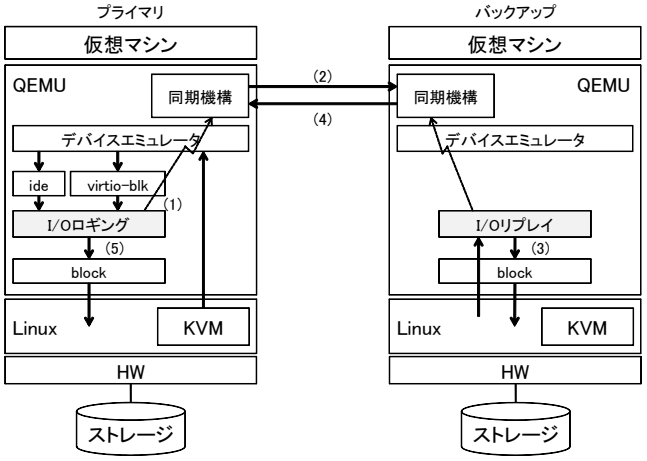

図 4 KVM/QEMU に実装した I/O エミュレーションロギングリプレイ機能の概観
Fig. 4 Architecture of Logging-Replay implemented with KVM/QEMU.

本機能は，QEMU のデバイスエミュレータレイヤと block レイヤの間に実装した．QEMU の I/O エミュレーションにおいて，各デバイスドライバから，block レイヤに処理が移行する直前にロギングを行うことにより，ゲスト OS が利用するデバイス (virtio, ide) に関わらず利用が可能という利点がある．

4.3 ロギングするデータ

ストレージ I/O の種類は，読み出し処理と書き込み処理の 2 つがあるが，今回バックアップでリプレイさせる処理は書き込み処理のみであるので，書き込み処理をリプレイするために必要なデータについて述べる．本実装で，I/O エミュレーションのロギングを実行するポイントは，図 4 で示したとおり，QEMU の I/O エミュレーションにおいて，各デバイスドライバから，block レイヤに処理が移行する直前である．ストレージ書き込みのエミュレーションを block レイヤに依頼するために必要な情報は，ストレージに書き出すデータの元である仮想マシンメモリへのポインタ，書き込む場所を示すセクタ番号などから成る．提案手法で述べた仮想マシンのメモリページアドレスをロギングするために，仮想マシンメモリへのポインタをページアドレスに変換して保存する必要がある．これは，QEMU の関数 $qemu\_ram\_addr\_from\_host\_nofail()$ で実現可能であり，また，バックアップがリプレイを行う際に，ページアドレスを仮想マシンメモリへのポインタに変換する際は，関数 $qemu\_get\_ram\_ptr()$ を利用すれば良い．以上により，バックアップがストレージ書き込みエミュレーションのリプレイを行うために，必要なデータは仮想マシンメモリのページアドレス，書き込む場所を示すセクタ番号である．ロギングするこれらのデータのサイズは，ストレージデータと比較して十分小さいため，大幅に転送量を削減できる．

4.4 バックアップでのストレージ書き込み

バックアップでのストレージ書き込みは，I/O エミュレーションのログを基に実行される．4.1 節で述べたとおり，書き込みリクエストはキューにためられ，非同期的に実行される．しかし，次の同期が始まる前に，全書き込みを完了させる必要があるため，非同期ではなく，同期的に実行する必要がある．また，書き込みが完了するとデバイスエミュレータのコールバック関数が呼び出され，後処理が行われるが，デバイス状態が更新されてしまうため，バックアップではコールバック関数を呼び出す必要がない．以上の問題に対応するため，バックアップでストレージ書き込みをする際，I/O エミュレーションのロギングリプレイ機能用にダミーのコールバック関数を用意して，書き込みが完了した際に，呼び出されるように実装した．バックアップは，実行した全書き込みに対応するダミーのコールバック関数が呼び出されることで，書き込みの完了を判断し，プライマリに同期完了を通知する．また，コールバックにより，書き込みの成否を判断し，書き込みが失敗した場合は，同期を停止することができる．

4.5 ダーティページを利用したストレージ書き込みの判別

本実装では，ロギング機能により，書き込みに利用するメモリページアドレスを保存しているため，それを基に，ダーティページを利用した書き込みかどうかの判別を，ダーティビットを利用して行うことができる．ダーティビットは，前回同期時から，メモリに変更がないかページ毎にビットで表しており，ページアドレスで参照可能である．これにより，本機構は，ライブマイグレーションなどの仮想マシンおよびストレージを複製が必要な技術に容易に適用可能となっている．

5. 実　　験

本節では，仮想マシンおよびストレージを同期するシステムにおいて，同期データ転送量および同期しているゲスト OS のファイル I/O 性能について，実験した結果について述べる．本システムは，同期用ネットワークでつながれた，プライマリ・バックアップ構成の 2 台の PC サーバから成る．また KVM を利用しているため，PC には仮想化支援機構が必要である．実験を行った環境を表 1 に示す．WAN を模擬した環境を構築するために，

Vol.2011-ARC-195 No.16
Vol.2011-OS-117 No.16
2011/4/13

表 1 実験環境
Table 1 Experimentation Environment.

サーバ	HP DL360G6
CPU	Intel Xeon Quad Core 2.66GHz x 4
メモリ	96GB
同期用ネットワーク	1GbE
仮想マシンモニタ	KVM (kernel 2.6.33), qemu 0.14
ホスト OS	Debian Lenny
ゲスト OS	Debian Lenny
メモリ	1GB
仮想 CPU 数	1

ethtool および netem により，帯域および遅延を制御している．また，本実験で測定した転送量は，プライマリのゲスト OS 上で，iozone ベンチマークを実行してから，終了するまでに，プライマリ・バックアップ間で発生した転送量を示している．iozone の測定パラメタはファイルサイズを 128MB，レコードサイズを 32KB として，バッファリング書き込みを実行している．本実験では，比較のために，提案手法を含め 3 つのストレージ複製手法を仮想マシン同期機構に組み合わせて，それぞれ実験を行った．ストレージの複製を I/O エミュレーションのロギングリプレイにより実現しているものを logging-replay，DRBD の C モード[*1]を利用しているものを DRBD，ストレージに書き込むデータを継続的にバックアップに転送する機能を実装したものを dirty-block-copy として，それぞれ示している．

5.1 転送量削減効果

仮想マシンとストレージを複製するために，必要なデータ転送量を図 5 に示す．図 5 のデータは，100 回同期させた時の平均値を示している．logging-replay により，ストレージ同期のために必要なストレージ情報のデータ転送量が，dirty-block-copy と利用して，99%削減することができ，仮想マシンとストレージを合わせた転送量も 50%程度削減することができた．dirty-block-copy では，ストレージデータを 4KB 単位で転送しているのに対し，logging-replay では，仮想マシンメモリのページアドレスが 8B しかなく，その他のデータも同様に小さいため，大きな削減効果を得ることができた．

次に，各ストレージ複製手法を利用した時の，iozone 実行時に仮想マシンおよびストレージを複製するために発生した，1 秒あたりのデータ転送量を図 6 に示す．logging-replay

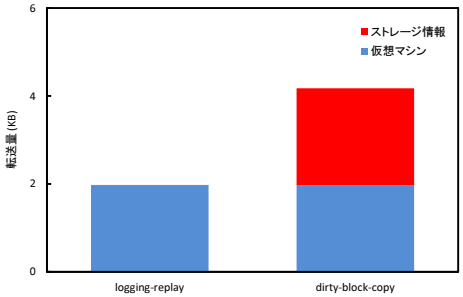

図 5 仮想マシンとストレージを複製するために必要なデータ量
Fig. 5 Replication data size of VM and storage.

は，ピーク時の転送量は変わらないが，全体的に転送量が小さく抑えられていることがわかる．同期のための総データ転送量は，DRBD，dirty-block-copy を利用する場合と比較して，logging-replay を利用した場合は，44%，60%削減できた．dirty-block-copy の総転送量が，比較的大きいのは，同期オーバヘッドにより，iozone の実行時間が長いため，プロセスが汚したメモリを余計に多く転送することになったためだと考えられる．また，同期用ネットワークの平均通信量は，logging-replay が 195Mbps, DRBD が 317Mbps，dirty-block-copy が 390Mbps であり，提案手法により，ストレージ同期のためのデータ転送量を 99%削減したことにより，dirty-block-copy の 50% 程度に通信量を抑えることができている．各手法とも，ピーク時に 700Mbps 程度の通信量が発生しているが，これは最初に多くのメモリを汚して，ストレージ書き込みを行う直前に同期を行うため，仮想マシン同期のための転送量が多く占めており，各手法で差がでなかったためと考えられる．

5.2 WAN を模擬した環境での性能評価

提案手法が，WAN を模擬した環境に置いて，DRBD，dirty-block-copy の欠点を補うことができたかどうか確認するための実験を行った．

まず，帯域の影響を測定するために，同期用ネットワークの帯域を ethtool により，変更させた場合のファイル I/O 性能を図 7 に示す．帯域を 1Gbps から 100Mbps にすることで，DRBD と dirty-block-copy は 30%以下に性能低下しているのに対し，logging-replay は 31%程度に抑えることができた．また，logging-replay は，100Mbps では，DRBD と比較して，1.35 倍の性能を得ることができたのに対し，1Gbps ではほぼ同程度の性能であり，転送量削減の効果を得ることができなかった．これは，プライマリ・バックアップのスト

⋆1 DRBD のレプリケーションは 3 種類のモードをサポートしている．C モードは，プライマリとバックアップの両方の書き込みが終わった時点で同期完了とする．

Vol.2011-ARC-195 No.16
Vol.2011-OS-117 No.16
2011/4/13

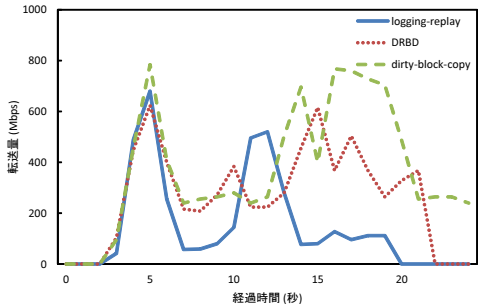

図 6 各複製手法における同期用ネットワークの通信量
Fig. 6 Network traffics with each replication technique.

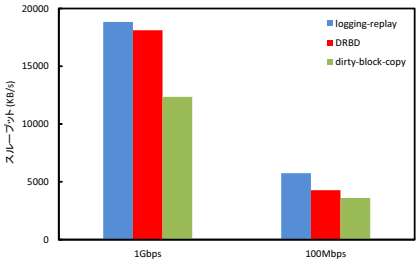

図 7 各帯域での iozone（バッファリング書き込み）の結果
Fig. 7 Results of iozone (buffered write) with each bandwidth.

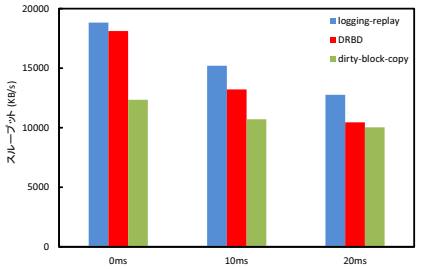

図 8 各遅延での iozone（バッファリング書き込み）の結果
Fig. 8 Results of iozone (buffered write) with each latency.

レージ書き込み方法の違いが原因として挙げられる． logging-replay の場合は，バックアップでストレージ書き込みを行い，プライマリに同期完了を通知し，プライマリがストレージ書き込みを行うため，逐次的に書き込みが行われる．一方，DRBD では，プライマリとバックアップが同時にストレージ書き込みを並列に行い，両方の書き込みが完了すると，同期完了と判断している．これにより，帯域が十分にある場合は，転送量削減の効果をあまり得ることができなかったと考えられる．

次に，遅延の影響を測定するために，同期用ネットワーク間の遅延を netem により，変化させた場合のファイル I/O 性能を図 8 に示す．遅延を大きくしていくことで，DRBD が 10ms で 73%，20ms で 58%まで性能低下しているのに対し，logging-replay では 10ms で 81%，20ms で 68%に抑えることができた．また，logging-replay の性能は，DRBD の性能と比較して 0ms で 1.04 倍，10ms で 1.15 倍，20ms で 1.22 倍と，遅延を大きくしていくに連れて，より高い性能を得ることができた．一方，dirty-block-copy の性能と比較すると，遅延が 0ms の場合は 1.5 倍，10ms で 1.4 倍，20ms で 1.3 倍と，遅延を大きくしていくことで，性能向上率が低下していることがわかる．これは dirty-block-copy と logging-replay の同期プロトコルが同一なので，遅延を大きくしてくことで，転送量削減の効果が小さくなっていくことが原因と考えられる．

最後に，低帯域・高遅延なネットワークでの同期を想定し，100Mbps，遅延を 20ms として実験を行ったところ，提案手法は DRBD と比較して 1.42 倍，dirty-block-copy と比較して，1.59 倍の性能を得ることができた．

6. 関 連 研 究

本論文では，仮想マシン同期機構に対して，提案手法を適用したが，関連する分野では，ライブマイグレーションを WAN 環境で効率よく利用するための研究が数多くされている．

文献[2),12)] では，仮想マシンとストレージのライブマイグレーションを実現しているが，ストレージデータそのものを転送する必要があるため，差分転送量が大きくなってしまうと考えられる．CloudNet[16)] は，1200km 離れたデータセンタ間で，4 つの仮想マシンを同時にライブマイグレーションする実験を行った．冗長なメモリなどの転送量を削減することで，転送に必要な帯域を 50%削減している．CloudNet では，ストレージの複製に DRBD を利用しているため，提案手法を適用することで，さらに削減が可能であると考えられる．

また，ライブマイグレーションのオーバヘッドを削減する研究もいくつか行われている．CR/TR-motion[11)] では，Revirt[7)] の機構をライブマイグレーションに適用することで，定期的に転送されるデータ量を 90 %程度削減すると共に，ライブマイグレーション中のオーバヘッドも 8 %程度に抑えているが，ロックステップ方式を利用しているため，本研究の目的とは合わない．Shrinker[14)] は，集約された仮想マシンは，同一の OS やライブラリを利用しており，メモリの内容が重複していることが多い事に着目し，移動先と移動元のホストに集約された仮想マシンのメモリを管理する機構を実装し利用することで，ライブマイグレーションで利用する帯域を 30%から 40%削減している．しかし，ストレージの転送はサポートしていない．また，ポストコピー方式のライブマイグレーション[8)9)] が提案，実装されている．従来のライブマイグレーションでは同じアドレスのページを何回も送るのに対

Vol.2011-ARC-195 No.16
Vol.2011-OS-117 No.16
2011/4/13

し，ポストコピー方式では移動する瞬間にアクセスしてるページを中心に送るため，ローカリティの高いアプリケーションでは，マイグレーションにかかる時間も短く，転送されるページ数も少ないという利点がある．しかし，継続的に仮想マシンを複製するようなシステムでの利用は難しい．

7. まとめと今後の課題

本論文では，I/O エミュレーションのロギングリプレイによるストレージ複製手法を提案し，先行研究で開発した仮想マシン同期機構に実装した．実験の結果，iozone 実行時に仮想マシンとストレージを複製するための総データ転送量を DRBD と組み合わせた場合と比較して 44%，ストレージに書き込むデータを転送する手法と組み合わせた場合と比較して 60%削減できることが確認できた．さらに，WAN を模擬した環境において，本手法を利用して同期しているプライマリのゲスト OS 上で，測定したファイル I/O 性能が，DRBD，ストレージデータ転送を組み合わせて，仮想マシンとストレージを同期した場合と比較して，それぞれ，1.42 倍，1.59 倍の性能向上を得ることができた．

今後の課題として，仮想マシンメモリの転送量削減と同期回数の削減が挙げられる．実験により，帯域の影響による性能低下率はまだ高いことがわかった．同期のためのデータ転送量の 9 割を仮想マシンのメモリが占めており，削減が必要である．これは，I/O エミュレーションのロギングリプレイを応用し，バックアップ側でプライマリと同様のストレージ読み込みを行わせることで，ある程度の削減が可能であると考えている．また，高遅延ネットワークで継続的にチェックポイントを作成するためには，同期頻度を削減するための手法も検討する必要がある．

参 考 文 献

1) Bellard, F.: QEMU, a fast and portable dynamic translator, *ATEC'05: Proceedings of the annual conference on USENIX Annual Technical Conference*, CA, USA, USENIX Association, pp.41–46 (2005).
2) Bradford, R., Feldmann, E. and Schiöberg, H.: Live wide-area migration of virtual machines including local persistent state., *3rd international conference on Virtual Execution environments*, ACM Press, pp.169–179 (2007).
3) Bressoud, T. C. and Schneider, F. B.: Hypervisor-Based Fault Tolerance, *ACM Trans. Computer Systems*, Vol.4, No.1, pp.80–107 (1996).
4) Clark, C., Fraser, K., Hand, S., Hansen, J. G., Jul, E., Limpach, C., Pratt, I. and Warfield, A.: Live Migration of Virtual Macihines, *2nd USENIX Symposium on NetworkedSystems Design and Implementation*, Boston, MA, USA, pp.273–286 (2005).
5) Cully, B., Lefebvre, G., Meyer, D., Feeley, M., Hutchinson, N. and Warfield, A.: Remus: High Availability via Asynchronous Virtual Machine Replication, *5th USENIX Symposium on Networked Systems Design and Implementation*, pp.161–174 (2008).
6) DRBD: http://www.drbd.org/.
7) Dunlap, G.W., King, S.T., Cinar, S., Basrai, M.A. and Chen, P.M.: Revirt: enabling intrusion analysis through virtual-machine logging and replay, *5th symposium on Operating systems design and implementation*, Boston, MA, USA, pp.211–224 (2002).
8) Hines, M.R. and Copalan, K.: Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning, *international conference on Virtual execution environments*, pp.51–60 (2009).
9) Hirofuchi, T., Nakada, H. and Sekiguchi, S.: Enabling Instantaneous Relocation of Virtual Machines with a Lightweight VMM Extension, *10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pp.73–83 (2010).
10) Kivity, A., Kamay, Y., Laor, D., Lublin, U. and Liguori, A.: kvm: the Linux virtual machine monitor, *OLS'07: The 2007 Ottawa Linux Symposium*, pp.225–230 (2007).
11) Liu, H., Jin, H., Liao, X., Hu, L. and Yu, C.: Live migration of virtual machine based on full system trace and replay, *18th International Symposium on High Performance Distributed Computing*, pp.101–110 (2009).
12) Luo, Y., Zhang, B., Wang, X., Wang, Z., Sun, Y. and Chen, H.: Live and Incremental whole-system migration of virtual macihnes using block-bitmap, *IEEE international Conference on Cluster Computing*, pp.99–106 (2008).
13) Marathon Technologies, Corp: everRun VM, http://www.marathontechnologies.com/.
14) Riteau, P., Morin, C. and Priol, T.: Shrinker: Efficient Wide-Area Live Virtual Machine Migration using Distributed Content-Based Addressing, http://hal.inria.fr/inria-00454727/PDF/RR-7198.pdf.
15) Scales, D., Nelson, M. and Venkitachalam, G.: The design and evaluation of a practical system for fault-tolerant virtual machines, *Vmware, Technical Report* (2010).
16) Wood, T. and Ramakrishnan, K.: CloudNet: A Platform for Optimized WAN Migration of Virtual Machines, *University of Massachusetts Technical Report* (2010).
17) 大村　圭，田村芳明，吉川拓哉，フェルナンド・バスケス，盛合　敏：KVM を利用した耐故障クラスタリング技術の開発，情報処理学会研究報告，Vol.2010-OS-115, No.20 (2010).
18) 田村芳明，柳澤佳里，佐藤孝治，盛合　敏：Kemari：仮想マシン間の同期による耐故障クラスタリング，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.1, pp. 13–24 (2010).