仮想クラスタ遠隔ライブマイグレーションにおけるストレージアクセス最適化機構

広渕崇宏[†] 小川宏高[†] 中田秀基[†]
伊藤智[†] 関口智嗣[†]

計算機センタやデータセンタの計算機資源の運用効率を高めるために, 仮想計算機とそのライブマイグレーション技術が注目されている. 仮想計算機を遠隔拠点に対して動的に再配置できれば, 資源運用の柔軟性向上や省電力化に寄与すると考えられる. しかし既存のライブマイグレーション技術のみでは, 複数拠点にまたがる動的な再配置を効率的・実用的に行うことができない. 本研究では仮想クラスタの遠隔マイグレーションを可能とするためにそのストレージアクセスの最適化機構を提案する. ライブマイグレーションにともなって仮想計算機のストレージも透過的に再配置することで, 周辺機器を含めた仮想計算機の実行環境全体の移動を可能にする. そして移動前後の性能低下を最小限に抑えることを目指す. プロトタイプ実装を用いて予備的な評価実験を行い, 提案手法の基本的な有効性を確認した.

A Storage Access Mechanism for Wide-Area Live Migration of Virtual Machines

Takahiro Hirofuchi ,[†] Hirotaka Ogawa ,[†]
Hidetomo Nakada ,[†] Satoshi Itoh [†] and Satoshi Sekiguchi [†]

Live migration of virtual machines is one of the important features provided by virtual machine monitors, which enables users to move a virtual machine to another physical computer without stopping its operating system. It is very useful for hardware maintenance and load balancing of hosting nodes. The practical use of this feature, however, is limited in LAN environments. In WAN environments, network latencies cause inevitable performance degradation of shared storage devices between source and destination host nodes, which are required to continue storage access of VMs while/after live migration. We propose a transparent, relocatable I/O mechanism for VM migration, which enables VM disk images to be completely migrated to remote nodes without any assumption of their internal contents and any modification of virtual machine monitors. Our prototype implementation showed that a virtual machine and its storage device were successfully migrated between different nodes.

1. はじめに

計算機センタやデータセンタの資源運用の効率性を高めるために, 仮想計算機 (VM) とそのライブマイグレーション技術が注目されている. 仮想計算機技術によって計算機資源を抽象化して論理的に分割・共有できる. さらに仮想マシンモニタ (VMM) が備えるライブマイグレーション機能によって, VM を一切停止することなく異なる物理ノード上に再配置可能になる.

例えば, VM を介したホスティングサービスにおいては, 物理ノードの負荷に応じた VM の再配置によって, 一定のサービス提供品質を維持できる. また, VM を一部の物理ノード上に集約して過剰な物理ノードの電源を停止することで, データセンタ全体の省電力化をはかれる. さらには, サービス提供を継続しながら, 物理計算機ハードウェアの点検や交換などを円滑に進めることもできる.

このようにライブマイグレーション機能は非常に有用な機能であるものの, その実用化は単一拠点内での運用にとどまっている. 遠隔拠点にまたがって利用できれば, 計算機センタやデータセンタの運用柔軟性を飛躍的に高められる. 拠点間をまたがった負荷バランスや省電力化が可能になるほか施設メンテナンスも容易になる. しかし, それらの実現は LAN 環境を前提とした現状のライブマイグレーション技術では困難である.

遠隔拠点間にわたる VM の動的再配置をとりわけ

† 産業技術総合研究所 / National Institute of Advanced Industrial Science and Technology (AIST)

困難にしている理由は, ライブマイグレーションに対応した VM のストレージアクセス手法にある. 一般的に VM をライブマイグレーションする際には, 移動元・移動先ホスト計算機両方からアクセスできる共有ストレージが必要とされる. しかし, 広域ネットワークを介して共有ストレージを構築すると, LAN 環境よりも大きなネットワーク遅延により I/O 性能が低下してしまい実用性を失う. また, 仮に異なる拠点に VM を再配置できたとしても, 依然としてストレージアクセス先を変更することができず柔軟性に乏しい.

そこで, 本研究では, VM の遠隔ライブマイグレーションに対応したストレージアクセス手法を提案する. VM の実行メモリイメージの再配置にともなって, VM のストレージ領域も透過的に遠隔拠点に対して再配置される. VM 内部の OS は移動前後においても引き続いてストレージ I/O を継続することができる. また VMM や VM 内部の OS に対して一切特別な変更を加える必要がない. 提案手法はネットワークを介したストレージアクセスプロトコルに対する透過的なプロキシおよびキャッシュ機構として実装される.

本稿では 2, 3 節で遠隔拠点間の VM 動的再配置における問題を明らかにし, 4 節で既存研究を概観する. 5, 6 節で提案手法とそのプロトタイプ実装について述べ, 7 節でまとめる.

2. VM 動的再配置とストレージアクセス

一般的に VM のライブマイグレーションを実現するためには, VMM および周辺機器 I/O 機構両方において対応が必要になる. VMM は VM メモリイメージを動的にノード間で再配置できる必要がある. VM に提供されるストレージ領域は何らかの方法で移動元ホスト・移動先ホスト双方からアクセス可能である必要がある. また VM 内部で実行中のネットワーク接続を維持するためには, 移動元・移動先ホスト双方において同一ネットワークセグメントを VM に提供する必要もある. ☆

このときストレージアクセス手法としては, VM のディスクイメージを移動元・移動先双方のホストからアクセス可能な領域に作成する. NFS[11] 等のネットワークファイルシステムや OCFS[5] や Lustre[4] 等のクラスタファイルシステム上に仮想ディスクイメージを作成し, 移動元・移動先双方のホストからマウントしておく. あるいは, iSCSI[9] などのブロックレベルの I/O プロトコルを用いて, 双方のホストから仮想ディスク領域をあらかじめ接続しておく. また VM 内部のシステムを NFS Root あるいは iSCSI Root によるディスクレス[15] な構成とし, ストレージサーバに VM 内部から直接アクセスする場合もある.

ライブマイグレーション時には VMM は次のように動作することで, 実行メモリイメージを再配置する.[3]

(1) 移動元ホストの VMM は, 移動先ホストに対して VM の実行に必要なメモリ領域を予約する.
(2) 予約したメモリ領域に対して移動元ホストで実行している VM メモリイメージのコピーを開始する.
(3) メモリイメージのコピー中にも VM 内の OS は動作しており, この間更新されたメモリ領域もさらに移動先ホストへコピーすることを繰り返す.
(4) 移動元ホストで VM を停止し, CPU 状態や残りのメモリイメージを移動先ホストへ転送する.
(5) すべての実行状態が転送できたら, 移動先ホストで VM の実行を再開する.

このとき第 3 段階までは, 移動元ホストを介して周辺機器 I/O が行われ, 第 5 段階以降は移動先ホストを介して周辺機器 I/O が行われる.

3. 遠隔ライブマイグレーションにおける問題

以上の仕組みを遠隔拠点間のライブマイグレーションに適用することを考える. VMM による実行メモリイメージのコピーは, 移動元・移動先ホスト双方の VMM 間で通信接続性があれば基本的には可能である. VM のメモリサイズやメモリページの更新頻度などに依存するものの, 今日の WAN において 1Gbps 程度の帯域を確保できる場合もあることを考えれば, 実用的な時間内でコピー可能であると考える. コピーされるメモリページの大半は, 第 2 段階開始直後におけるチェックポイント時のものであり, バースト的な転送で処理できる. したがって, ネットワーク遅延に対して TCP のバッファサイズを十分多くなるように最適化しておけば, 広帯域に見合うだけの速度で迅速にコピーが完了するはずである. また VM に対して割り当てられたメモリ領域のうち実際に使用しているページのみをコピーすればよい. キャッシュ領域として使用しているメモリページはマイグレーション直前に解放することで, データ転送量をさらに小さくできる.

移動元・移動先双方で同一のネットワークセグメントを提供するためには, イーサネット VPN が使用できる. 我々が開発しているマルチサイト仮想クラスタ[14),16)] においては, 複数拠点にまたがる仮想クラスタを構築し大規模ノード群を管理するために, イーサネット VPN によって分散仮想ノードを単一クラスタとしてのビューの元に容易に取り扱えるようにしている. ゆえに我々の実装においては開発済みのコンポーネントを利用できる.

☆ VM 内部において再配置を検知し OS やアプリケーションが独自に対応するのであればこの限りではない. 異なるネットワークセグメントに移動しても, DHCP による IP アドレスの更新とアプリケーションレベルでのネットワーク接続の再確立を行ったり, Mobile IP を使って同一 IP アドレスを維持することもできる.

しかし, ストレージアクセスについては大きな問題があり, 既存技術での解決は難しい. 第一に, 移動元・移動先拠点間で WAN を介して VM のストレージ領域を共有した場合, ストレージサーバに対するネットワーク遅延が増大してしまい, WAN 経由で遠隔ストレージアクセスする際に十分な I/O 性能を維持できない. 一般に大きな I/O バッファサイズでのシーケンシャル I/O が行われる場合は, 高遅延環境下でも TCP のバッファサイズを最適化することにより十分な I/O 性能を発揮できることが知られている[13]. しかし, VM のストレージ領域として使われた場合, 必ずしもシーケンシャルな読み込みや書き込みを VM が要求するだけではなく, 内部の OS やアプリケーションに依存して, ランダムな読み込みや書き込みも数多く要求される. また一度に発行される I/O バッファサイズは, 内部の OS のストレージドライバや上位のアプリケーション, VMM に付随するブロックデバイスドライバなどに依存し, 高遅延環境下でも十分な I/O 性能を発揮できる I/O バッファサイズで要求を常に発行するわけではない. ゆえに, これらの手法においては遠隔ライブマイグレーションにともなって VM のストレージ I/O 性能を維持することが困難である.

第二に, VM のストレージデータは移動元の拠点に残り続けてしまう. 移動元の拠点のストレージサーバが常に維持される必要があり, また WAN 経由での通信接続性も継続的に保持しなければならない. マイグレーションによって移動元拠点の設備を柔軟にメンテナンスすることが難しくなり, また移動後の VM を運用する上での信頼性も損なう可能性がある.

また, 別の手法として遠隔拠点間で VM のストレージデータを同期的に常にミラーしておく手法が考えられるものの, 実用面でのコストを考慮すれば適用が難しい. ある拠点がホストするすべての VM ストレージを同期的にミラーするためには, すべてのデータを格納できるだけの大容量のストレージが移動先拠点においても確保される必要がある. またミラーリングのために WAN を介したデータ転送量が常に発生してしまう. 計算機センタやデータセンタのように VM を大規模に運用する環境において一般的に利用するにはコスト面において問題がある.

4. 関 連 研 究

遠隔拠点間における VM のマイグレーションに取り組んだ研究は少ない. 文献 1) においては, Xen において特殊なバックエンドストレージドライバを作成し, 先にストレージデータのすべてを移動先ホストへコピーし, その後メモリの再配置を開始する. 2 節で述べた第 3 段階終了までは, 移動元ホストでは VM が動作し続けており, 次々に新しいデータがストレージに書き込まれ続ける. これらのデータも移動先ホストへ転送される. またこのときバックエンドストレージドライバにおいて意図的に VM の書き込み速度を低下させることで, 有限時間内にデータ転送が完了するようにしている. メモリ再配置の完了と移動先での VM の起動以前に, ストレージの移動が完了しているという特徴を持つ. ライブマイグレーション完了後には I/O 性能の低下が一切存在しない. 一方で, マイグレーション開始から移動先ホストで VM が起動するまでの時間が長く, 意図的な書き込み速度低下による性能低下も存在する. VM の起動ホストをすばやく切り替えることができないため, きめの細かい負荷バランスや省電力化には不向きであると予想する.

また文献 8) では, あらかじめ VM の雛形となるディスクイメージを移動元・移動先双方に用意しておき, マイグレーション時にはその変更差分のみを移動先ホストからオンデマンドに取得する. VMware GSX Server のサスペンド・レジューム機能を利用して静的なマイグレーションのみに対応し, システムコールの上書きを用いて実装されている. あらかじめ雛形ディスクイメージを用意することでデータ転送量を軽減できる. しかし移動先となりうる拠点すべてに対してあらかじめ共通の雛形イメージをコピーしておかなければならず, あらかじめ移動先拠点が知られていない場合や共通となるべき雛形イメージが存在しない場合には不向きである. また雛形イメージを各拠点にコピーしておくことは, ソフトウェアライセンス上制限される場合もある.

文献 7) では, VMware Workstation のサスペンド・レジューム機能を利用した静的なマイグレーションを, 広域分散ファイルシステムである Coda[10] との組み合わせにおいて実現している. VM の仮想ディスクを特殊なブロックデバイスドライバによって Coda サーバ上に小さく分割したファイルとして保存する. そしてその分割ファイルごとに移動元・移動先ホストであらかじめキャッシュしておいたり, オンデマンドな更新差分の取得を可能にしている. しかし Coda がディスコネクト・オペレーションに対応しているとはいえ, VM が拠点外に存在するかもしれない特定の Coda サーバに常に依存し続けるのは, 運用における柔軟性や信頼性を損なってしまう. さらに VM のディスク I/O がさまざまなソフトウェアコンポーネントを経ることになりそのオーバヘッドによる性能低下が危惧される.

VMware 社による Storage VMotion[12] は, VM を起動したまま仮想ディスクを異なるストレージサーバ上に透過的に移動する技術である. 仮想ディスクのスナップショット機能やロギング機能を利用して実装されている. 単一拠点内で用いられることを想定したものであり, 遠隔拠点間のストレージ再配置は対応していない。

また, 以上に述べたいずれの手法も特定の VMM に依存した実装である. さまざまな VMM が活発に開発

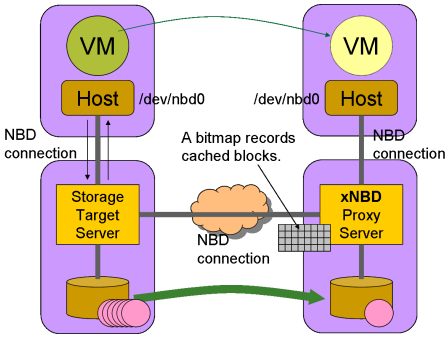

図 1 提案アーキテクチャ概要

されている現状においては, 特定の VMM のみに対応した手法は望ましくない.

5. 提 案 手 法

以上の問題点をふまえて, 遠隔ライブマイグレーションを実環境において実用化するためには, 我々は VMM が備えるメモリイメージの動的再配置だけではなく, VM のストレージ領域の透過的な動的再配置が不可欠であると考える. VM が異なる拠点のホストに移動することと同時に, その VM がアクセスするストレージ領域も異なる拠点に移動させる. VM が依存する資源を移動元に残さずにすべて移動先に移すことで, VM 運用の柔軟性と信頼性を向上させ, WAN を経由した遠隔資源アクセスによる性能低下を避けることができる. また以上の仕組みは VM に対して透過的に実装されて, VM 内部のシステムが動作を中断することなく動的移動を完了できる.

5.1 提案手法とその基本動作

我々が提案する VM ストレージの動的再配置機構の概要を**図 1** に示す. 提案システムは VMM に対しては標準的なブロックデバイスレベルのストレージサーバとして振舞いながらも, 遠隔拠点間の提案システム同士の間でメモリイメージの再配置にあわせてストレージデータのコピーを行う.

VM の移動元・移動先ホストはそれぞれ提案ストレージサーバに接続し, ホスト OS 上のブロックデバイスとして VM のストレージ領域を提供する. 例えば, 図 1 において提案システムは NBD[2] サーバとして振る舞いながら, ホスト OS 上ではそれぞれ`/dev/nbd0`として見えている. ☆1

VM が移動元ホストで動作しているときには, 一般的なブロックデバイスレベルのストレージサーバ同様に振舞う. VM 内部の OS が発行した I/O リクエストは, ローカルのストレージデバイスに対して処理される.

VM がメモリイメージの再配置を開始すると, 2 節で述べた第 3 段階まではすべての I/O リクエストは移動元ホストにおいて引き続き実行されるので, 移動元拠点に存在するストレージデバイスに対して読み書きを続ける. しかし, その後, 移動元ホストでの VM の停止を経て, 第 5 段階において移動先ホストで VM が開始されると, すべての I/O リクエストは移動先ホストにおいて発行される. この開始時点では移動先拠点のストレージデバイスには, 直前まで VM が読み書きしていた移動元のデータは存在していない.

移動先における提案システム (図 1 中 xNBD Proxy Server, xNBD プロキシ) は, VM のリクエストに対して不足しているデータをオンデマンドに移動元拠点から取得する. 同時に移動先のストレージデバイスにデータをコピーし, 最終的にはすべての I/O が移動先拠点内で完結するようにする. 移動元拠点のストレージサーバは複数のクライアント接続を同時に受け付けるように実装する. xNBD プロキシは標準的な NBD プロトコルを使用して移動元拠点のストレージサーバに接続し, NBD クライアントとして不足しているデータを取得する. ☆2

このときの詳しい振る舞いを**図 2** に示す.

- VM が移動先に存在しないしないブロックを読み込もうとした場合, 移動元拠点のストレージデバイスから遠隔読み込みを行う. 読み込んだデータを VM に返すと同時に, 移動先拠点のストレージデバイス (以降本段落では**ローカルディスク**☆3) にも保存する. 対象ブロックがキャッシュされた (移動先に存在済み) としてビットマップテーブルに記録する. ☆4
- VM がデータを書き込もうとした場合, 対象ブロックがすでにローカルディスクに存在するしないにかかわらず, ローカルディスクの対象ブロックにデータを保存する. 移動元拠点のストレージデバイスは更新しない. この操作前に対象ブロックがキャッシュされていなかったならば, 新たにキャッシュされたとしてビットマップテーブルに記録する. ☆5

☆1 後に述べるプロトタイプ実装ではプロトコルが単純な NBD を用いているものの, 将来的には iSCSI による実装を視野に入れている.

☆2 移動元拠点のストレージサーバは移動元ホストおよび xNBD プロキシの両方と接続する. しかし, 2 節で述べたメモリイメージの再配置処理における第三段階と第五段階で I/O パスが移動元ホストから移動先ホストへと切り替わるため, 同時に I/O リクエストを受けることはない. 分散ロック機構を利用せずともデータの一貫性が保たれる.

☆3 すでに VM は移動先で起動しており, VM が起動している拠点を基準に表現している.

☆4 便宜的にキャッシュという表現を用いているが, 移動先拠点のストレージデバイスは一時的なデータの保存場所ではない点に注意されたい.

☆5 書き込みリクエストの対象領域の始端オフセットや終端オフセットがブロック単位でない場合は, 移動元拠点から始端ないし終端

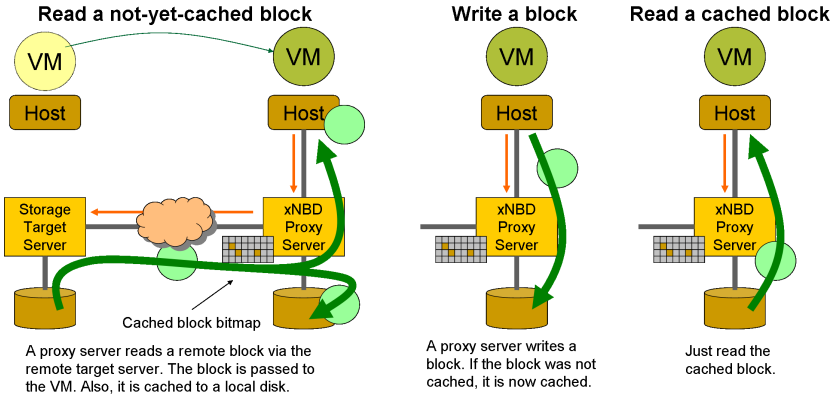

図 2　基本動作概要

- VM がすでに移動先に存在済みのブロックを読み込もうとした場合, ローカルディスクの対象ブロックからデータを読み込む. 移動元ストレージデバイスから読み込む必要はない.

ストレージデバイスへのキャッシュは, 一定サイズのブロック単位で行われる. このブロックサイズは, 処理の効率化のため NBD サーバが提供するディスクブロックサイズの整数倍に設定される.

VM が移動先で I/O 処理を行うたびに, 徐々に移動先にキャッシュされるディスクブロックが増えていく. 以上の動作に平行して, 残りのブロックを移動元拠点からローカルディスクにコピーすることで, 最終的にすべてのブロックが移動先拠点にキャッシュされた状態になり, VM のストレージ領域の再配置が完了する. xNBD プロキシは移動元拠点のストレージサーバに対する接続を終了する. 以降 xNBD プロキシは標準的なストレージサーバとして振舞う. さらに同様の仕組みで他の拠点に移動できる.

5.2　再配置アルゴリズムの最適化

計算機センタやデータセンタなどにおいて大規模な VM 集合を拠点間で動的に再配置するためには, 効率的なストレージ動的再配置機構が要求される. 迅速に完了し WAN 経由のデータ転送量を低く抑えることが求められる. さらに再配置時の性能低下も最小限にする必要がある. そこで先に述べた基本動作に加えて, 仮想ディスクの hot-spot 領域解析によるプロアクティブなキャッシング, シーケンシャルリードに対する先読み, データ転送の圧縮などを的確に組み合わせる予定である.

6.　プロトタイプ実装

提案システムの基本的な妥当性を検証すべくプロトタイプ (xNBD) を開発した. 5.1 節に述べた基本動作部分を実装しており, NBD プロトコルのマルチクライアント対応ストレージサーバおよびプロキシサーバから成る.

ブロックを先に取得する必要がある.

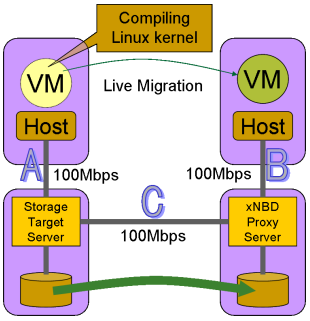

図 3　プロトタイプ動作確認環境

その基本動作を LAN 環境 (**図 3**) において確認した. VMM として Xen 3.0 を設定した 2 台の計算機を用意し, それぞれにプロトタイプサーバから NBD デバイス (`/dev/nbd0`) を提供する. VM 上に CentOS 5 をインストールし, Linux カーネルのコンパイルを行いながらライブマイグレーションを行った. 動作確認中の図 3 における各リンク A, B, C における NBD プロトコルの通信量をそれぞれ**図 4**, **図 5**, **図 6** に示す. また xNBD プロキシにおけるキャッシュ済みブロックの割合を**図 7** に示す☆. あらかじめカーネルキャッシュをクリアした上でコンパイルを開始している.

実験開始直後 VM は図 3 の左側ホストで動作している. ディスク I/O においてはカーネルコンパイルにより発生したオブジェクトファイルの書き込みによる通信量が大きい. その後, 時刻 06:46:00 付近においてライブマイグレーションを行ったため, 前後して VM のディスク I/O がリンク A からリンク B に切り替わっている. また同時に xNBD プロキシによる移動元ストレージからの読み込みトラフィックも発生して

☆ メモリイメージの再配置にともなうトラフィックは別のリンクに分離して影響を排除している.

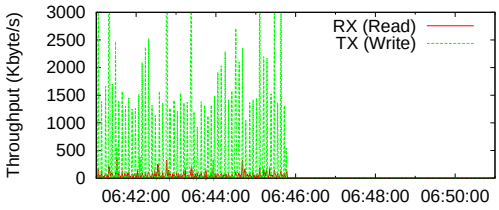

図 4 NBD プロトコル通信量 (図 3 中リンク A). 書き込み (TX) が大半.

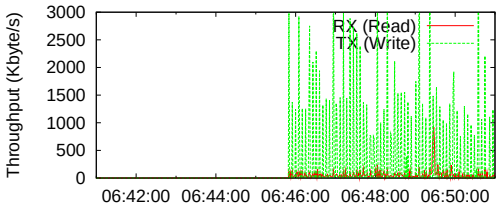

図 5 NBD プロトコル通信量 (図 3 中リンク B). 書き込み (TX) が大半.

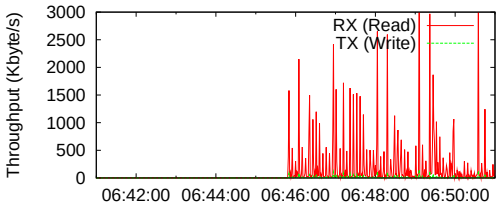

図 6 NBD プロトコル通信量 (図 3 中リンク C). xNBD プロキシの読み込み (RX) が大半.

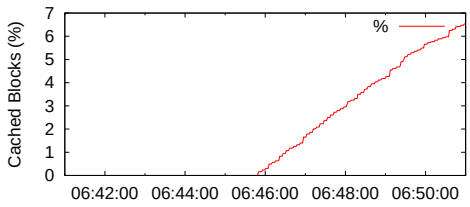

図 7 キャッシュ済みブロックの割合

いる. カーネルコンパイルにより新しいブロックを読み込んだり, 新しいブロックにデータを書き込むだびにキャッシュ済みブロックの割合が増えていく. 図中の時刻内には表れていないが, 残りの未キャッシュブロックを強制的にキャッシュして以降は, 一切 xNBD プロキシによる移動元ストレージからの読み込みは発生しなくなり, 正しく動作している.

今後 WAN 環境を再現した環境で詳しい性能評価に取り組む. リンク C にネットワークエミュレータを挿み込み実験を行う予定である.

7. ま と め

VM の遠隔ライブマイグレーションを実現するために, 遠隔拠点間での VM ストレージの透過的な動的再配置機構を提案した. プロトタイプ実装を通して提案手法の基本部分の動作を確認した. 今後は WAN 環境における性能評価やより高度な再配置アルゴリズムの検討に取り組む.

謝辞 本研究は科研費 (20700038) および CREST の助成を受けたものである.

参 考 文 献

1) Bradford, R., Kotsovinos, E., Feldmann, A. and Schiöberg, H.: Live wide-area migration of virtual machines including local persistent state, *Proceedings of the 3rd International Conference on Virtual Execution Environments*, ACM Press, pp. 169–179 (2007).
2) Breuer, P. T., Lopez, A. M. and Ares, A. G.: The enhanced network block device, *Linux Journal*, Vol.Issue 73 (2000).
3) Clark, C., Fraser, K., Hand, S., Hansen, J. G., Jul, E., Limpach, C., Pratt, I. and Warfield, A.: Live migration of virtual machines, *Proceedings of the 2nd conference on Symposium on Networked Systems Design and Implementation*, USENIX Association, pp. 273–286 (2005).
4) Cluster File Systems, Inc.: Lustre: A Scalable, High-Performance File System, (White Papaer) (2002).
5) Fasheh, M.: OCFS2: The Oracle clustered file system, version 2, *Proceedings of the Linux Symposium*, Linux Symposium, pp. 289–302 (2006).
6) Hirofuchi, T.: A relocatable storage I/O mechanism for live-migration of virtual machines over WAN, USENIX Annual Technical Conference (Poster) (2008).
7) Kozuch, M., Satyanarayanan, M., Bressound, T. and Ke, Y.: Efficient state transfer for Internet suspend/resume, Technical Report IRP-TR-02-03, Intel Research Pittsburgh (2002).
8) Sapuntzakis, C. P., Chandra, R., Pfaff, B., Chow, J., Lam, M.S. and Rosenblum, M.: Optimizing the migration of virtual computers, *ACM SIGOPS Operating System Review*, Vol. 36, No. SI, pp. 377–390 (2002).
9) Satran, J., Meth, K., Sapuntzakis, C., Chadalapaka, M. and Zeidner, E.: Internet Small Computer Systems Interface (iSCSI), RFC 3720 (2004).
10) Satyanarayanan, M., Kistler, J. J., Kumar, P., Okasaki, M.E., Siegel, E.H. and Steere, D.C.: Coda: A highly available file system for a distributed workstation environment, *IEEE Transactions on Computers*, Vol. 39, No. 4, pp. 447–459 (1990).
11) Sun Microsystems: NFS: Network file system protocol specification, RFC 1094 (1989).
12) VMware, Inc.: VMware Storage VMotion: Non-disruptive, live migration of virtual machine storage, (Product Datasheet).
13) 山口実靖, 小口正人, 喜連川優: iSCSI 解析システムの構築と高遅延環境におけるシーケンシャルアクセスの性能向上に関する考察, 電子情報通信学会論文誌 D-I, Vol. 87, No. 2, pp. 216–231 (2004).
14) 中田秀基, 横井威, 江原忠士, 谷村勇輔, 小川宏高, 関口智嗣: 仮想クラスタ管理システムの設計と実装, 情報処理学会論文誌コンピューティングシステム, Vol. ACS19, pp. 13–24 (2007).
15) 小川宏高, 中田秀基, 広渕崇宏, 伊藤智, 関口智嗣: 仮想クラスタのステートレス化のための Rocks 5 ディスクレス化機構, 並列/分散/協調処理に関するサマー・ワークショップ (SWoPP 佐賀 2008) (to be published) (2008).
16) 広渕崇宏, 中田秀基, 横井威, 江原忠士, 谷村勇輔, 小川宏高, 関口智嗣: 複数サイトにまたがる仮想クラスタの構築手法, 第 6 回先進的計算基盤システムシンポジウム SACSIS 2008, pp. 333–340 (2008).