2008－HPC－117 (2)
2008／10／15

# 仮想クラスタ遠隔ライブマイグレーションにむけた仮想計算機ストレージの透過的再配置機構の評価

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においては計算資源の運用効率向上と電力消費の削減が強く求められている．そこで我々は計算資源の需要と供給に応じた柔軟な仮想計算機群の拠点間再配置を実現すべく，仮想クラスタの遠隔ライブマイグレーション技術の実用化を目指している．しかし，LAN 環境を想定した既存の仮想計算機技術ではストレージアクセス手法に問題を抱えており，WAN 環境においてライブマイグレーションを効率的に行うことが困難である．我々はこの問題を解決すべく先行研究において仮想計算機ストレージの透過的かつ動的な拠点間移動手法を提案した．提案機構はブロックレベルのストレージ I/O プロトコルに対するターゲットサーバおよびプロキシサーバとして振る舞い，仮想計算機の動作に支障を与えることなく透過的に仮想ディスクの拠点間再配置を完了する．本稿では，WAN 環境を想定した各種実験をとおして得られた，提案手法の詳細な性能評価を報告する．結果，提案手法はライブマイグレーションにともなう仮想計算機 I/O の性能低下を抑えながら，仮想計算機ストレージを遠隔拠点間で再配置できることがわかった．

# The Evaluation of a Transparent Storage Relocation Mechanism for Wide-Area Live Migration of Virtual Machines

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

We are developing a multi-site virtual cluster system that seamlessly integrates distributed computing resources in many sites. It massively improves management flexibility and power efficiency beyond resource limitations in a single site. One of its key features is *wide-area* live migration of virtual machines; administrators can dynamically relocate virtual machines to other sites without any downtime of their hosting services. Due to network latencies, however, existing virtualization technologies do not provide an efficient storage access mechanism for relocatable virtual machines in WAN environments. In the previous work, we proposed a novel storage access system incorporated into live migration technologies. It works as a target and proxy server of a block-level I/O protocol, and transparently copies virtual disk blocks among source and destination sites. This paper focuses on detailed evaluations of the proposed system under an emulated WAN environment. Results showed that our system alleviated I/O performance degradation involved in wide-area live migration, enabling complete relocation of virtual disks between remote sites.

## 1. はじめに

計算機センタやデータセンタにおいて仮想化技術の導入が進みつつある．物理的な計算機資源を抽象化し論理的に分割・共有可能になり，資源利用効率を向上させ運用コストを低減できる．特に注目を集める仮想化技術として仮想マシンモニタ (VMM) が備えるライブマイグレーション機能が挙げられる．仮想計算機 (VM) を一切停止することなく異なる物理ノード上に移動できる．

我々は計算資源運用コストの削減や電力消費の効率化を実現するため，拠点横断的な仮想化資源運用技術の確立に取り組んできた[13),14)]．特に，ライブマイグレーション機能を利用して遠隔拠点にまたがって VM を動的に再配置できれば，拠点横断的な負荷バランスや省電力化が可能になり，また施設メンテナンスも容易になると考えている．

しかし現状では VM のライブマイグレーション機能の実用化は単一拠点内での利用にとどまっている．ライブマイグレーションに際しては，VM の継続的な I/O を可能にするため，移動先・移動元双方のホスト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，遠隔拠点間にまたがって共有ストレージを構築すると LAN 環境よりもはるかに大きいネットワーク遅延を原因として十分な I/O 性能を得ることがで

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

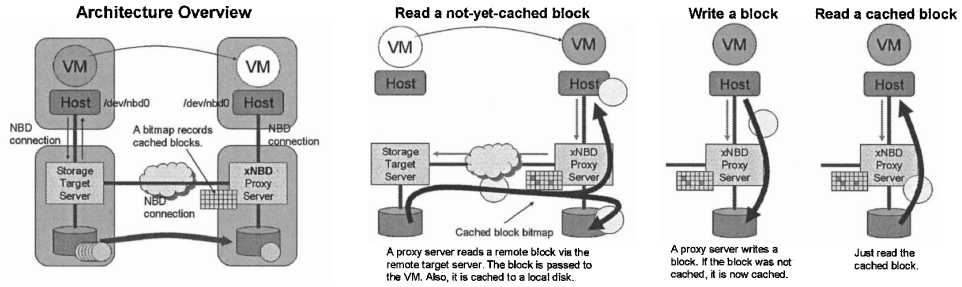


**図 1** 提案機構とその基本動作概要

きない．また VM が拠点外のストレージサーバに常に依存するため運用の柔軟性にも乏しい．

そこで我々は先行研究[12]において，遠隔拠点間にまたがった，VM ストレージの動的な再配置機構を提案した．再配置機構は，ブロックレベルのストレージ I/O プロトコルのターゲットサーバおよびプロキシサーバとして動作する．VM の I/O にともなうオンデマンドなデータコピー，およびバックグラウンドでのデータコピーによって，VM の動作に支障を与えることなくストレージの再配置を完了する．WAN 環境下での VM 再配置に取り組んだ既存研究[3),7),8),11]と比較しても，VM を一切停止させることなく迅速な実行ホストの変更を可能にし，VMM に非依存な機構となる点に優位性がある．先行研究においては，VMM と連動して提案機構が正しく動作することを初期段階の実装によって確認した．

我々は提案機構の実装をさらに進め，WAN をエミュレーションした環境で評価実験を行っている．本稿では，評価実験の詳細について述べその性能特性を明らかにする．2 節では提案動作の基本動作を説明し，先行研究では触れなかった実装について述べる．3 節では評価実験について述べる．4 節でまとめと今後の課題を述べる．

## 2. VM ストレージ移動手法

本節では文献[12]で述べた提案機構の基本動作について簡単にまとめる．また先に触れなかった実装の詳細についても述べる．

### 2.1 基本動作

提案手法はブロックレベルのストレージ I/O プロトコル (iSCSI[9] や NBD[4] プロトコル) のストレージサーバとして振舞いながらも，VMM による VM 再配置と連動して遠隔拠点間でストレージデータのコピーを行う．VM はホスト OS 上のブロックデバイス (/dev/nbd0 等) を介して仮想ディスクを読み書きする．

VMM はライブマイグレーションを開始すると，VM の実行メモリイメージを移動先ホストにコピーし始める．すべてのメモリイメージがコピーできた時点で移動元ホストでの VM の実行を停止し，移動先ホストにて実行を再開する．この時点以降，VM のすべての I/O は移動先ホストを介して行われる．このとき，提案機構のプロキシサーバは，移動先拠点において VM の I/O リクエストに応じて，移動元拠点のターゲットサーバからストレージ I/O プロトコルによってデータを取得していく．さらに移動先拠点にデータを保存 (キャッシュ) していく☆．

**図 1** において，読み込み及び書き込みリクエストそれぞれにおける処理を示す．未キャッシュのブロックに対する読み込みリクエストは，移動元拠点ストレージサーバからデータを取得し，キャッシュ済みとマークする．書き込みリクエストは，移動先拠点にデータを保存し，対象ブロックが未キャッシュだったならキャッシュ済みとマークする．キャッシュ済みのブロックに対する読み込みリクエストは，単に移動先拠点からデータを読み込めばよい．一度キャッシュされた領域については移動元への遠隔読み込みは発生しない．VM によって一度アクセスされた領域から徐々に再配置が完了していく．

### 2.2 バックグラウンドコピー

上述した VM の I/O に応じたオンデマンドなデータ再配置に加えて，未キャッシュなブロックはバックグラウンドでもコピーされる．最終的に仮想ディスクのすべての領域がキャッシュされると，仮想ディスクの再配置は完了し，プロキシサーバから移動元拠点ターゲットサーバへのストレージ I/O プロトコル接続は終了する．バックグラウンドコピーは，VM が特定のブロックに初めてアクセスする前にあらかじめブロックをコピーしてキャッシュしておくことで，移動元への遠隔読み込みを防ぐ側面もある．

### 2.3 実　　装

現在我々は比較的単純な NBD プロトコルに対して提案機構を実装している．ネットワーク遅延がある WAN 環境でも性能低下を軽減するために I/O リクエストの多重キューイング機能を備え，図 1 のターゲットサーバにあるように複数のクライアントからの同時接続をサポートする．またプロキシサーバもクライア

☆ 本稿では単にキャッシュと述べた場合プロキシサーバに保存されたディスクブロックを示す．カーネルメモリ中のキャッシュを指さない．

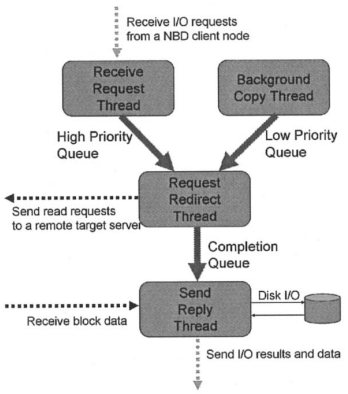


**図 2** 提案機構におけるプロキシサーバ実装

ントからは通常のターゲットサーバとして見えるので，さらに次の拠点に再配置することもできる．仮想ディスクファイルやキャッシュファイルに対しては `mmap()` を用いることでメモリコピーを最小限に抑えている．バックグラウンドコピーはオンデマンドな I/O 処理を妨げないように低い優先度で処理する．

### 2.3.1 I/O リクエスト処理

**図 2** にあるように，多重キューイングのサポートおよび低い優先度のバックグラウンドコピーを実装するために，プロキシサーバとして動作するときには 4 つのスレッドが動作している．

まず VM が I/O リクエストを発行すると，クライアントからのリクエスト受信スレッドがリクエストを受信する．移動元拠点からの遠隔データ読み込みの必要性を一定のキャッシュブロックサイズ (4KB) ごとに判定したうえで，優先リクエストキューに中間リクエストをキューイングする．一方，バックグラウンドコピースレッドは外部プログラムから先読みすべきブロック番号を受け取り，非優先リクエストキューに中間リクエストをキューイングする．

中間スレッドは優先リクエストキューから先に中間リクエストを取り出し，必要があれば移動元拠点のターゲットサーバに読み込みリクエストを送信し，完了スレッドへ中間リクエストをキューイングする．優先リクエストキューが空であれば非優先リクエストキューから中間リクエストを取り出し，未キャッシュブロックであることを確認して，同様に処理する．

完了スレッドは中間リクエストを取り出し，必要があれば移動元拠点のターゲットサーバから先程投げた読み込みリクエストのデータを受信する．書き込みや未キャッシュブロックの読み込みの場合は，ディスクへデータを保存する．最後に，I/O リクエストの結果 (と読み込みリクエストの場合には読み込みデータ) をクライアントに返信する．

### 2.3.2 バックグラウンドコピー処理

バックグラウンドコピーにおいては，任意の未キャッシュブロックの取得を外部プログラムから命令可能にしている．仮想ディスクの先頭から未キャッシュブロックを順にコピーするという単純な動作だけではなく，利用頻度の高いブロックから先にコピーすることも可能である．

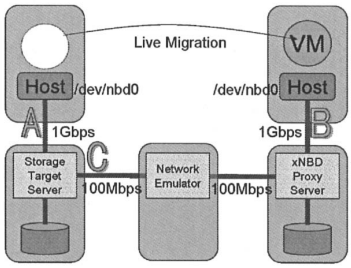


**図 3** 実験環境

実際，我々は仮想ディスク上に作られた ext3 ファイルシステム[5),10)] を解析することによって，利用頻度が高い領域を判別し優先的にコピーする仕組みを実装している．ext3 はディスク領域を複数のグループに分割して，関連性の高いブロックを同じグループ内に保存することで，キャッシュ効率を高めディスクヘッド移動を減らすよう試みる．同じディレクトリに属するファイルは同じグループに，同じファイルに属するブロックは同じグループに属することが多い．また使用中のディスクブロックを管理するために，inode ブロックビットマップおよびデータブロックビットマップとしてディスク上にビットマップを保存している．

そこで，ターゲットサーバで I/O リクエスト領域の統計を取っておくことで，直近にアクセス頻度の高いグループからバックグラウンドコピーに取り掛かれるようにしている．さらにターゲットサーバにおいて残された ext3 イメージのビットマップブロックを解析して，データが保存されているブロックから優先的にコピーを開始できる．

## 3. 評価実験

実験環境を**図 3** に示す☆．WAN を介して VM 移動元・移動先の 2 拠点が存在する状況を想定する．拠点内は LAN 環境を想定して Gb Ethernet によりノード間を接続する．拠点間は帯域 100Mbps を設定しネットワークエミュレータ (Linux カーネルの netemu 機能) によりネットワーク遅延 (RTT20ms) を発生させる．最初に NBD ドライバに直接的に I/O 命令を発行して提案機構の基本的な性能特性を確認する．次に VM 内部における I/O 性能を検証する．特に一般的に遠隔ストレージアクセスに最も不向きとされるメールサーバを想定した実験を行う．

### 3.1 基本的な I/O 性能特性

最初にカーネルによるバッファリングやキャッシングの影響を排して提案機構自体の基本的な性能特性

☆ VM ホスト計算機: Intel Xeon 2.8GHz, メモリ 2GB
ストレージサーバ計算機: AMD Opteron 250, メモリ 4GB

を確認する。一般的にカーネルは読み込みに際してはデータの先読みを行ったり，書き込みデータを一旦内部でバッファリングして，それらのデータをメモリ上でキャッシュする．またブロックデバイスに発行されるリクエストがハードウェアのブロックサイズ区切りとなるよう調整している．しかし、Linux の Direct I/O 機能を用いるとこれらの処理を経ることなく直接的にブロックデバイスに I/O 命令を発行できる。

ホスト OS に対して接続された NBD のデバイスファイルに対して，リクエストサイズを変えながら以下の条件のもとで，Direct I/O による `read()`/`write()` に要する時間を計測した．Linux カーネルが NBD に対して発行する上限サイズ 128KB までで計測している．

- **local**: 移動元ホストでターゲットサーバに対する性能を計測する．LAN 環境での利用に相当する．
- **remote**: 移動先ホストで提案機構を用いずに移動元拠点のターゲットサーバに遠隔アクセスした場合の I/O 性能を計測する．WAN 経由での遠隔アクセスとなる．図 3 リンク B をプロキシサーバを経由せずに直接ネットワークエミュレータにつなぐ．
- **proxy caching**: 移動先ホストで提案機構を経由して I/O 性能を計測する．未キャッシュ状態のブロックに対する I/O 性能を計測するために実装を改変しており一度アクセスされたブロックをキャッシュ済みとしてマークしない☆．同じ領域に何度アクセスしても常に未キャッシュ状態のブロックとして動作する．
- **proxy precached**: 移動先ホストで提案機構を経由して I/O 性能を計測する．キャッシュ済み状態のブロックに対する I/O 性能を見るために，あらかじめすべてのブロックをキャッシュ済みとしておく．

**図 4** において，`read()` に要する時間に対して LAN 環境 (**local**) および事前キャッシュした場合 (**proxy precached**) に差は見られない．一度キャッシュされたブロックは，WAN を経由して移動元ターゲットサーバにアクセスすることなく取得できる．そのため，LAN 環境でのアクセスと全く同様の I/O 性能を得られ，遠隔アクセスする場合 (**remote**) よりも大幅に性能が向上している．**remote** が示すように移動元ストレージへの遠隔アクセスは RTT 分さらに完了時間を要してしまう．また多くの場合 LAN に比べて小さい帯域を利用してデータ転送しなければならない．**proxy caching** では未キャッシュのブロックを取得するために遠隔アクセスとほぼ同様の完了時間を要する．プロキシサーバ内の処理において，遠隔取得したブロックを再度クライアントに転送するので若干完了時間が増加している．

**図 5** において，`write()` に要する時間に対しては **local, proxy precached** および **proxy caching** についてほとんど差はない．プロキシサーバは，対象ブロックがキャッシュ済みかどうかかかわらず直接キャッシュファイルに書き込む．原則遠隔アクセスが発生せず LAN 環境でのアクセスと同様の I/O 性能を得られ，遠隔アクセス (**remote**) に比べ大幅に性能が向上している．**proxy caching** において書き込みサイズが 1KB および 2KB のときに遠隔読み込みが発生しているのは，キャッシュブロックサイズとして設定した 4KB よりも小さいサイズの書き込みであるため，プロキシサーバが対象ブロックの 4KB 分のデータを先に読み込んだ上で，内部で書き込みデータで上書きしてディスクに書き戻しているからである．仮想ディスク上にファイルシステムを作成するのであれば，ファイルシステムのブロックサイズをあらかじめ 4KB にしておくことで，この遠隔読み込みを防ぐことができる．

以上より，提案機構は，未キャッシュのブロックを読み込む場合を除けば，LAN 環境でのストレージアクセスと全く同様の I/O 性能を有し，遠隔アクセスに比べて大幅に性能を改善できることが確認された．また，一般的にカーネルは内部に読み込み頻度が高いディスクブロックのキャッシュを持つことによって，I/O 性能の向上を図っている．ライブマイグレーション直前に VM の内部で動作していたプログラムが使用するデータは，移動直後にもカーネルキャッシュ上に存在する可能性がある．そのためプロキシサーバから未キャッシュブロックを読み込む頻度は必ずしも高くはないと

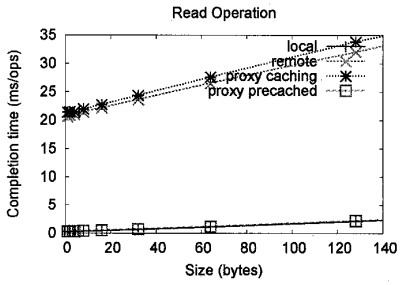


**図 4** Direct I/O `read()` 完了時間．local および proxy precached はグラフ下部でほぼ重なっている．

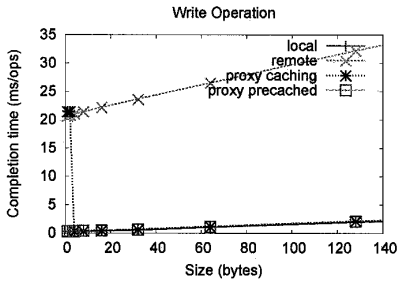


**図 5** Direct I/O `write()` 完了時間．local, proxy caching および proxy precached はグラフ下部でほぼ重なっている．

☆ データの一貫性が保証されないため本実験のみで用いる．

表 1 シーケンシャル I/O 性能 (KBytes/s)

| | Read | Write |
|---|---|---|
| **local** | 71435 (0%) | 83403 (23%) |
| **remote** | 4501 (0%) | 12879 (3%) |
| **proxy precached** | 72320 (0%) | 82224 (23%) |

括弧内はゲスト OS における CPU 使用率 (%)

表 2 ランダムシーク性能 (ops/s)

| | Random |
|---|---|
| **local** | 7644 (1%) |
| **remote** | 159 (0%) |
| **proxy precached** | 7198 (1%) |

括弧内はゲスト OS における CPU 使用率 (%)

いえる.

### 3.2 VM 内部での性能

次に VM で動作する OS 上で性能評価を行った. Xen 3.0[1] を用いてメモリ 512MB を割り当てた VM を作成し Linux をインストールした. Linux をインストールした仮想ディスクとは別に, 実験用のファイルを作成する仮想ディスクを VM に追加した. ext3 によりフォーマットおよびマウントしている.

#### 3.2.1 Bonnie++

ベンチマークプログラム Bonnie++1.03c[2] により計測したシーケンシャル I/O 性能を**表 1** に示す. 遠隔アクセス (**remote**) の場合においては, ネットワーク遅延 (RTT20ms) および狭い帯域 (100Mbps) が原因で低い I/O 性能 (読み込み 4.5MBytes/s, 書き込み 12.9MBytes/s) しか達成できていない. しかし提案機構によりキャッシュされた状態 (**proxy precached**) においては, LAN 内での利用 (**local**) と同様に高い I/O 性能 (読み込み 72.3MBytes/s, 書き込み 82.2MBytes/s) を得られる.

**local** および **proxy precached** において, ホスト OS においてプロセスおよびカーネルスレッドの CPU 使用率を観察すると, Xen のバックエンドブロックデバイスドライバ (xvd) がこの実験中は 90%を超えていた. 実験環境においては xvd におけるメモリコピー処理がボトルネックとなっている.

**表 2** に示したランダムアクセス性能において, 提案機構は遠隔アクセス (**remote**) における値 (159ops/s) よりも遥かに良い値 (7198ops/s) を示す. Bonnie++ におけるランダムアクセス性能の計測は, 対象となるファイルをランダムに `lseek()` したのち 90%の割合で `read()` および 10%の割合で `write()` を発行する. 対象となるファイルは OS のメモリサイズに対して 2 倍の大きさ (1GB) で作成されるために, カーネルキャッシュのみならずディスクへのアクセスも頻発する. したがってネットワーク遅延を介した遠隔アクセスにおいては非常に性能が悪くなる一方, 提案機構による改善効果が顕著に現れている.

**表 3** では, メールサーバプログラムを想定したファイル操作性能を示す. Bonnie++のファイル操作ベンチマーク機能を用いている. ひとつあたり 10KB の 102400 個のファイルを 100 個のディレクトリの下に均等になるように作成 (`creat()`, `write()`, ファイルとディレクトリの `fsync()`, `close()`) する. すべてのファイルをランダムな順番にアクセス (`stat()`, `open()`, `read()`, `close()`) する. その後すべてのファイルをランダムな順番で削除 (`unlink()`, ディレクトリの `fsync()`) する. 一般に小さなデータサイズの読み書きを大量に同期的に行うアプリケーションは, 高遅延環境下での遠隔ストレージアクセスに不向きとされる. 作成されるファイルの総量 (10KB*102400 個) はゲスト OS に割り当てたメモリサイズ (512MB) の約 2 倍となるよう設定している. そのため, 約半分のファイルはカーネルメモリ内にキャッシュされず, 実際にディスクにアクセスして読み込まれる.

表 3 ファイル操作性能 (ops/s)

| | Create | Access | Delete |
|---|---|---|---|
| **local** | 699 (1%) | 3856 (1%) | 426 (1%) |
| **remote** | 15 (0%) | 65 (0%) | 21 (0%) |
| **proxy precached** | 674 (1%) | 4107 (1%) | 425 (1%) |

括弧内はゲスト OS における CPU 使用率 (%)

ファイルの作成および削除時にはひとつのファイルを処理するごとにディスクへの同期的な書き込みが発生し, ランダムなアクセス時にもディスクからの読み込みが多くの場合発生する. 遠隔アクセス (**remote**) においては, それらの処理ごとにネットワーク遅延に起因する少なくとも 20ms の完了時間を要するために, 操作性能が大きく低下している. しかし提案機構においては一度キャッシュされたブロックであれば LAN 環境と同様の操作性能を得られている.

#### 3.2.2 未キャッシュブロック読み込みとバックグラウンドコピー

次に未キャッシュブロック読み込みの影響およびバックグラウンドコピーの効果を確認した. 3.1 節において確認したように, プロキシサーバで未キャッシュ状態であるブロックを読み込もうとする際には遠隔アクセス同様の長い完了時間を要する. このことが影響して, ライブマイグレーション以後にデータがプロキシサーバにキャッシュされるまでに, VM 内で動作するアプリケーションの実行性能が低下してしまう可能性がある. 特に, アプリケーションが I/O インテンシブなプログラムであって, カーネルキャッシュに収まりきらないほどの大量のデータに対して継続的にアクセスし続ける場合に顕著であると考えられる. さらにランダムアクセスにおいてはカーネルによる先読みが動作せず, より非効率であるといえる. 一方, これらの問題に対して, バックグランドコピー機能がブロックを事前にキャッシングすることで, 性能低下を抑制できると期待される.

まず 3.2.1 節のファイル操作性能測定時と同様のファイル群をあらかじめ移動元ターゲットサーバ上に作成しておく☆. 次に移動先ホストからプロキシサーバに接

☆ プロキシサーバに接続した移動先ホスト上 VM で実験に用いるファイルを作成すると, そのブロックは読み込みテスト前からキャッシュ済みとなってしまう

表 4 プロキシサーバ経由ファイル操作性能 (ops/s)

| | | Access |
|---|---|---|
| proxy | bgcopy off | 42.2 |
| proxy | bgcopy on | 1957.2 |
| proxy | precached | 2050.1 |

続して, 未キャッシュ状態 (**bgcopy off**) そしてキャッシュ済み状態 (**precached**) のファイル群のランダムなアクセス性能を 3.2.1 節のファイル操作測定時と同様に計測した. いずれも計測前にゲスト OS およびホスト OS のカーネルメモリ中のキャッシュを破棄している. さらに未キャッシュ状態の計測において開始直後からバックグラウンドコピーも動作させた場合 (**bgcopy on**) も実験した. ext3 ファイルシステム上で使用中のブロックを分析し, それらを順次プロキシサーバへコピーしている.

**表 4** に実験結果を示す☆. 未キャッシュ状態の読み込みは平均 42.2ops/s しか実行できず非常に遅い操作となってしまう. しかしバックグラウンドコピーによって同時並行的にキャッシュすれば平均 1957.2ops/s と, 事前キャッシュ済み状態における読み込み (2050.1ops/s) と同等の性能まで向上している.

未キャッシュ状態の読み込みにおいて, バックグラウンドコピー無効時には WAN を経由する遠隔読み込み方向のトラフィックは 200-600KBytes/s 程度を推移する. カーネルの先読み機能はランダムアクセスに対しては有効ではなく, また NBD ドライバのリクエストキューイングサイズは最大でも 128KB であり高遅延環境下で可用帯域を埋めるには十分な大きさではない. 一方, バックグラウンドコピー動作時には, 継続的に 12MBytes/s 程度のトラフィックが発生しており, 可用帯域を利用して事前キャッシュが効率的に行われていることが確認できる.

### 3.2.3 カーネルコンパイル

実際のアプリケーションによる評価として, 提案機構を用いて VM 上で Linux カーネルのコンパイルを行った☆☆. 移動先ホスト上の VM でコンパイルを行う. 仮想ディスクに対して, 提案機構を用いず遠隔アクセスした場合 (**remote**), 提案機構を用いた場合 (**proxy**) を比較する.

結果を**表 5** に示す. 遠隔アクセスに比べて書き込み操作が速い分, 提案機構 (**bgcopy off**) において改善が見られる. バックグラウンドコピーが有効であれば, 事前にすべてのブロックがキャッシュされていた場合と同等の所要時間で完了している. しかしカーネルコンパイルは CPU インテンシブな処理であるので, 各方式における完了時間にはメールサーバ型のアプリケーションで見られたほどの劇的な差は現れなかった.

## 4. ま と め

本稿では, VM ストレージの動的かつ透過的な再配置機構の評価について述べた. 今後は提案機構を我々が開発中の仮想クラスタシステムに適用していく. 評価実験で得られた結果を以下にまとめる.

表 5 Linux カーネルコンパイル時間 (s)

| | | Elapsed Time |
|---|---|---|
| remote | | 6572 |
| proxy | bgcopy off | 5393 |
| proxy | bgcopy on | 5012 |
| proxy | precached | 4958 |

提案機構におけるプロキシサーバはブロックデバイスの特性として, 書き込みおよびキャッシュ済みブロックの読み込みは LAN 環境と同等に高速, 未キャッシュブロックの読み込みは WAN 経由の遠隔読み込みが発生するゆえに低速, という特徴を持つ. しかし一度読み込めばプロキシ上でキャッシュされ次回の読み込みは高速である.

提案機構を使用する VM 上では, データの書き込みや一度プロキシサーバでキャッシュされたデータの読み込みは LAN 環境と同等に高速である. 特に多数のファイルにアクセスし同期的な I/O を頻繁に行うタイプの I/O インテンシブなアプリケーションにおいては遠隔アクセスするよりも改善は顕著である.

そのようなアプリケーションがライブマイグレーション後も移動元拠点で作成されたファイルに大量にアクセスし続けるのであれば, 未キャッシュブロックの遠隔読み込みが遅いために, 大きく性能低下してしまう. しかしバックグラウンドコピーによって可用帯域を十分利用しながら使用中のブロックを効率的にコピーすることで, 性能低下を大きく緩和できる.

**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) Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I. and Warfield, A.: Xen and the art of virtualization, *Proceedings of the nineteenth ACM symposium on Operating systems principles*, ACM Press (2003).
2) Bonnie++: http://sourceforge.net/projects/bonnie/.
3) 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).
4) Breuer, P.T., Lopez, A.M. and Ares, A.G.: The network block device (1999).
5) Card, R., Ts'o, T. and Tweedie, S.: Design and Implementation of the Second Extended Filesystem, *Proceedings of the First Dutch International Symposium on Linux* (1995).
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) Tweedie, S. C.: Journaling the Linux ext2fs Filesystem, LinuxExpo '98 (1998).
11) VMware, Inc.: VMware Storage VMotion: Non-disruptive, live migration of virtual machine storage, (Product Datasheet).
12) 広渕崇宏, 小川宏高, 中田秀基, 伊藤智, 関口智嗣: 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, 情報処理学会研究報告 (2008-HPC-116), 情報処理学会, pp. 19–24 (2008).
13) 中田秀基, 横井祐, 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).
14) 広渕崇宏, 中田秀基, 横井祐, 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).

☆ 事前にメモリキャッシュを破棄しているためキャッシュ済み状態 (**precached**) に対する結果は 3.2.1 節よりも低下している.

☆☆ Linux 2.6.24 を用い `make allmodconfig && make` と実行.