Vol.2011-OS-118 No.3
2011/7/27

ハイバネート中の仮想マシンに対する透過的アクセス機構

轟　侑樹[†1]　藤田　肇[†2]　石川　裕[†1,†2]

仮想化を用いた環境において, 仮想マシンに対してアクセス可能な状態を維持するためには常に仮想マシンを起動しておかなければならない. しかし, その仮想マシンが必ずしも常にアクセスされ続けるわけではない以上, それはリソースの無駄につながる. そこで, 本論文では利用していない仮想マシンをハイバネートすることで使用リソースを抑えながらも, 接続を検知して仮想マシンを再開することでユーザからの接続を受け付けることのできるシステムである ODRS を提案する. ハイバネートに関する処理はユーザからは透過的に行われるため, 通常の仮想環境とほぼ変わらない環境を提供できる. 更に, 仮想マシンを再開するノードは最も負荷の少ないノードが選ばれるので, 負荷分散も同時に実現される. 実験の結果, ハイバネート時の応答時間は仮想マシンのレジューム時間に依存するが, アクセスの少ない仮想マシンが多い環境においては最大で 54%のメモリ使用量を削減できた.

Transparent Access to Hibernating Virtual Machines

Yuuki Todoroki,[†1] Hajime Fujita[†2]
and Yutaka Ishikawa[†1,†2]

If we would like to make a virtual machine always accessible in a virtualization environment, the virtual machine has to keep running. However this wastes resources, when it is taken into consideration that the virtual machine is not always accessed continuously. Therefore, this paper proposes ODRS (On-demand Resume System), a cloud system which saves resources by putting unused virtual machines into the hibernation state and resuming them in response to network accesses so that they can accept connections from the Internet. This action is executed transparently to users, and therefore ODRS provides an environment just as like a normal cloud environment. In addition, because the least-loaded node is selected as a node where the virtual machine is restarted, load balancing is accomplished at the same time. When a VM is hibernating, the response time depends on a resume time of virtual machine, but memory usage was reduced by 54% at most in environment where VMs are not accessed frequently.

1. はじめに

サーバ環境において仮想化技術を用いることは, 既に一般的な手法になりつつある. 仮想化技術のメリットの内の一つは, リソースの使用率の改善である. 複数台のサーバにサーバ毎に異なるアプリケーションを動作させているような場合においては, 個々のサーバのリソース使用率は低い状態であることは多く, その場合十分にリソースを活用できているとは言いがたい. 仮想マシンを用いてそれらのサーバの機能を一つの物理マシンに統合することでリソース使用率の改善[6] を図ることができ, 結果としてサーバ維持費を削減することが可能である. 更に, 仮想マシンは相互に独立した環境であり, 一つの仮想マシンの動作が他の仮想マシンに影響を及ぼさない. よって,1 つの仮想マシンで複数のサーバのアプリケーションを動作させた場合に比べ, ある特定のアプリケーションのクラッシュが他のサーバ機能に対して影響を与えないという点でより信頼性の高いものとすることができる.

また, 異なる環境を一つの環境に共存可能[5] であり, 一つの物理マシンを複数の仮想マシンとして切り分けて提供できることから, クラウドコンピューティングにおいてもしばしば仮想化は用いられる[3]. 以上のように仮想化技術の重要性は近年増加しており, 仮想化環境を改善することは大きな意味を持つ.

複数の仮想マシンを物理マシン上に展開した仮想化環境において, 実際にサービスを提供する仮想マシンはユーザからの接続がない場合でも常に動作させておかなければならない. あまり頻繁に利用されないような状況の仮想マシン, もしくは利用に時間的な偏りがあるような仮想マシンでは, その仮想マシンが一定時間利用されない時間が存在する. 利用されていない仮想マシンも他の仮想マシンと同様, それが動作するのに必要な一定のリソースを占有し続ける.

この問題を解決するために, 本研究では仮想マシンの自動ハイバネーション, 自動レジュームを可能にする On-Demand Resume System(ODRS) を提案する. ODRS はロードアベレージ, 通信状況, 動作プロセスなどの基準から使われていない判断される仮想マシンを自動的に, 物理ノード間で共有されているネットワークファイルシステムにハイバネートする.

†1 東京大学大学院情報理工学系研究科
Graduate School of Information Science and Technology, The University of Tokyo
†2 東京大学情報基盤センター
Information Technology Center, The University of Tokyo

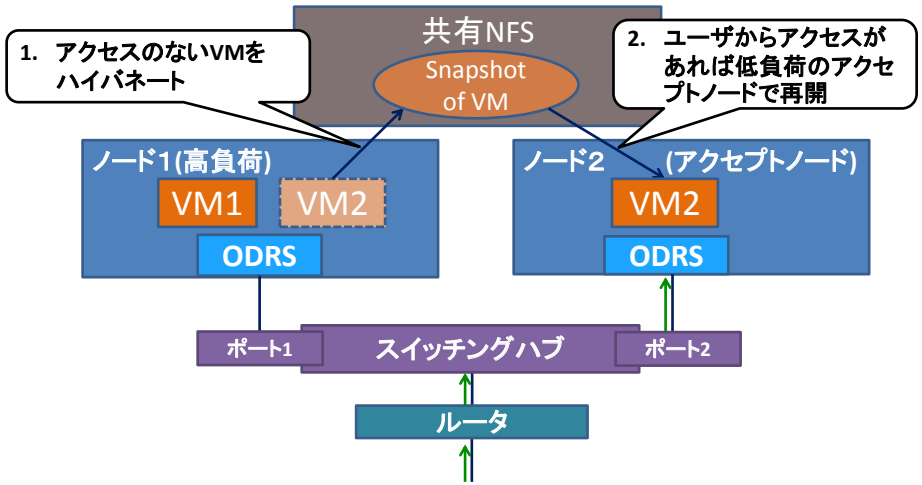

図 1　ODRS の動作概要

ハイバネートされた仮想マシンは占有されているリソースを解放するため, システム全体でのリソースの使用量を削減する. そして, ユーザからの接続に応じて, 最も負荷の低いノードで自動的に仮想マシンを再開し接続を受け付ける. この一連の処理はユーザに対して透過的であるため, ユーザは仮想マシンが常に稼動しているのと同様の方法で仮想マシンを利用することができる.

2. 提 案 手 法

ODRS は, 仮想マシンを動作させてサービスを提供している複数の物理ノードが存在する環境において, デーモンプロセスとして動作する. そして通信パケットや仮想マシンのリソース使用状況などを監視し, 自動で仮想マシンをハイバネーション, レジュームを行う. 通信パケットの監視など, 下位プロトコルでパケットを扱いたい場合には RAW ソケットを用いた読み書きを行って実現している. ユーザからの接続は TCP によるものを想定し, 仮想マシンには KVM(Kernel-based Virtual Machine)[1] を用いている.

図 1 に ODRS の動作概要を示す. ODRS における物理ノードの内, ユーザからの接続を受け付けるノードを本論文ではアクセプトノードと呼ぶことにする. 図 1 ではノード 2 がアクセプトノードになっている. ODRS は全ての物理ノードで動作しており, 自ノードで稼動している仮想マシンの通信状況, リソース使用状況を一定時間ごとに監視している. その結果から, 利用されていないと判断される図 1 の仮想マシン 2 をハイバネートする. 仮想マシンのイメージファイル, ハイバネート時のスナップショットは全ての物理ノードから共有されている NFS に保存されているため, どの物理ノードからでもそれらのファイルにアクセスできるようになっている. ハイバネーション中の仮想マシン 2 に対して接続が来た場合には, アクセプトノードとなっているノード 2 がその接続を検知する. そして, 共有 NFS に先ほど保存したスナップショットから仮想マシンの状態を復元し, 直ちに仮想マシンを再開する. 再開された仮想マシンが接続を受け付けることで, ユーザはハイバネーションを意識せずに接続することができる.

以上のようなシステムを構築する上で問題となる点は主に以下の 3 点である.

- ハイバネート中の仮想マシンに対する ARP 要求,TCP 接続要求をどう処理するか
- 負荷分散をどのように実現するか
- 仮想マシンをいつハイバネートするか

これらの問題を ODRS ではどのように解決しているかを, 本章で述べる.

2.1 ハイバネート中の仮想マシンへのアクセスを可能にする機構

まず最初の課題であるハイバネート中の仮想マシンに対するアクセスを可能にする機構について説明する. 当然, 仮想マシンがハイバネート中は接続を受け付けることは不可能である. そこで, ホストとなる物理マシンが接続を検知して, 接続があった場合には直ちに仮想マシンを再開することで接続を可能にする. TCP による接続において, 接続の最初に受け取る可能性のあるデータは二つある.ARP 要求と SYN である.

イーサネットフレームを送信する際には, 相手の MAC アドレスを知る必要がある. IP アドレスから MAC アドレスを調べる方法には ARP が用いられる. IP アドレスを元にイーサネットフレームを送信する必要のあるノードは, まずそのネットワークセグメント内に ARP 要求をブロードキャストする. ARP によって得られた IP アドレスと MAC アドレスの対応付けは,ARP キャッシュに保存され, 次回の通信の際には ARP は省略される. ARP キャッシュは一定時間通信が行われないと破棄され, その際にはまた ARP によるアドレス解決が行われる. つまり, 仮想マシンへの接続の際には,ARP キャッシュがない限りこの ARP 要求が最初に届く. ホストで動作している ODRS はこの ARP 要求を検知して, この要求に対する ARP 応答を, アクセスのあった仮想マシンの代わりに送信し, 同時に仮想マシンを再開する. 以前の接続の際に保存された ARP キャッシュが保存されている場合には,ARP が省略されるため ARP 要求は届かずに, 直接 IP パケットが配送される. TCP の接続の開始時に

おいてはそのパケットは SYN であり, この SYN を ODRS が検知して仮想マシンの再開を行う.

しかしながら, 仮想マシンの再開には一定の時間がかかるため, ただ再開するだけでは再開途中の仮想マシンは最初の SYN を適切に処理することができず, パケットをロストしてしまう. そのため,ODRS は接続の最初の SYN を保存しておき, 仮想マシンの再開が完了した時にもう一度ネットワークに送信する. この SYN はイーサネットレベルで全く同一のパケットを送信するので, このパケットを受け取った仮想マシンは, 接続元のクライアントと正常にコネクションを確立することができる.

2.2 負荷分散を実現する機構

ODRS では, アクセプトノードを一定時間ごとに, その時点で最も低負荷のノードに受け渡すことで負荷分散を実現している. ある時刻におけるアクセプトノードは一定時間アクセプトノードの役割を果たした後, 他の物理ノードのリソース使用状況を調べる. このリソース使用状況の監視には SNMP[4] を用いている. 他の物理ノードの負荷を, ロードアベレージ, メモリ使用量からスコアを用いて評価し, 最も負荷の低いノードを次のアクセプトノードに選ぶ. ロードアベレージは低いほど, メモリ使用量は少ないほどスコアは低くなる. ロードアベレージとメモリ使用量を個別に評価し, その内の高いほうがそのノードの評価値として用いられる. これは, メモリ使用量とロードアベレージのどちらかが偏って高い場合には, その片方がボトルネックとなり, 十分なサービスを提供できない可能性があるからである. 次のアクセプトノードには, 他の物理ノードの内, 最もスコアの低いノードが選ばれる. アクセプトノードは全ての仮想マシンがどの物理ノードで稼働しているかの情報を保持しており, この情報を参照して稼働していない仮想マシンへの接続だけを検知してその再開の処理を行う. アクセプトノードの役割が移動する場合には, その時点でのアクセプトノードは次のアクセプトノードに対して, その情報を送信する. そして新しいアクセプトノードは, 他のノードに対してアクセプトノードが変更されたことを通知する. これは仮想マシンがハイバネートした際にアクセプトノードに対してそれを通知する必要があるためである.

以上のようにアクセプトノードは一定時間ごとに変更されるが, この時次のような問題が生じる. ハイバネート中の仮想マシンに対して接続がある場合に, ルータに ARP キャッシュが保存されていなければ ARP 要求がブロードキャストされるため, アクセプトノードがどのノードであっても容易にそれを検知することができる. しかしながら,ARP キャッシュが残っている場合には ARP は省略され直接 IP パケットが配送されるが, その際にどの物理ノードに配送されるかはスイッチングハブの MAC アドレステーブルに従う. これはスイッチングハブのポートと MAC アドレスの対応付けを保存しておくテーブルであり, このテーブルはポート側からイーサネットフレームを受信する度に学習される.

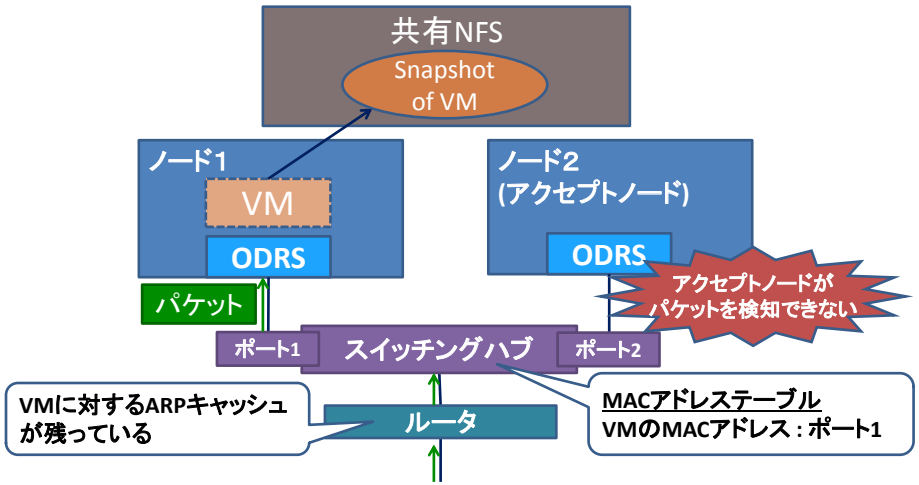

図 2 アクセプトノードがパケットを検知できない例

それを踏まえた上で, 問題となるケースを図 2 に示す. 図 2 は, ノード 1 で稼動していた仮想マシンがハイバネーションされ, 更にその仮想マシンに対するアクセスが発生した場合を示している. ルータには ARP キャッシュが保存されているために,ARP 要求は送信されず, 直接 IP パケットが配送される. 更に, スイッチングハブには MAC アドレステーブルが保存されており, 仮想マシンがハイバネートされる前に稼動していたノード 1 に仮想マシンがまだ存在していると認識している. このような場合には, 仮想マシンへのパケットはノード 1 に配送されてしまい, そのパケットをアクセプトノードであるノード 2 は検知することが出来ない. よってアクセプトノードは仮想マシンを再開することができず, このままではサービスを継続できない.

これを解決するためには, アクセプトノードに常にハイバネート中の仮想マシンに対するパケットが配送されるように MAC アドレステーブルを更新する必要がある. 具体的には, アクセプトノードの受け渡しが行われる際に, 新しくアクセプトノードとなったノードが, ハイバネート中の仮想マシンを送信元と偽装したイーサネットフレームを送信することで,MAC アドレステーブルの内容を更新する. これにより, ハイバネート中の仮想マシンに対するパ

ケットは常にアクセプトノードに届くために, それを検知してアクセプトノードが仮想マシンを再開することが可能になる.

2.3 仮想マシンを自動でハイバネートする機構

物理ノード上に展開された仮想マシンは,ODRS によりその通信状況とロードアベレージ, 動作しているプロセスを監視される. 通信状況は ODRS が常に RAW ソケットを用いた監視を行い, ロードアベレージと動作プロセスは, 仮想マシンを ssh サーバとして動作させ, ホストマシンからの ssh によるアクセスで監視する. 現在の実装では, 仮想マシン宛の TCP パケットが到着すると仮想マシン毎に保持しているタイムスタンプを更新し, また一定時間パケットが届かなければハイバネーションをする. ただしロードアベレージが事前に設定していた一定値を超えた場合にはハイバネートされなくなる. これはユーザからの接続を必要としない長時間かかる処理に対応するためである. また, 事前にプロセス名を登録しておくことで, 時間基準で動作を始めるアプリケーションなどを動作させている場合のハイバネーションを防ぐことができる. 以上の三つの基準を用いてハイバネートする仮想マシンを決定する.

そして, 仮想マシンのハイバネーション時には, その仮想マシンが稼動していたノードはアクセプトノードに対して仮想マシンが停止した旨を通知する. アクセプトノードの変更時に仮想マシン停止の通知が送信された場合, この通知は古いアクセプトノードに届く可能性がある. その場合は, そのメッセージを新しいアクセプトノードに転送することで, 正常に仮想マシンの停止を処理することができる.

3. 評　　価

3.1 実 験 環 境

表 1 実験環境 (サーバ, クライアント共に同じ)

CPU	CPU Intel Xeon L5520 (2.26GHz, 4cores Hyper-Threading and Turbo Boost)
Memory	6GB
OS	Ubuntu 10.10(maverick)
Kernel	Linux 2.6.35-24-generic x86_64
Virtual Machine	QEMU PC emulator version 0.12.5 (qemu-kvm-0.12.5)

実験を行った環境を表 1 に示す. 3 台の同一のスペックを持ったサーバ上で ODRS を動作させ, クライアントも同一のスペックを持つマシンで実験を行った. 3 台のサーバの内,1 台は NFS サーバを兼ねる. サーバ, クライアントは同じネットワークセグメントに属しており, 相互にギガビットイーサネットで接続されている. 仮想マシンは仮想ブリッジでネットワークに接続されており,6 台の仮想マシンをサーバ上で動作させた. 仮想マシンの OS もサーバやクライアントと同様のものを用いている.

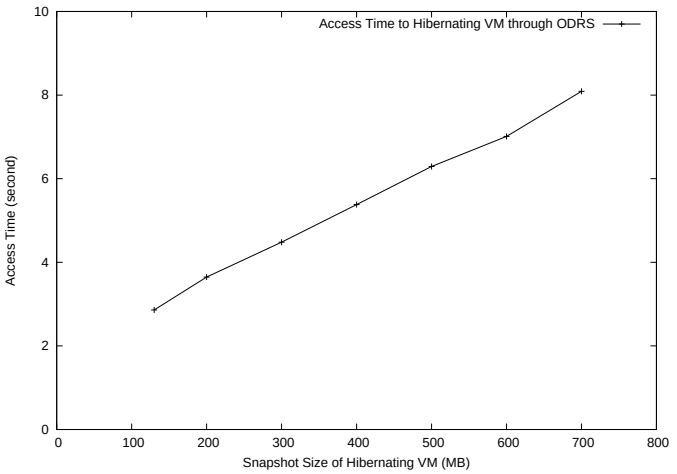

図 3 ハイバネート中の仮想マシンへのアクセス時間

3.2 アクセス時間に関する評価

ODRS における, ハイバネート中の仮想マシンへのアクセス時間に関する実験結果を図 3 に示す. 送られてきたデータをそのまま返す単純な TCP のエコーサーバを実行した状態の VM をハイバネートし, その VM に対して新たな TCP コネクションを確立し,1 バイトのデータを送信する. そして送信したデータが返ってくるまでの時間をアクセス時間として計測した. 仮想マシンのイメージデータ, スナップショットがアクセプトノードのキャッシュに残っている場合はアクセス時間に大きな影響を及ぼすが, 今回の実験ではキャッシュにそれらが存在しない場合を計測している. また,NFS サーバで仮想マシンを再開する場合はアクセス時間は短くなるので,NFS サーバ以外がサーバとなるように実験を行った.

結果として, アクセス時間はスナップショットのサイズに大きく依存したものとなっている. スナップショットが 130MB であれば 2.86 秒でアクセスできる一方, スナップショットが

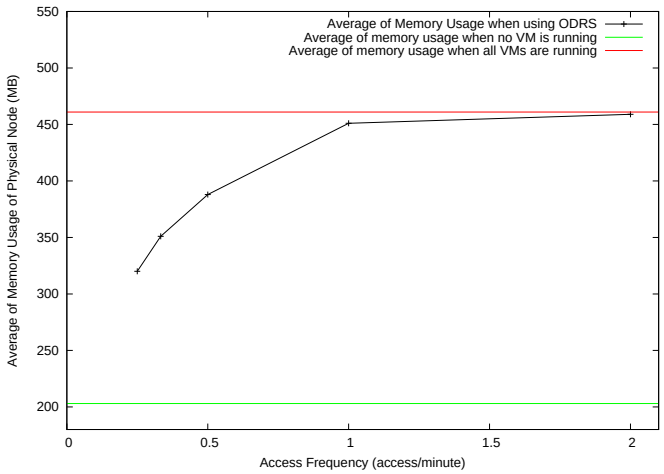

図 4 ODRS 使用時の物理ノードのメモリ使用量

700MB のサイズであれば 8.08 秒アクセスにかかってしまう．ODRS のアクセス時間の遅延の大部分は仮想マシンの再開にかかる時間によるものである．そして仮想マシンの再開時にはスナップショットから仮想マシンの状態を復元する必要があるため, スナップショットのサイズが大きければそれだけハイバネート中の仮想マシンへのアクセス時間が長くなってしまう．レジューム時間は入出力, ネットワーク帯域幅, ファイルシステム, 仮想マシンの性能, 機能にも大きく依存するため, 環境次第でアクセス時間も大きく変化することが予想される．

3.3 メモリ使用量に関する評価

ODRS を使った場合の物理ノードのメモリ使用量を図 4 に示す．ロードアベレージ, 起動プロセスによるハイバネーションの停止は考慮せず,2 分間アクセスのなかった仮想マシンをハイバネートすることにし,15 秒ごとに疑似乱数を用いて確率的に仮想マシンへのアクセスを発生させる．アクセス頻度が横軸の数値になるようにその確率を定め, メモリ使用量を 30 秒ごとに 30 分間サンプリングして平均値をとり, 更にその値の全ノードにおける平均値を算出して縦軸の数値としている．

使用メモリ量の減少量はアクセス頻度そのものよりも自動ハイバネートまでの時間とアクセス頻度の比に依存している．自動ハイバネートまでの時間を短いものに設定すればハイバネートする仮想マシンの数が増加するためによりメモリ使用量を削減できるが, こまめなハイバネートにはいくつかの問題点もあることを考慮しなければならない．まずセーブ中は仮想マシンが接続を受け付けることができないため, 頻繁なハイバネートはシステム全体の応答性を下げる結果になってしまう．そして, ハイバネート時にはスナップショットを書き込むためのネットワークを介したアクセスが行われるために, 通常に稼動している仮想マシンや物理ノードの通信性能や, 仮想マシンの再開の処理に悪影響を与える可能性がある．以上のような利点, 欠点を考慮した上でシステムのパラメータを設定する必要がある．

4. 関 連 研 究

本研究では, 仮想マシンによって無駄に使用されているリソースをハイバネートにより解放することができる．このような仮想マシンの余剰リソースに着目した研究にバーチャルクラスタにおけるリモートスワッピング[2] がある．このリモートスワッピングに関する研究では, 割り当てられたが使われていないメモリを他の仮想マシンと共有するためのフレームワークを提案している．仮想マシンのスワップデバイスを HDD から他の仮想マシンの未使用メモリに変更し, パフォーマンスを向上させることができる．このフレームワークは完全には実装されておらず, そのプロトタイプを評価し, その有用性を検証している．

また,ODRS ではハイバネートした仮想マシンを別のノードで再開することで負荷分散を実現しているため, 仮想マシンのマイグレーションを用いて負荷分散を行う研究と類似性がある．7) ではクラウド環境における負荷分散のためのマイグレーションを行うアルゴリズムを提案している．このようなマイグレーションを用いた負荷分散と違い, 本研究で提案している手法は積極的なマイグレーションを行わない．つまり, 負荷の高い物理ノードが存在したとしても, そのノード上の仮想マシンが使われていると判断されればその物理ノードに対しての負荷分散を行わないという問題点がある．

5. ま と め

本論文では, ユーザに対して透過的な, 仮想マシンの自動ハイバネーション, レジューム, 及び負荷分散を行う分散システムである ODRS を提案した．このシステムは仮想マシンのリソース使用率, 通信状況を監視して, 使われていないと判断される仮想マシンをハイバネートし, 物理ノードの使用リソースを減少させることができる．ハイバネートされた仮想マシンは, ユーザから接続がきた際には自動的に再開され接続を受け付けるため, ユーザはハイバネーション処理を意識することなくサービスを利用できる．この再開されるノードには最も負荷の低いノードが選ばれるため, 物理ノード間の負荷分散も同時に実現している．

Vol.2011-OS-118 No.3
2011/7/27

そして本論文では,ODRS をアクセス時間とメモリ使用量の面から評価した. アクセス時間は仮想マシンの再開時間, すなわちスナップショットの大きさに依存しており, スナップショットが 130MB の場合 2.86 秒,700MB の場合 8.08 秒となっている. メモリ使用量の実験結果から,ODRS はアクセスが少ない, 使われていない仮想マシンが多い環境では効果的にメモリ使用量を削減することが可能であることが示された. ハイバネートまでの時間を 2 分, アクセスの平均間隔を 4 分と設定した場合では 54%のメモリ使用量削減が見られた.

参 考 文 献

1) KVM: *Kernel Based Virtual Machine*, http://www.linux-kvm.org/page/Main_Page.
2) Okuda, T., Nagai, Y., Okamoto, Y. and Kawai, E.: A Remote Swap Management Framework in a Virtual Machine Cluster, *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, pp. 546 –547 (online), DOI:10.1109/CLOUD.2010.13 (2010).
3) Regola, N. and Ducom, J.-C.: Recommendations for Virtualization Technologies in High Performance Computing, *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, pp. 409 –416 (online), DOI:10.1109/CloudCom.2010.71 (2010).
4) Rose, M.T. and McCloghrie, K.: RFC 1155: Structure and identification of management information for TCP/IP-based internets (1990).
5) Smith, J. and Nair, R.: The architecture of virtual machines, *Computer*, Vol.38, No.5, pp.32 – 38 (online), DOI:10.1109/MC.2005.173 (2005).
6) Umeno, H., Parayno, M., Teramoto, K., Kawano, M., Inamasu, H., Enoki, S., Kiyama, M., Aoyama, T. and Fukunaga, T.: Performance Evaluation on Server Consolidation Using Virtual Machines, *SICE-ICASE, 2006. International Joint Conference*, pp.2730 –2734 (online), DOI:10.1109/SICE.2006.315198 (2006).
7) Zhao, Y. and Huang, W.: Adaptive Distributed Load Balancing Algorithm Based on Live Migration of Virtual Machines in Cloud, *INC, IMS and IDC, 2009. NCM '09. Fifth International Joint Conference on*, pp. 170 –175 (online), DOI:10.1109/NCM.2009.350 (2009).