ソケットアウトソーシングを適用した仮想計算機でのライブマイグレーションの実現

三戸健一[†] 齊藤剛[†] 新城靖[†]
佐藤聡[†] 中井央[†] 板野肯三[†]

1. はじめに

今日、仮想計算機は広く用いられている。仮想計算機によって、限られた計算機資源を効率的に使用できる。また、様々な計算機環境を簡単に用意することもできる。仮想計算機の入出力をより高速に実行するため、我々はソケットアウトソーシングと呼ばれる手法を提案している[1)]。ソケットアウトソーシングでは、ゲスト OS の入出力要求を仮想計算機モニタ (VMM: Virtual Machine Monitor) が提供する仮想デバイスよりも高水準のモジュールの段階でホスト OS へ委譲している。これにより入出力処理を高速かつ低負荷に行うことができる。しかし、VMM においてソケットアウトソーシングを利用すると、仮想計算機を稼働中に移動させる機能であるライブマイグレーションが利用できなくなるという問題が存在する。

本研究では、ソケットアウトソーシングを導入した VMM においても、従来と同じくライブマイグレーションが利用できるようにするため VMM を改良し、この問題を解決する。

2. 本研究の対象

2.1 ソケットアウトソーシング

仮想計算機においてエミュレーション方式で入出力を行う場合、ゲスト OS は VMM が提供する仮想デバイスにアクセスする。すると VMM のエミュレータはデバイスをエミュレートし、ホスト OS に入出力要求を伝える。

VMM によるデバイスのエミュレーションによって発生するオーバーヘッドを低減するため、ゲスト OS のデバイスドライバという低水準のモジュールを置き換える準仮想化が広く用いられている。これに対して、我々はゲスト OS の高水準モジュールを置き換えるアウトソーシングという手法を提案している[1)]。たとえば、ソケットやファイルシステムへの操作をホスト OS のモジュールへ委譲し、仮想ハードウェアデバイスを迂回することでオーバーヘッドの大幅な低減を実現している。ソケットをアウトソーシングの対象としたソケットアウトソーシングでは、完全仮想化に比べて通信速度の大幅な向上と低遅延化が実現されている。準仮想化と比較しても、アウトソーシングにより割り込み処理のオーバーヘッドと遅延が小さくなるという利点がある。

2.2 ライブマイグレーション

仮想計算機の有用な機能としてライブマイグレーションが存在する[2)]。ライブマイグレーションとは、ある VMM 上で動作している仮想計算機の状態を別の VMM へ転送することにより、稼働中の仮想計算機を異なる VMM へ移動させる機能である。

ライブマイグレーションでは、仮想計算機を停止させることなく、物理的には異なるハードウェアへ移動させる。このため、負荷に応じて仮想計算機を適当なハードウェアへ移動させることができる。また、仮想計算機を待機系へ移動させることによって、ゲスト OS を停止させずにホスト OS の更新やハードウェアの交換を行うことも可能となる。

ライブマイグレーションでは、仮想計算機の CPU の状態とメモリの内容を転送する必要がある。また、ネットワーク通信やディスクへの I/O を行っている場合、移動元のホスト OS で TAP デバイスやディスクイメージへのファイルディスクリプタを削除し、移動先のホスト OS で新しく作成する必要がある。その場合、CPU やメモリとは異なり、ネットワークデバイスやディスクデバイスの状態を移動先のホスト OS へ転送する必要はない。

3. ソケットアウトソーシングとライブマイグレーション

3.1 ソケットのマイグレーション

現在、ソケットアウトソーシング機能を持つ VMM では、ライブマイグレーション機能が利用できない。ソケットアウトソーシングを行う場合、VMM はホスト OS にソケットの処理を委譲しているため、ホスト OS のカーネル内にソケットを実装するため次のような状態 (ソケットの状態) が存在する。

- 接続元と接続先のアドレスとポート番号
- TCP の再送用キュー (TCP の受信キューは通信相手の再送により回復可能)
- シーケンス番号

† 筑波大学

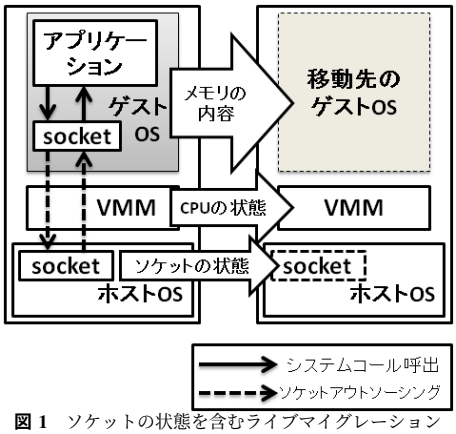

図 1 ソケットの状態を含むライブマイグレーション

- ソケットオプション
- TCP オプション

ライブマイグレーションのために移動元のホスト OS において通常の手順でソケットを閉じると、通信相手に通信の終了を知らせるパケットが送信される。また、ソケットの状態も失われてしまう。従って、ライブマイグレーション後に新しくソケットを作成して通信を再開することはできない。

3.2 ソケットの情報を保存し復元する手順

3.1 節で述べたように、マイグレーション先でソケットを新しく作成しても通信を再開することは出来ない。そこで本研究では、図 1 で示されるようにソケットの状態をホスト OS から回収し、移動先の VMM に転送して、転送前のソケットの状態を復元する。この転送と復元には、Linux Kernel 3.5 で新しく導入された TCP repair mode[3] を、以下の様に用いる。

(1) 通信相手から、移動元のホスト OS への通信をパケットフィルタ (iptables) でブロックする。
(2) TCP ソケットを repair mode に移行する。
(3) 移動元の VMM を用いてホスト OS 上の TCP ソケットの状態を読み取る。
(4) FIN パケットを送信せずソケットを close する。
(5) 3 で読み取った TCP ソケットの状態を、移動先の VMM へコピーする。
(6) コピーされた状態を元にして、移動先の VMM を用いてホスト OS 上にソケットを復元する。
(7) 新しく作ったソケットにおいて、TCP repair mode を終了する。

4. 関 連 研 究

仮想計算機ではなくプロセスのマイグレーションを行う場合、通常、TCP コネクションを維持したまま移動することはできない。TCP コネクションを維持したまま移動することを可能とするために、ローダブル・カーネル・モジュール (LKM) を用いて Linux カーネルのシステムコールを置き換える手法がある[4]。この研究では、ライブマイグレーションを行う対象となる 2 つのノードの OS だけでなく、通信相手となるノードの OS にも LKM をインストールしシステムコールを置き換えておく必要がある。

また、同じく LKM を用いて TCP コネクションを維持したままプロセスマイグレーションを可能とする手法が提案されている[5]。この研究ではクラスタの負荷均衡を目的として、全てのノードにパケットをブロードキャストすることでマイグレーションを可能としている。

これに対して、本研究では Linux Kernel 3.5 の機能を利用して、ホスト OS のユーザ空間で動作する VMM を修正することでソケットを開いたままマイグレーションを行う。文献 4) の方法と異なり、通信相手の OS には手を加える必要はない。また、文献 5) の方法とも異なり、ブロードキャストは不要である。

5. お わ り に

現在までに、Linux Kernel 3.5 において TCP repair mode を用いることで、仕様通り正常に TCP ソケットの保存と復元ができることを確認している。今後、Linux Kernel 2.6 上の Kernel-based Virtual Machine(KVM) と QEMU において動作しているソケットアウトソーシング機能を Linux Kernel 3.5 上の KVM と QEMU へ移植し、ソケットの保存と併せてライブマイグレーションができるよう VMM を修正する。

参 考 文 献

1) Eiraku, Hideki, Shinjo, Yasushi, Pu, Calton, Koh, Younggyun and Kato, Kazuhiko: “Fast Networking with Socket-Outsourcing in Hosted Virtual Machine Environments”, Proceedings of the 2009 ACM symposium on Applied Computing, pp.310–317, 2009.
2) C. Clark, K. Fraser, S. Hand, J.G. Hansen, E. Jul, C. Limpach, I. Pratt and A. Warfield: “Live Migration of Virtual Machines”, the 2nd conference on Symposium on Networked Systems Design & Implementation, pp.273–286, USENIX, 2005.
3) Pavel Emelyanov: “TCP connection repair”, `http://lwn.net/Articles/484472/`, February 2012, accessed: 2012/09/24.
4) H. Zhong and J. Nieh: “CRAK: Linux Checkpoint/Restart As a Kernel Module”, Columbia University Dept of CS, Technical Report CUCS-014-01, 2001.
5) G. Balazs, F. Hajime and I. Yutaka: “Live Migration of Processes Maintaining Multiple Network Connections”, IPSJ Transactions on Advanced Computing Systems (ACS), Vol.3, No.1, pp.1–12, March 2010.