6A-2

VM 間同期における性能評価と I/O 性能の解析

佐藤 隆佑† 伊藤 孝之† 鶴 薫†
†三菱電機株式会社 情報技術総合研究所

1 はじめに

社会インフラシステムなどのミッションクリティカルなシステムでは、高い信頼性・可用性が求められる。このようなシステムでは、ハードウェアに障害が発生した場合において業務を継続するため、フォールトトレラントサーバ (以下 FT サーバ) による多重化が利用される。しかし、専用の計算機を必要とするため、導入コストが高くなるという問題がある。

このような問題を解決する方法の 1 つとして、ソフトウェアにより FT を実現する方法がある。仮想化技術を組み合わせ、異なる計算機上で動作する仮想計算機 (VM) を同期し続けることで、FT サーバと同等の機能を実現することができる [1]。しかし、VM 間で同期を行うと、同期処理によるオーバヘッドが発生することから、大幅な性能低下が懸念されている。

本稿では、VM 間同期について、評価と解析を行った結果について報告する。VM 間同期処理を評価・解析した結果、大幅な性能低下を確認した。また、更新メモリの検出処理が性能低下の原因であることが判明した。この課題を解決するメモリ管理方法について提案し、実装と性能評価を行った結果について報告する。

2 VM 間同期における性能課題

Linux の仮想化技術である KVM において、2 つの VM のコンテキストの同期を実現する技術として kemari がある。kemari は、I/O を契機として更新メモリを運転系から待機系に転送する。これにより、各系で動作する VM の I/O コンテキストの整合性を保つことにより、ソフトウェア FT を実現する。この概要を図 1 に示す。

kemari には、同期処理によるオーバヘッドにより、性能が低下するという課題がある。kemari による同期時および非同期時における netperf のリクエストレスポンスによる性能評価結果を表 1 に示す (詳細は後述の表 2 評価環境を参照)。この結果から、メモリ 1GB の VM は同期により性能が約 1/50 に低下し、メモリ量が多いほど性能低下幅が大きいことがわかる。kemari 同期による性能低下の原因として、同期処理のための動作停止が挙げられる。VM が I/O を発行するたびに同期処理を行い、同期処理では VM の動作を一時停止させる。このため、I/O が頻発する状況では VM 動作の停止と再開を繰り返す。したがって、VM が実際に動作する時間が削減されるため、性能が大幅に低下してしまう。

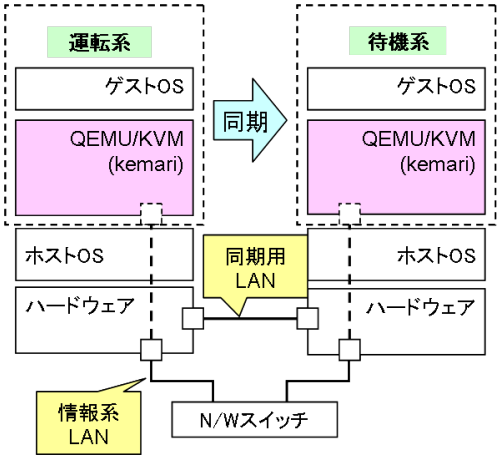

図 1: kemari による VM 間同期の概要

表 1: メモリ量とリクエストレスポンス (単位：TPS)

割当てメモリ量	非同期時	同期時
512MB	7075.42	184.66
1024MB	7163.67	164.00
4096MB	7213.12	80.72

同期処理のオーバヘッドを削減する取り組みとして、転送データ量の削減がある [2]。この取り組みは同期通信について、転送遅延や帯域制限が発生する環境では性能改善の効果が期待できる。しかし、表 1 の通り、同期通信が十分に高品質な環境でも性能低下が発生していることから、同期処理のオーバヘッドを削減するためにはこれらの取り組みでは十分ではない。

kemari による VM 間同期 (以下 kemari 同期) 処理について解析を行った結果、転送する更新メモリの検出処理において、多くの時間が費やされていることが判明した。kemari は前回同期時以降に更新されたメモリページのみを転送するが、実際に更新されたページ量にかかわらず、メモリ全領域について更新判定を行うことで、転送するメモリページを決定している。このため、メモリ量が増加するほど kemari 同期時の性能低下が大きくなると考えられる。

kemari を利用する上で、VM の割当てメモリ量に制

I/O Performance Evaluation and Analysis in VM Synchronization
†Ryusuke SATO, Takayuki ITO, Kaoru TSURU
†Information Technology R&D Center, Mitsubishi Electric Corporation

限を与えることは望ましくない。VM への割当てメモリ量に関する性能低下を解決し、任意のメモリ量を割当てた場合でも kemari 同期によるオーバヘッドが生じない仕組みの構築が必要である。

3 同期処理 (更新メモリ管理) の変更

3.1 従来の更新メモリ管理方法

kemari の更新メモリ管理方法として、ダーティフラグの利用がある。各メモリページごとにダーティフラグがあり、フラグのセットとクリアを利用して更新を管理する。ページが更新されると対応するフラグをセットし、ページが転送されるとフラグをクリアする。全ページのフラグがクリアされることで、VM の同期が完了となる。

この方法では、実際にダーティフラグがセットされているメモリページ量にかかわらずメモリ全体をチェックする必要がある。このため、特に kemari のように更新メモリの転送を繰り返す場合には、更新メモリの検出処理に多くの時間が費やされてしまう。

3.2 キューによる更新メモリ管理

割当てメモリ量に依存しない方法として、ダーティフラグと共に、キューを用いて更新メモリを管理方法を提案する。メモリの更新を行う場合には、フラグをセットすると同時に更新メモリアドレスを enqueue する。逆にメモリの転送を行う場合には、dequeue して転送メモリページを決定すると同時に、該当するメモリのフラグをクリアする。

この方法は、キューの操作が定数時間で実行可能なため、転送メモリの決定も定数時間で完了する。また、フラグを併用することで、同一のメモリページを enqueue してしまい、転送容量が増加することはない。

4 評価と考察

VM に対する割当てメモリ量を 512MB、1G、4G と変えたときの、kemari 同期時における性能評価を行う。提案するキューを用いた更新メモリ管理手法の効果を確認するため、netperf によるリクエストレスポンスの計測により性能評価を行う。評価環境は表 2 の通り。

実験結果を図 4 に示す。この結果から、従来ではメモリ量の増加とともに性能が劣化していたのに対し、提案手法では性能が向上していることがわかる。転送メモリを決定するために、従来では全メモリページの更新フラグをチェックしていたのに対し、提案手法では dequeue のみでよい。このため、探索処理が削減され、kemari 同期のオーバヘッドが削減できたと考えられる。

表 2: 評価環境

サーバ	項目	詳細
物理サーバ (運転系/待機系)	機種	HP DL380G7
	CPU	Intel Xeon X565 2.67GHz x2(HT 無効)
	Memory	12GB
	Disk	72GB x3
	NIC	Broadcom BCM5709 1Gb Ether(通信用)
		NetXen NX3031 10Gb Ether(同期用)
	OS	CentOS6.2
仮想マシン	CPU	仮想 CPU x1
	Memory	512MB / 1GB / 4GB
	Disk	10GB
	OS	CentOS6.2
	QEMU	ver 1.0
	kemari	next(d162772)

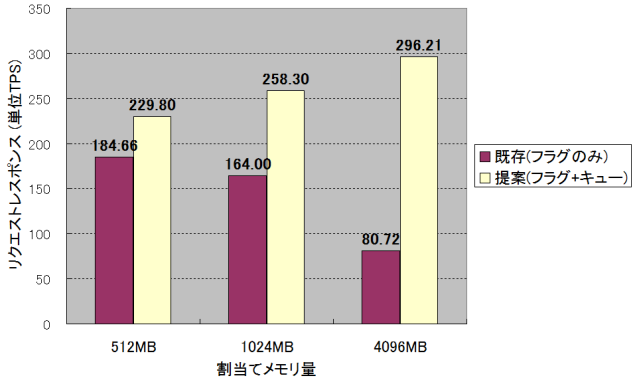

図 2: VM 搭載メモリ量と応答性能

5 おわりに

KVM において VM 間のコンテキストの同期を実現する kemari の評価と解析を行った。提案するキューを用いた更新メモリの管理方法を用いることで、転送メモリの決定を定数時間で実現し、性能改善を行った。

本手法は kemari だけでなく、通常のライブマイグレーションにも適用可能である。本手法によりライブマイグレーション完了直前の停止時間の短縮が期待できる。

kemari 同期時の性能向上は実現したが、非同期時と比べると依然として性能低下幅は大きい。kemari を実適用する行う上では、更なる性能改善が必要である。

参考文献

[1] 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.

[2] 大村 圭. 田村 芳明, 湯口 徹, 盛合 敏. 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. 情報処理学会研究報告. , Vol. 2011-OS-116, No. 16, pp. 1–8, 2011.