

# 複数 VM 環境における KSM を用いた KVS 性能の向上に関する一考察

## A Study on KVS Performance Improvement using KSM in Multi VM Environment

徳田 大輝<sup>†</sup>      御代川 翔平<sup>†</sup>      山口 実靖<sup>†</sup>  
Tokuda Taiki      Shohei Miyokawa      Saneyasu Yamaguchi

### 1. はじめに

クラウドコンピューティングの普及によりスケーラビリティの高い分散データベースが重要視されており、その一つに KVS (Key-Value Store) がある。KVS は、データ構造の簡素化と一貫性保証の度を下げることでスケーラビリティの向上を図っており、サーバ増設による性能向上と耐障害性向上を実現している[1]。KVS は仮想計算機 (VM) を用いたクラウドコンピューティング環境で実行されることも多く、仮想化環境における性能が重要であると考えられる。

本研究では、KVS の一つである Cassandra と、仮想化システムの一つである KVM に着目し、メモリ共有による複数 VM 環境における KVS の性能向上手法について考察する。

### 2. KVS

KVS は、Key と Value の組を書き込み、Key を指定することで Value を得ることができるデータベース管理ソフトウェアである。代表的な KVS の実装に Cassandra [2] がある。

Cassandra はオープンソースの KVS であり、BigTable [3] のデータモデルと Dynamo [4] の分散ハッシュテーブルを併せ持った分散データベース管理システムである。Cassandra はハッシュ法を使用して各ノードにトークンと呼ばれる値を割り当て、Key と Value の組を各ノードに割り当てる。このとき、トークン値の担当範囲をノード別に指定することで、各ノードに公平にデータ量を分散することができる。また、耐障害性の高さ、ノードの非集中性、高可用性、動的に伸縮可能なスケーラビリティ、設定可能な一貫性などの機能を持っている。

### 3. 仮想計算機

#### 3.1. KVM (Kernel-based Virtual Machine)

一般にクラウド環境などは仮想計算機を用いて構築される。KVM は代表的な仮想化システムの一つであり、本研究では KVM を用いて調査を行う。KVM は Linux カーネル内に実装されており、OS をハイパバイザとして稼働する。

KVM の仮想 HDD の構築方法には、イメージファイルを使用するモードと、パーティションを使用するモードがある。本研究では、仮想 HDD はイメージファイルモードを使用し、同モードではゲスト OS 上のアプリケーションはゲスト OS ファイルシステム、仮想計算機、ホスト OS ファイルシステムを介して HDD へのアクセスが行われる。

#### 3.2. KSM (Kernel Same-page Merging)

KSM は VM 群が利用しているページの内、同一の内容のメモリページ群を 1 つの物理ページにまとめる機能で、実際に搭載しているメモリ量よりも見かけ上多くのメモリ

を仮想マシンに割り当てることができる。また、KSM がスキャンを積極的に行うか、消極的に行うか、停止するかなど状況に応じて調整するための ksmtuned というデーモンが用意されている。

### 4. 仮想化環境における Cassandra の性能評価

本章にて、仮想化環境における Cassandra 性能の評価を行う。

#### 4.1. 測定方法

ゲスト OS (仮想計算機) 上で Cassandra システムを稼働させ、YCSB (Yahoo! Cloud Serving Benchmark) [5] で得られる性能を評価した。

Cassandra はレプリカ数を 1 とし、データベースサイズは 17[GB]、トークン範囲は均等に割り当てた (例: ノード数 6 の場合、各ノードのトークン範囲は全データの 1/6)。2 台の物理計算機を使用し、各ホスト OS 上で 3 台の VM を稼働させ、計 6 台の VM を使用した。また、YCSB をクライアント PC 上で実行し、ゲスト OS で稼働する Cassandra に対して負荷をかけた。YCSB のスレッド数は 18、読込負荷 100% とした。測定環境を図 1 に示し、使用したホスト OS とゲスト OS の仕様は表 1 の通りである。

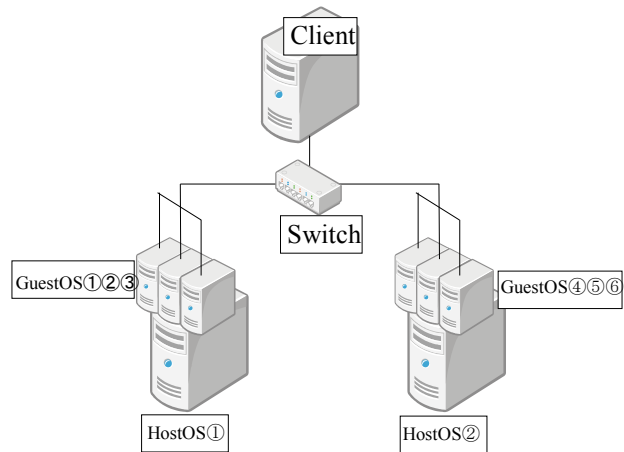


図 1 測定環境

表 1. 使用計算機の仕様

	Host OS	Guest OS
OS	CentOS6.5	
Kernel	Linux 2.6.32.27	Linux 2.6.32-431.el6.x86_64
Memory	8[GB]	2[GB]
CPU	AMD Turion II Neo N54L	

#### 4.2. 測定結果

初期状態 (default)、KSM を適用した場合の 2 種類の状態の性能の比較を行った。測定結果を図 2、図 3 の“default”と“KSM”に示す。

図 2 より、KSM を適用した場合、初期設定よりもスル

ープロットが25%高くなることが確認された。これは、すべてのゲスト OS 上で同じ OS, 同じカーネルが起動しているため、KSM によるページマージングが効果的に働き、物理ページの重複が減ることでホスト OS のメモリキャッシュが増えたことが原因だと考えられる。KVS のデータ (Key 値と Value 値) は乱数であり、これらの共有は行われていないと考えられる。

図3は測定中の平均空きメモリ量を示している。図3より、KSM を使用することでホスト OS の空きメモリ量が増加することを確認できる。

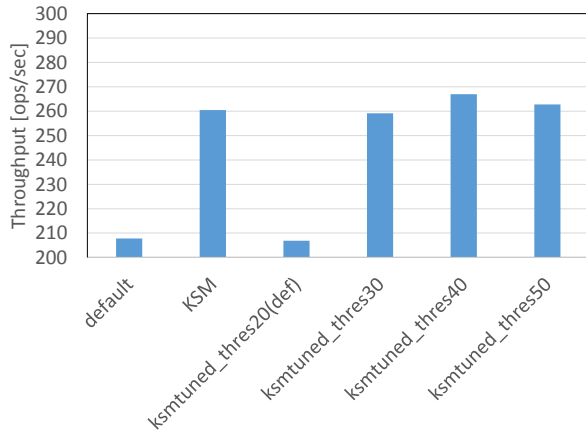


図2 YCSBのスループット

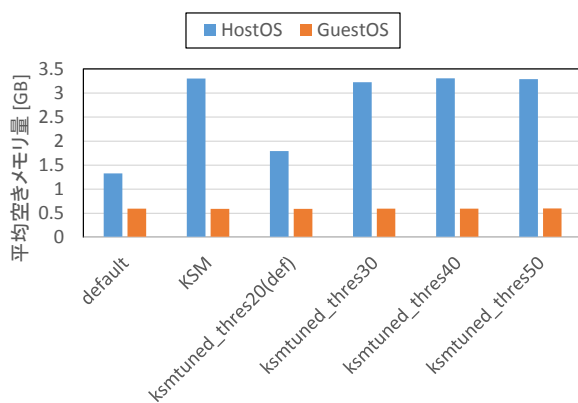


図3 平均空きメモリ量の比較



図4 pages\_shared および pages\_sharing の比較

次に、無変更で ksmtuned を適用したときの性能と空きメモリ量を図2, 図3の“ksmtunde\_thres20(def)”に示す。図2より、無変更の ksmtuned を適用すると通常の KSM(ksmtuned なし)適用時よりも低い性能となることが分かる。これは ksmtuned の閾値が適切でなく、KSM が適切に起動、制御されていないことが原因であると考えられる。

そこで、ksmtuned の起動の閾値(thres)を上げ、KSM の起動条件を緩くした。初期設定では thres は 20 であり、これを 30, 40, 50 と変更し測定した。本稿ではこれらを thres20(def), thres30, thres40, thres50 と呼ぶ。これらの性能と空きメモリ量を図2, 図3に示す。図2より、KSM 起動閾値を修正にすることにより小規模ながら通常の KSM(ksmtuned なし)よりも性能が向上することが確認された。図4にそれぞれの設定における pages\_shared(共有されている物理ページ数)と pages\_sharing(共有により節約されたページの数)を示す。図より、thres の値を制御することにより KSM が適切に起動され共有が行われていることが分かる。また、ksmtuned により通常の KSM よりもスキャン間隔の延長が実現されており、KSM 処理の負荷の低減とアプリケーション性能の向上が達成されたと考えられる。

## 5. まとめ

本稿では、仮想化環境における KVS の性能に着目し、性能評価を行った。そして、複数 VM 上で同一の OS が実行されている点に着目し、KSM の適用について考察を行った。評価の結果、KSM を用いて同一ページのマージを行うことによりホスト OS の空きメモリ量が増加し、KVM の性能が向上することが確認された。

今後は、KSM の改良による Cassandra 性能のさらなる向上についての考察を行う予定である。

謝辞

本研究は JSPS 科研費 25280022, 26730040, 15H02696 の助成を受けたものである。

## 参考文献

- [1] 堀内 浩基, 山口 実靖, “KVS における動的性能拡張性の向上”, 研究報告マルチメディア通信と分散処理(DPS-154, 03, 2013)
- [2] Avinash Lakshman and Prashant Malik, “Cassandra- A Decentralized Structured Storage System”, LADIS 09, 2009
- [3] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall and Werner Vogels, “Dynamo: Amazon’s Highly Available Key-value Store”, SOSP ’07, 2007
- [4] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes and Robert E. Gruber, “Bigtable: A Distributed Storage System for Structured Data”, IOSDI ’06 pages 205--218, 2006
- [5] Brian F. Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, Russell Sears “Benchmarking Cloud Serving Systems with YCSB”