C-011

読込み書込み比率を考慮した 動的 VM メモリ割り当てによる DBMS 性能向上に関する一考察

A Study on DBMS Performance with Dynamic VM Memory Allocation

坂本　雅哉†　　山口　実靖†
Masaki　Sakamoto　　Saneyasu　Yamaguchi

1. はじめに

データセンタ等における，サーバの消費電力の増加や設置スペースの肥大化が問題となっている．この問題に対する解決策の一つとして，仮想化技術を用いて複数のサーバ OS を一台の物理マシンに集約する手法がある[1].

仮想化環境では，仮想マシン(VM)を停止させることなく割り当てメモリ量を変更することが可能である．一つの物理マシン上に複数の VM が稼働しており，それぞれの VM の負荷が時間により変化する場合は，静的なメモリ割り当てを行うとメモリを効果的に活用できないことになる．負荷変動に対応するためには，動的に VM メモリ割り当て量を変更する必要がある．

仮想化ソフトウェアの Xen には，動的に VM メモリ割り当て量を変更させる機能として xenballoon がある．しかし，xenballoon がメモリ割り当てを行うために考慮するパラメータは，VM 内のプロセスが使用するメモリ量のみであり，ページキャッシュとして利用されるメモリの量を考慮していない．よって xenballoon では適切に I/O 性能向上を行えないと考えられる．

本研究では，Xen を用いた仮想化環境における，メモリ割り当て量，キャッシュヒット率，DBMS 性能の関係性について調査を行い，キャッシュヒット率を考慮した動的 VM メモリ割り当て量の最適化の手法の提案を行う．

2. xenballoon

xenballoon は，Xen が持つ機能の一つで，VM に割り当てるメモリ量を動的に変更する機能である．VM 上でデーモンとして動作し，ゲスト OS 内のプロセスの推定メモリ使用量(Commited_AS)を監視してホスト OS への要求メモリ量を調整する[2].

3. キャッシュヒット率と DBMS 性能の関係

本章で，キャッシュヒット率と DBMS 性能の関係を示す．

3.1 キャッシュヒット率の解析手法

図 1 のようにホスト OS，ゲスト OS のカーネル内でキャッシュ前後の I/O 量を監視し，キャッシュヒット率を求めた．

キャッシュ前の I/O 量であるページキャッシュへのアクセス量を測定するためにカーネルの改変を行った．ゲスト OS では，カーネル構成ファイルの `mm/filemap.c` 内にある「`find_get_page`」関数の実行回数とブロックサイズの監視，ホスト OS では，`drivers/xen/blkback/blkback.c` 内にある「`dispatch_rw_block_io`」関数の実行回数とブロックサイズの監視をするための改変を行った．

キャッシュ後の I/O 量を測定するために VM の仮想ブロックデバイス(XVD)における I/O 量，物理 HDD における I/O 量を測定する様にカーネルの改変を行った．ゲスト OS では，「`do_blkif_request`」関数の実行回数とブロックサイズの監視，ホスト OS では，`drivers/scsi/sd.c` 内にある「`sd_prep_fn`」関数の実行回数とブロックサイズの監視をするための改変を行った．

キャッシュ後の I/O 量がキャッシュミスした量となり，測定したキャッシュ前後の I/O 量の比よりキャッシュヒット率を求める．

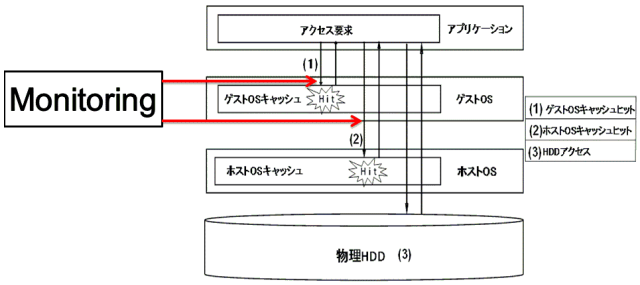

図 1　キャッシュヒット率の測定

3.2 キャッシュヒット率と DBMS 性能の関係

キャッシュヒット率と DBMS 性能の関係を調査するため，VM 上で TPC-H を実行し，クエリの終了までの時間とメモリ量とキャッシュヒット率を測定した．本論分では TPC-H の SF(scale factor)は 1 ,2 ,3 とし，クエリ 4 を用いて評価を行った．TPC-H には 22 種類のクエリが用意されているが，本稿ではメモリ量の影響が大きく出ると予想される I/O 使用率が最も高いクエリ 4 を選択した．I/O 使用率は上記環境における事前実験により調査した．実験に用いた物理マシンとホスト OS の仕様は CPU Intel Celeron G530 2.40GHz, OS CentOS 6.3, カーネル Linux 2.6.32.57, 仮想化システム Xen 4.1.2, メモリ 8GB, HDD 1TB である．仮想マシンとゲスト OS の仕様はカーネル Linux 2.6.18.8, メモリサイズは動的に変更し, HDD 100[MB]である．

測定結果を図 2 に示す．SF=1, 2, 3 においてはおおむね 1GB, 2GB, 3GB のデータが作成，使用される．図より，VM のメモリ割り当て量がデータサイズよりも少ないとキャッシュヒット率が低く，クエリ終了までの時間も長いことが分かる．一方 VM メモリ割り当て量がデータサイズよりも多いと，キャッシュヒット率は 100%となり，クエリ終了までの時間が著しく短くなっていることがわかる．以上より，メモリ増加量と，処理時間短縮の関係に着目すると，低ヒット率時はメモリ量を増加させても時間短縮の効果が小さく，100%を除く高ヒット率時は時間短縮の効果が大きいことがわかる．

†工学院大学大学院工学研究科電気電子工学専攻
Electrical Engineering and Electronics, Kogakuin University Graduate School

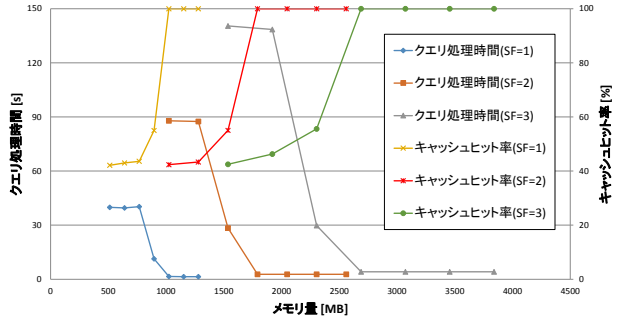

図 2　クエリの終了までの時間とメモリ量とキャッシュヒット率の推移

4. 提案手法

本章で，ゲスト OS のページキャッシュヒット率を監視し，それを元に VM にメモリ割り当てを行う手法を提案する．

本手法では，各 VM のキャッシュヒット率の値をホスト OS にて集計し，以下の手順により VM メモリ割当量を決定する．

1. 全 VM のメモリ割当量を 10%減少させ，これも再配分用メモリとする．
2. 上記 1 にて得た再配分用メモリをキャッシュヒット率が 100%未満の VM に各 VM のキャッシュヒット率により比例配分する．

提案手法は，ゲスト OS 上で動作する xenballoon に変更を加えることにより実装できる．キャッシュヒット率の取得は 3.1 節の手法により実現する．キャッシュヒット率の測定機能，メモリ量とキャッシュヒット率の情報をホスト OS に伝える機能，メモリ割り当て量の情報をホスト OS から取得する機能を追加で実装する．

上記の動作は，キャッシュヒット率が 100%未満の場合 5 秒ごとに行われ，キャッシュヒット率が 100%の場合は 30 秒ごとに行われる．キャッシュヒット率が 100%を下回るとクエリ終了までの時間が，著しく長くなることが 4 章の実験から確認されているため，情報取得と割り当てメモリ量の変更のインターバルを調整することによりキャッシュヒット率が 100%に近い状態を維持する時間が長くしている．

またホスト OS 上では，各 VM のメモリ割り当て量とキャッシュヒット率を取得，提案手法に従った各 VM のメモリ割り当て量の決定，メモリ割り当て量の変更が行われる．これらはインターバル 3 秒毎に実行される．

5. 性能評価

提案手法の有効性を検証するために性能評価実験を行った．本章にて，性能評価結果について述べる．

5.1 評価方法

実験では，Xen を用いて 1 台の物理マシン上に 3 台の仮想マシンを立ち上げ，全 VM 上で TPC-H を並列実行し I/O 性能を測定した．各 VM でそれぞれの SF にてクエリを 10 回実行し，すべてのクエリが終了するまでの時間，各 VM のメモリ割り当て量とキャッシュヒット率の推移を観察した．SF の変更順はランダムとした．TPC-H の設定，実験に用いた物理マシンと仮想マシンの仕様は 3 章と同様である．

5.2 I/O 性能の測定

図 3 に提案手法と静的メモリ割り当て手法の性能を示す．静的メモリ割り当て手法とは全 VM のメモリを 1365 [MB](均等割り当て)で固定したものである．図より静的メモリ割り当て手法より提案手法の方がクエリ終了までの時間が短いことがわかる．

図 4 に VM のメモリ割り当て量の推移を示す．図より，SF 値が小さくすべてのデータをメモリ内に格納可能である状況(SF=1 など)に対しては積極的にメモリを与えており，メモリ割り当てが性能向上に寄与していることが分かる．

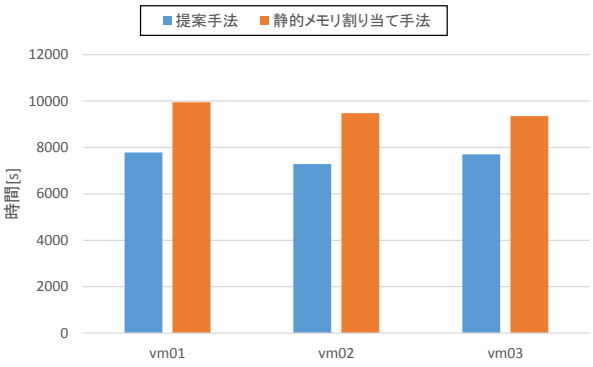

図 3　性能評価

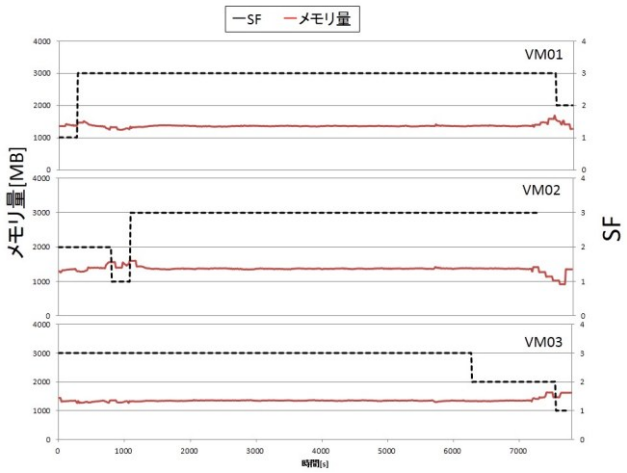

図 4　ベンチマークデータサイズと VM のメモリサイズの推移

6 おわりに

本稿では，キャッシュヒット率を考慮した VM メモリ割り当て手法を提案し，TPC-H による評価を行った．

今後は，TPC-H の他のクエリを用いての評価を行う予定である．

謝辞　本研究は JSPS 科研費 24300034，25280022，26730040 の助成を受けたものである．

参考文献

[1] 坂本 雅哉,山口 実靖, "仮想化環境におけるキャッシュヒット率を考慮した VM メモリ割り当て", 第 12 回情報科学技術フォーラム FIT 2013 RC-009,2013

[2] Stephen Spector, “Memory Overcommit”, August 27, 2008, http://blog.xen.org/index.php/2008/08/27/xen-33-feature-memory-overcommit/