ホストマシンのNUMAトポロジーに基づくvCPUスケジューリング

林　遼[†]　味曽野　雅史[†]　品川　高廣[†]

1. はじめに

マルチコアの普及により，NUMA (Non-Uniform Memory Access) アーキテクチャのマシンはデータセンター等でも広く使われるようになった．NUMA は，ローカルノード内のメモリアクセスは高速，リモートノードへのメモリアクセスは低速という非対称的構造になっており，メモリアクセスの局所性を利用して，頻繁に参照するデータをローカルノードに置くことで，全体としてメモリアクセスの高速化を実現している．従って，メモリアクセスが頻繁なアプリケーションを効率よく実行するためには，プロセスのスケジューラが NUMA ノードを考慮する必要がある．

しかし，現在の仮想化環境では NUMA アーキテクチャを十分に活用することは難しい．まず，通常の仮想化環境ではホストマシンの NUMA トポロジーは VM (仮想マシン) からは隠蔽されており，ゲスト OS が NUMA ノードに基づくスケジューリングをおこなうことは難しい．また，VMM (仮想マシンモニタ) が NUMA ノードに基づくスケジューリングをおこなう手法[1] も提案されているが，ゲスト OS の内部状態を用いずに効率の良いスケジューリングをおこなうことは難しい．pCPU（物理 CPU）を vCPU（仮想 CPU）にピンニングすることで，VM が NUMA ノードをまたがないように設定する手法もあるが，VM に割り当てられる pCPU の数が固定化されてしまう．

本研究では，ゲスト OS が NUMA に基づく効率の良いスケジューリングをおこなうために，ホストマシンの NUMA トポロジーを再現した VM を作成する．一方，VM の負荷変動時に利用する pCPU 数を柔軟に増減するために，ゲスト OS 内のエージェントで vCPU を専有・解放して，ゲスト OS がプロセスをスケジューリング可能な vCPU の数を実質的に増減する．解放した vCPU に対応する pCPU を別の VM に割り当てることで，VM 間での負荷分散を実現する．

本稿では初期評価として，ホストの NUMA ノードを意識した vCPU 割り当てをするかどうかで，メモリアクセス中心のアプリケーションの実行時間に差が出るのかどうかを評価した．

† 東京大学
The University of Tokyo

2. 提案手法

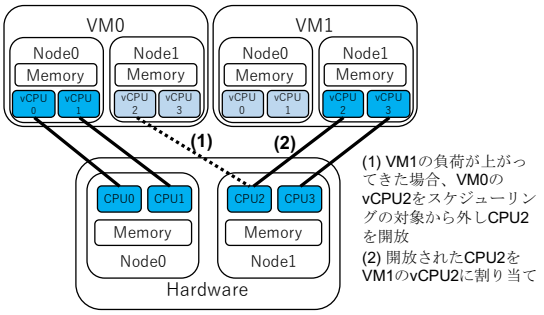

図 1　NUMA トポロジーを意識した動的な物理コアの受け渡し

本研究では，NUMA アーキテクチャをもつホストマシン上に複数の VM が存在する環境を想定する．まずそれぞれの VM にホストマシンの NUMA トポロジーを再現して，vCPU と pCPU の 1 対 1 のピンニングをおこなう．vCPU の総和が pCPU よりも多い場合は pCPU と vCPU は一対多の対応関係となってしまうが，VM の負荷に応じてゲスト OS で vCPU を動的にスケジューリング対象から外して，対応する pCPU を VM 間で受け渡しすることで，実質的に pCPU に対応する vCPU の数は常に高々1 個であるようにする．受け渡しの際は，切り離すコアと受け取るコアの NUMA ノードを意識して，切り離す際はリモートノードのコアを，受け取る際はローカルノードのコアを優先して受け渡しする．これによって，メモリアクセスを全体で最適化して，メモリアクセス中心のアプリケーションの実行時間短縮を実現する．

図 1 の例では，VM0 と VM1 のそれぞれにホストマシンの NUMA トポロジーが再現されており，初期状態では VM0 には pCPU0,1,2，VM1 には pCPU3 が実質的に割り当てられている (図 1(1))．このとき，VM0 のゲスト OS には NUMA トポロジーが見えているため，NUMA ノードを跨いでいても効率の良いスケジューリングを行うことができる．ここで，VM1 の負荷が上昇したため vCPU の割当数を増やすことを考える．まず，VM0 でリモートノードである pCPU2 を選択し，対応する vCPU2 をスケジューリングの対

象から外すことで pCPU2 を実質的に解放する．次に，VM1 で vCPU2 をスケジューリング対象に加えることで，pCPU2 を専有して利用可能にする (図 1(2)))．一連の処理を動的に行い，不必要なメモリアクセス遅延要因を生まない vCPU の負荷分散を実現する．

3. 予備評価実験

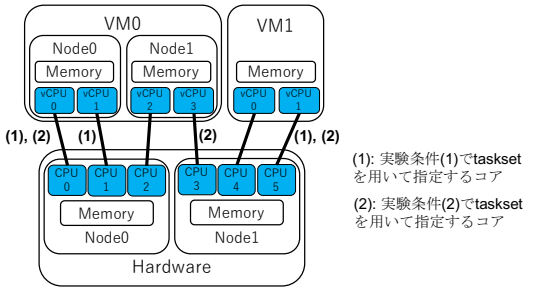

図 2 実験条件 (1)，(2)

VM 上でホストマシンの NUMA トポロジーを再現することで，ゲスト OS 上のアプリケーションの実行時間に差が出るのかを確かめるために，まずは静的ピンニングによる予備評価実験をおこなった．この実験では，VM に 1 つの NUMA ノード内の pCPU を割り当てた場合と，複数の NUMA ノードの pCPU を割り当てた場合で実行時間の差を測定した．実験に用いたホストマシンの CPU は Intel Xeon X5690 × 2，各 CPU は 6 コア 12 スレッドで計 24 個の pCPU，2 ソケットのそれぞれが NUMA ノードである．

実験では，ホストマシン上で CentOS7 を動かし，KVM を用いて 2 つの VM を作成した．VM0 には Node0 の 12 個+Node1 の 4 個の pCPU をピンニングした vCPU16 個，VM1 には Node1 の 8 個の pCPU をピンニングした vCPU8 個を割り当てた．それぞれの VM のメモリは 4GiB，ストレージは 20GB である．

実験では，VM1 でメモリ負荷をかけながら，VM0 でベンチマークを実行した．メモリ負荷をかけるのに用いたのは stress-ng コマンドで，ベンチマークは PARSEC[3] の canneal である．VM1 は vCPU6 個を taskset で指定して stress-ng を実行し常に Node1 のメモリに負荷をかける状態である．この状態で，VM0 において以下のそれぞれの条件を満たす vCPU6 個を taskset で指定して実験をおこなった（**図 2**）．

(1) Node0 にピンニングされた vCPU のみ
(2) Node0 にピンニングされた vCPU と Node1 にピンニングされた vCPU それぞれ 3 個ずつ

図 3 に実験結果を示す．NUMA-aware は (1) の条件，NUMA-unaware は (2) の条件での結果を示し，同じベンチマークを 5 回実行して実行時間をそれぞれ計測した．実験結果から，NUMA を意識した vCPU

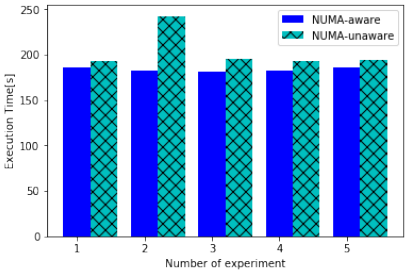

図 3 canneal 実行結果

割り当てをした場合の方が実行時間が安定しており，平均 8.9%，最大 24.6%削減できていることがわかる．

4. 関 連 研 究

vProbe[1] では，VMM で一定の周期ごとに各 vCPU で実行されているアプリケーションのメモリアクセス量を性能カウンタで取得して，各 NUMA ノードに再分配している．提案手法では，VM に NUMA ノードを再現することで，ゲスト OS による NUMA ノードに基づくスケジューリングを可能にしている．

vScale[2] では，マイクロ秒単位で vCPU の数を増減できる機構により，OS と VMM の二重スケジューリングの問題を避けつつ VM 間の vCPU の受け渡しを柔軟におこなえるようにしている．しかし，vScale は NUMA アーキテクチャを考慮していない．

5. まとめと今後の予定

NUMA アーキテクチャが仮想化環境では十分サポートされていない問題に対して，VM 上でホストマシンの NUMA トポロジーを再現した vCPU の割り当てを動的に行う手法を提案し，その予備実験として静的に NUMA を再現した割り当てをした場合の性能評価をおこなった．実験によって，NUMA を再現した vCPU を割り当てをした場合はそうでない場合に比べて，ベンチマークの実行速度が平均 8.9%程度高速化するという結果が得られた．

今回は静的に割り当てた場合の評価を行なったが，今後は動的に vCPU を受け渡す手法を実装して，その性能を評価する予定である．

参 考 文 献

1) S. Wu, et al. “vProbe: Scheduling Virtual Machines on NUMA Systems”, In Proc. of 2016 IEEE International Conference on Cluster Computing (CLUSTER), 2016
2) L. Cheng, et al. “vScale: Automatic and Efficient Processor Scaling for SMP Virtual Machines”, In Proc. 11th European Conference on Computer Systems (EuroSys ’16), 2016.
3) http://parsec.cs.princeton.edu/