ORACLE VM

Oracleホワイト・ペーパー
2013年1月

Oracle VM 3：10GbEネットワークのパフォーマンス・チューニング

ORACLE

はじめに

このテクニカル・ホワイト・ペーパーでは、Oracle VM Server for x86をチューニングして、10GbE（10ギガビット・イーサネット）ネットワークを効果的に使用する方法について説明します。

ネットワーク・パフォーマンスに関する考慮事項

ネットワーク・パフォーマンスは、特に仮想マシン環境においては、全体的なシステム・パフォーマンスの極めて重要な構成要素です。ネットワークは、ライブマイグレーション、クラスタ・ハートビート、NFSやiSCSIの仮想ディスクへのアクセス、システム管理、ゲスト仮想マシン自身のネットワーク・トラフィックに対するさまざまなロールで使用されます。これらのロール、およびOracle VMネットワークのその他の側面については、『Looking "Under the Hood" at Networking in Oracle VM Server for x86』で説明しています。

10GbEネットワーク・インターコネクトは、ネットワーク集中型アプリケーションのパフォーマンスを強化するために、現代のデータセンターに配置されることが多くなっています。10Gの回線速度、あるいは期待されるスループットを達成することには困難が伴う場合があります。単一ストリームの送受信を使用する場合は特に難しく、仮想I/Oのオーバーヘッドと待機時間を増やしがちな仮想化システムではさらに難しくなります。

多くのCPUと大容量RAMを搭載するシステムの場合、これは大きな課題となります。これらのシステムは、仮想マシンのホストとしては非常に魅力的ですが、NUMA（Non-Uniform Memory Access）プロパティがあります。その結果、パケット間間隔のためにメモリ待機時間の変動の影響を受けやすい高速ネットワークでパフォーマンスとスケーラビリティが低下する可能性があります。パフォーマンスは、相互に作用するプロセスとカーネル・コンポーネント、およびアクセスされるRAMの場所が、同じCPUノードまたはコアで緊密に調整されているかどうかによって異なります。メモリ待機時間は、同じCPUノードへのアフィニティによって軽減されます。特定のアプリケーション・パフォーマンスのニーズに合わせて最適に調整するには、システムのCPUトポロジと能力を把握することが重要です。

NUMAの影響を示す例

次の例では、2種類のハードウェア・アーキテクチャ上の10Gネットワーク・パフォーマンスについて説明します。

Netperf TCP_STREAMが測定用に使用されました。最初の結果は、ネットワーク・パフォーマンス改善のための調整を行う前のものです。2個のVCPUと2GBのメモリを搭載したLinux仮想マシンがすべてのテストで使用されています。ネットワーク・パフォーマンスは、2つのマシン（物理ネットワーク・アダプタに直接アクセス）上のdom0インスタンス間、および1台のサーバー上のdom0ともう一方のサーバー上のゲスト仮想マシン間で測定されました。

- 適切なシステム・リソースを持つ非NUMAシステム
    - 2台のシステム。それぞれに2個のソケット、ソケット当たり4個のCPUコア、コア当たり2個のスレッド（合計16個のCPUスレッド）、および24GBのメモリ搭載
    - dom0。8個のVCPU、2Gのメモリ搭載

    - dom0 -> dom0 :~9.4Gb/秒
    - dom0（a） -> dom0（b）で稼働する1個のVM：~7.8Gb/秒
- 大量のリソースを持つNUMAシステム
    - 2台のシステム。それぞれに8個のソケット、ソケット当たり10個のCPUコア、コア当たり2個のスレッド（合計160個のCPU）、および256GBのメモリ搭載
    - dom0。160個のVCPU、5GBのメモリ搭載
    - dom0 -> dom0：~5.2Gb/秒
    - dom0（a） -> dom0（b）で稼働する1個のVM：~3.4 Gb/秒

大規模システムの結果は、CPUとメモリがより少ない非NUMAシステムより低い数値が出ました。dom0環境間のスループットは45%、dom0と隣接サーバーのゲスト間のスループットは56%低下します。このことから、名目上は同じ速度のマシンでも、チューニングが行われないとスケーラビリティが低いことがわかります。

パフォーマンス・チューニング

パフォーマンスを改善するため、大規模マシンで複数の重要なチューニングを行いました。これらの変更では、利用可能なCPUのサブセットへのCPUスケジューリングと割込み処理を制限することで、メモリ待機時間を改善しました。

また、高速ネットワーク用にTCPパラメータを調整しました。これらの変更は、小規模サーバーと大規模サーバーの両方に適しています。

注：これらの変更はガイドラインとして参考にし、自社のハードウェア・アーキテクチャとリソースに合わせて調整してください。幅広い種類の構成とワークロードでの徹底的なテストは実施されていません。保守性と管理作業への効果を考慮の上、パフォーマンス要件に照らして個別に評価する必要があります。次のような変更を行っています。

dom0 CPUの数

たとえば、大規模な8ソケットx86システムで、dom0 VCPUの数を160から20に減らしました。これは、NUMAシステムの最初のソケットのCPUスレッド数です。この変更により、最初のCPUソケットへのdom0のCPUスケジューリングが制限されて、メモリとL2キャッシュのアフィニティが確保されます。dom0 VCPUは固定化されたため、仮想CPUは物理CPUへの厳密な対応関係を持ちます（VCPU0 ->CPU0など）。

ソケットのCPUの数は、サーバーのCPUトポロジによります。CPUの数を判別するには、サーバーにログインして、`"xm info"`コマンドを実行します。`cores_per_socket`と`threads_per_core`の数が表示されます。これらの値の結果は、次のステップで使用するソケット当たりのCPUスレッド数です。

dom0のVCPUを固定化し、変更するには、

`"dom0_vcpus_pin dom0_max_vcpus=X"`をXenカーネルのコマンドライン（この場合はX=20）に追加して、再起動します。この例では、`/boot/grub/grub.conf`からの完全な行は、

`kernel /xen.gz dom0_mem=582M dom0_vcpus_pin dom0_max_vcpus=20`となります。

これは、このドキュメントで説明するチューニング手順でもっとも重要な部分です。

ポート割込みアフィニティ

ほとんどの10GポートはMSI-X割込みベクターを使用します。これらは利用可能なdom0 VCPU全体に分散されており、少なくとも1つがCPU0に関連付けられています。多くのデバイスでは、CPU0がデフォルトの割込みCPUであるため、CPUが別のデバイスからの割込みを処理している間に、10Gポートで割込みが発生すると、優先的に処理することができません。そのため、他の割込みサービス・ルーチンが終了するまで、ネットワーク割込みの処理が遅れます。10Gポート割込みベクターを、割込み処理がそれほど多くないCPUに移動すると、有効な場合があります。この例では、割込みアフィニティの変更方法を示します。

1. "cat /proc/interrupts| grep ethX"を実行してポートのIRQ IDを検索します。ethXをネットワーク・ポートの名前で置き換えてください。最初の列にIRQ IDが示されます。これがどのように見えるか（このページに収まる小規模サーバーで）、次に例を示します。

```
# cat /proc/interrupts|head -1; cat /proc/interrupts|grep eth
           CPU0       CPU1       CPU2       CPU3
 35:    3562070        243    1307766        249  PCI-MSI-edge eth4
 39:   11828156     108367     101470     108262  PCI-MSI-edge eth0
 40:   11748516   11991143   11688127   11988451  PCI-MSI-edge eth1
```

2. 各割込みベクターが関連付けられているCPUを検索します。そのようなポートが1つだけの場合、"cat /proc/irq/$irqid/smp_affinity"を実行します。$irqidをステップ1の値で置き換えてください。
   ポートが多数ある場合は、次のようなスクリプトを使用できます。

```
#!/bin/bash
for irqid in `cat /proc/interrupts| grep eth | awk
{'sub(":","",$1);print $1'}`;
do
 affinity=`cat /proc/irq/$irqid/smp_affinity`
 echo "irqid $irqid affinity $affinity"
done
```

これにより、IRQ番号、およびその割込みに関連付けられたCPUを示すビット・マスクが、CPU0を示す右端のビットから出力されます。たとえば、1の値はCPU0を示し、"f"の値はCPU0からCPU3を示します。CPU0に割り当てられたIRQがある場合、smp_affinityの内容を新しいビット・マスクで置き換えることで、使用率の高くないCPUに再割当てすることができます。たとえば、次のコマンドを実行すると、IRQ 323のCPUアフィニティはCPU2に設定されます。

```
# echo 00004 > /proc/irq/323/smp_affinity
```

このメカニズムを使用すると、割込み処理の負荷を静的に分散することができ、特定のCPUに負荷が偏っている場合に役立ちます。このように静的にIRQを割り当てる場合は、irqbalance サービスを停止する必要があります。もし、irqbalance サービスが起動していた場合は、このサービスを停止してください。

この変更は、Linuxのカーネル設定の変更が伴うため、慎重に評価する必要があります。管理者は、パフォーマンス達成のために変更が必要かどうか、CPUの負荷が割込み処理によって増大していないかを考慮する必要があります。インストールの標準およびカーネル設定の変更に関する最適レベルも考慮する必要があります。

プロセス・アフィニティ

NUMAシステムの場合、相互に作用するプロセス（この場合、XenのNetbackとNetfront）を同じCPUソケット上で実行することを推奨します。そうすることで、共有データ構造へのアクセスにかかるメモリ待機時間が軽減します。2個のVCPUのあるVMの場合、dom0が実行されている同じソケットのVCPUを選択し、VMのVCPUをこれらに固定します。この処理を実行するには、Oracle VM 3ユーティリティの`ovm_vmcontrol`コマンドを使用して、仮想CPU設定を取得および変更する方法を推奨します。Oracle VM 3ユーティリティは、Oracle VM 3のオプションのアドオンとして提供されています。ドキュメントおよびダウンロード方法は、Oracle VMダウンロード・ページに掲載されています。

ユーティリティをインストールしたら、Oracle VM Managerをホストするサーバーにログインし、次のコマンドを実行します。管理者のユーザーID、パスワード、およびゲストVM名の変数を必要に応じて置き換えてください。

```
# cd /u01/app/oracle/ovm-manager-3/ovm_utils

# ./ovm_vmcontrol -u $ADMIN -p $PASSWORD -h localhost ¥

            -v $GUEST -c vcpuget
```

これにより、ゲストが"Current pinning of virtual CPUs to physical threads"として実行できる一連の物理CPUが表示されます。CPUの固定化が行われていない場合、これがサーバー上のCPUの合計リストになります。この合計は、ゲストの仮想CPUの数より多いのが一般的です（たとえば、8個のCPUを搭載したサーバーがわずか1個のVCPUでゲストをホストする）。次に例を示します。

```
# ./ovm_vmcontrol -u admin -p Welcome1 -h localhost -v myvm -c vcpuget

Oracle VM VM Control utility 0.5.2.

Connected.

Command :vcpuget

Current pinning of virtual CPUs to physical threads : 0,1,2,3,4,5,6,7
```

ゲスト・ドメインを最初のCPUソケットの物理CPUに固定するには、次のコマンドを実行します。ソケット当たりのCPU数より1つ少ない値でNを置き換えます（CPUはゼロからカウントされるため）。

```
# ./ovm_vmcontrol -u $ADMIN -p $PASSWORD -h localhost ¥

            -v $GUEST -c vcpuset 0-N
```

ソケット当たり20個のCPUを搭載したサーバーの場合、値は`vcpuset 0-19`になります。次に例を示します。

```
# ./ovm_vmcontrol -u admin -p Welcome1 -h localhost ¥

            -v myvm -c vcpuset -s 0-19

Oracle VM VM Control utility 0.5.2.

Connected.

Command : vcpuset

Pinning virtual CPUs
```

```
Pinning of virtual CPUs to physical threads  '0-19' 'myvm' completed
```

この例では、仮想CPUの数より多い多数の物理CPUにゲストを固定します。その結果、ゲストはOracle VMスケジューラにより、最初のソケットのどの物理CPUにも割り当てられます。物理CPUが仮想CPUの数とまったく同じになるように任意で指定できます。その結果、レベル1のキャッシュ・ヒット率は改善されますが、使用率の低いCPUがある場合は、負荷の高い物理CPUにゲストを固定することもできます。

物理CPUを指定するもう1つの方法は（Oracle VM 3ユーティリティがインストールされていない場合）、VM構成ファイルを手動で特定、編集することです。まず、Oracle VM Managerのユーザー・インタフェースを使用して、リポジトリと仮想マシンのIDを表示します。次に、プール内のサーバーにrootとしてログインし、ディレクトリ/OVS/Repositories/$RepositoryID/VirtualMachines/$VMID内のファイルvm.cfgを編集します。vm.cfgの"cpus"行を次のように変更します。

```
vcpus = 2
```

```
cpus = "1,4"
```

これにより、VMの2個のVCPUがCPU1とCPU4に固定されます。コマンド"xm vcpu-list"を使用すると、dom0を含めたドメインに正しい数のCPUと固定化があるかどうかを確認できます。`ovm_control`コマンドは、内部ファイルの内容の"アウトオブバンド"編集ではなく、実証済みの方法であるため、`ovm_control`コマンドを推奨します。

これらの例では、ゲスト仮想CPUをdom0とともに最初のソケットに固定します。その結果、NUMAのオーバーヘッドが回避されますが、他のソケットの使用されなくなります。この処理をすべてのゲストに対して実行すると、複数のソケットに拡張可能なサーバーを使用することの利点が失われます。したがって、この方法は、もっとも重要なネットワーク・パフォーマンス要件を持つ仮想マシンに対しては選択的に使用する必要があります。他のゲストは、CPUを固定化せずに他のCPUソケットで実行でき、最初の過負荷のソケットにすべて割り当てる場合よりも良好なパフォーマンスが実現します。また、CPUの固定化は、やたらに実行すべきではない管理オーバーヘッドでもあります。

仮想マシン内のNUMAの待機時間とキャッシュ・ミスは、ネットワークの問題に関係なく、パフォーマンスを低下させることがあるため、あまり多くのCPUを持たないように仮想マシンのサイジングを行うことが重要です。ベスト・プラクティスとして、予想されるCPU負荷とある程度の余裕分を見込んでサイジングする方法があります。Oracle VMでは、ゲストが起動時に受け取るCPU数、例外的な負荷のときに調整できる最大数を設定できます。この設定により、仮想CPUが複数のCPUソケットにまたがって、NUMA待機時間を生じさせる可能性が低くなり、CPUコアのウォーム・キャッシュの再使用が増え、複数のCPUにまたがってシリアライズする必要のあるデータ構造の内部ロック競合が軽減されます。

TCPパラメータ設定

ほとんどのLinux VMディストリビューションのデフォルトのTCPパラメータは保守的で、100Mb/秒または1Gb/秒のポート速度を処理するように調整されているため、バッファ・サイズが10Gbネットワークには小さすぎる結果となります。これらの値を変更すると、10Gbs VMネットワークでパフォーマンスを大幅に向上できます。RTTとBDPは、TCPパラメータの影響を受ける必要不可欠な値です。ラウンドトリップ時間（RTT）は、パケットが宛先に到達して送信元に戻ってくるまでの合計時間です。ギガビット・ネットワークの場合、この値は1～100ミリ秒になります。RTTはpingを使用して測定できます。帯域幅遅延積（BDP）は、いつでも転送できるデータ量であり、リンク帯域幅とRTT値の積です。RTTを100ミリ秒と想定した場合、次の値になります。

BDP =（.1s *（10 * 10^9 ）ビット）= 134217728バイト

転送されているバイトの最大数を許可し、トラフィック・スロットリングが回避されるように、バッファ・サイズを調整する必要があります。仮想マシンならびにdom0で次の値を設定できます。

```
Maximum receive socket buffer size (size of BDP)
# sysctl -w net.core.rmem_max=134217728

Maximum send socket buffer size (size of BDP)
# sysctl -w net.core.wmem_max=134217728

Minimum,initial,and max TCP Receive buffer size in Bytes
# sysctl -w net.ipv4.tcp_rmem="4096 87380 134217728"

Minimum, initial, and max buffer space allocated
# sysctl -w net.ipv4.tcp_wmem="4096 65536 134217728"

Maximum number of packets queued on the input side
# sysctl -w net.core.netdev_max_backlog=300000

Auto Tuning
# sysctl -w net.ipv4.tcp_moderate_rcvbuf=1
```

図1：上記のコマンドを使用して、TCPパラメータを調整

これらの設定は即座に適用されますが、再起動後は適用されません。これらの値を固定するには、次の行を`/etc/sysctl.conf`に追加します。

```
net.core.rmem_max = 134217728
net.core.wmem_max = 134217728
net.ipv4.tcp_rmem = 4096 87380 134217728
net.ipv4.tcp_wmem = 4096 65536 134217728
net.core.netdev_max_backlog = 300000
net.ipv4.tcp_moderate_rcvbuf =1
```

図2：上記の行を/etc/sysctl.confに追加

Oracle VM 3.2 より前のバージョンでは、bridge デバイスの netfilter をOFFにしてください。

Oracle VM 3.2 以降では、自動的にOFFに設定されています。

```
# sysctl -w net.bridge.bridge-nf-call-iptables=0
# sysctl -w net.bridge.bridge-nf-call-arptables=0
# sysctl -w net.bridge.bridge-nf-call-ip6tables=0
```

有効になった設定を表示するには、次のように、`sysctl`の後にパラメータ名を入力します。

```
# sysctl net.core.rmem_max
```

ジャンボ・フレーム

デフォルトのMTU値は1500で、ほとんどの10Gポートは最大64KB MTU値に対応しています。パフォーマンスを改善し、より安定させるには、9000のMTU値が適切です。ジャンボ・フレームは、リリース

3.1.1以降のOracle VM Server for x86でサポートされています。MTUを変更する場合、変更した値を通信ホスト間のすべてのデバイス（ルーターなど）で設定する必要があります。スイッチ・ベンダーまたはスイッチ・モデルによっては、一部のスイッチ構成からジャンボ・フレーム（QoS、トランキングのほか、VLANなども）の使用が除外されています。

NICオフロード機能

NICによってサポートされるオフロード機能を使用すると、CPUの消費を軽減でき、パフォーマンスの大幅な改善を達成できます。次の設定は、テスト中に使用した値を示します。

ポートをブリッジ・インタフェースとして追加する場合は、Large Receive Offload（LRO）を**OFF**に設定する必要があります。デフォルトではポートがブリッジ・インタフェースに追加された際に自動的にOFFに設定されます。次に、これらの設定の例を示します。

```
# ethtool -k ethX
Offload parameters for ethX:
rx-checksumming: on
tx-checksumming: on
scatter-gather: on
tcp-segmentation-offload: on
udp-fragmentation-offload: off
generic-segmentation-offload: on
generic-receive-offload: on
large-receive-offload: off
```

図3：ethtool 設定の例

結果

次の結果は、上記のチューニング・オプションを適用した後のものです。

dom0 -> dom0:~9.5Gb/秒 － ワイヤー・スピードとベースラインの非NUMA値に非常に近く、ベースラインのNUMA値より83%高速

dom0（a）-> dom0（b）で稼働する1個のVM：~8.2Gb/秒 － 非NUMAベースラインより5%高速で、ベースラインNUMA値より240%高速

1つのホスト上で稼働するVM -> 異なるホストで稼働する1個のVM：~8.2Gb/秒 － 2つのブリッジ接続を通過しても高帯域幅を達成

システム・アーキテクチャ、ネットワーク、リソース、およびアプリケーションの要件によっては、このホワイト・ペーパーのすべてが必要とは限らない場合があります。NUMAシステムでのもっとも重要な対策は、dom0 VCPUの数を減らして、これらすべてを1個のCPUソケット/ノードに収めることです。

まとめ

ネットワーク・パフォーマンスは、システム・パフォーマンス全体における重要な構成要素であり、システム構成とチューニング・オプションによって大きく変わることがあります。NUMA動作を有効にした大規模システムのパフォーマンスは、簡単なチューニング手順を行うことで劇的に改善できます。Oracle VM Server for x86 3.1.1以降でこれらのチューニング方法を使用できます。

すべての状況で機能するチューニング・ルールは1つもありません。チューニングに関する推奨事項は、システム・パフォーマンス全体の他の側面とのバランスを図り、現実的な構成でテストして、実際のパフォーマンスの効果を判別する必要があります。記載したチューニング手順の一部を適用して、許容パフォーマンスを達成したら、それ以上のチューニングを適用しなくても良い場合があります。チューニングは、インストールの標準、サポート機能、および管理のベスト・ベストプラクティスに照らして、慎重に評価してください。これらの調整を実施することで、パフォーマンスを大幅に改善できます。

Oracle VM 10GbEネットワークのパフォーマンス・チューニング
2013年1月
著者：Adnan Misherfi

貢献者：
Jeff Savit

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

海外からのお問い合わせ窓口：
電話：+1.650.506.7000
ファクシミリ：+1.650.506.7200

www.oracle.com