テクニカルホワイトペーパー

Veritas InfoScale 7.4.2 for RHEL on AWS
パフォーマンステスト結果大公開

2020 年 9 月

ベリタステクノロジーズ合同会社
テクノロジーセールス&マーケティング本部

VERITAS
The truth in information.

免責事項

ベリタステクノロジーズ合同会社は、この文書の著作権を留保します。また、記載された内容の無謬性を保証しません。Veritas InfoScale は将来に渡って仕様を変更する可能性を常に含み、これらは予告なく行われることもあります。なお、当ドキュメントの内容は参考資料として、読者の責任において管理/配布されるようお願いいたします。

目次

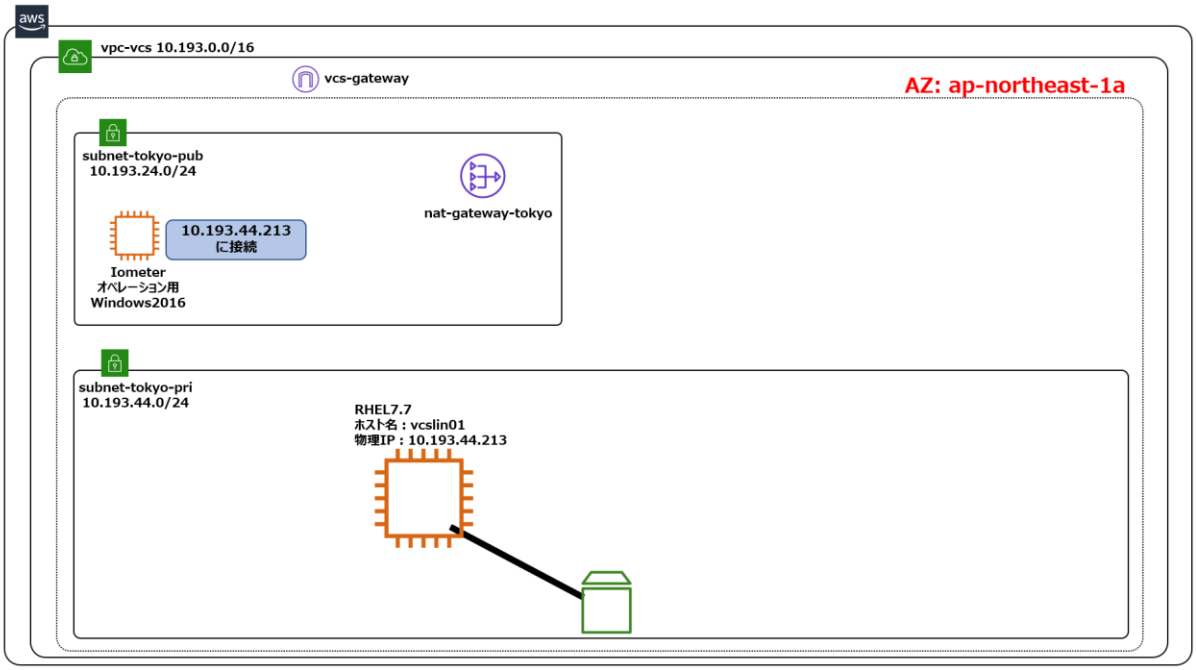
1. はじめに

本書の目的

本書は、InfoScale 7.4.2 RHEL 版を AWS 上に構築した代表的な構成における I/O パフォーマンステスト結果を元に、構成毎に達成可能なパフォーマンスについての理解を目的に作成されています。また、パフォーマンスを向上させるための簡単なチューニングについても説明しています。

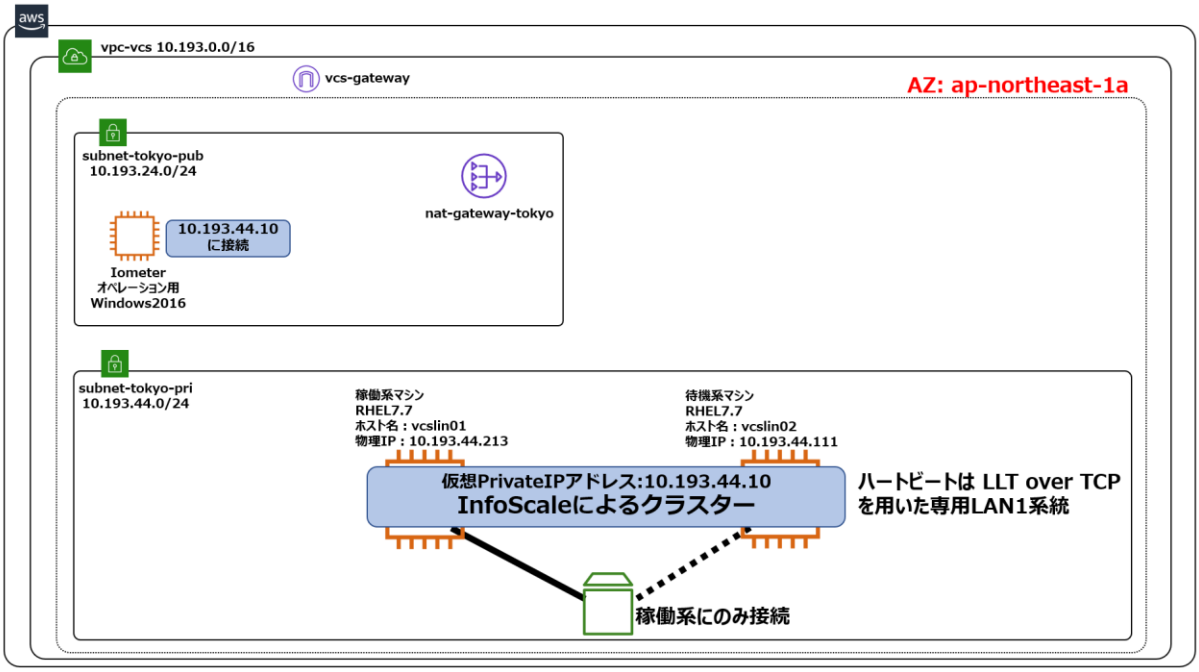
2. パフォーマンステストを行う条件

I/O パフォーマンステストを行った 5 種類の構成



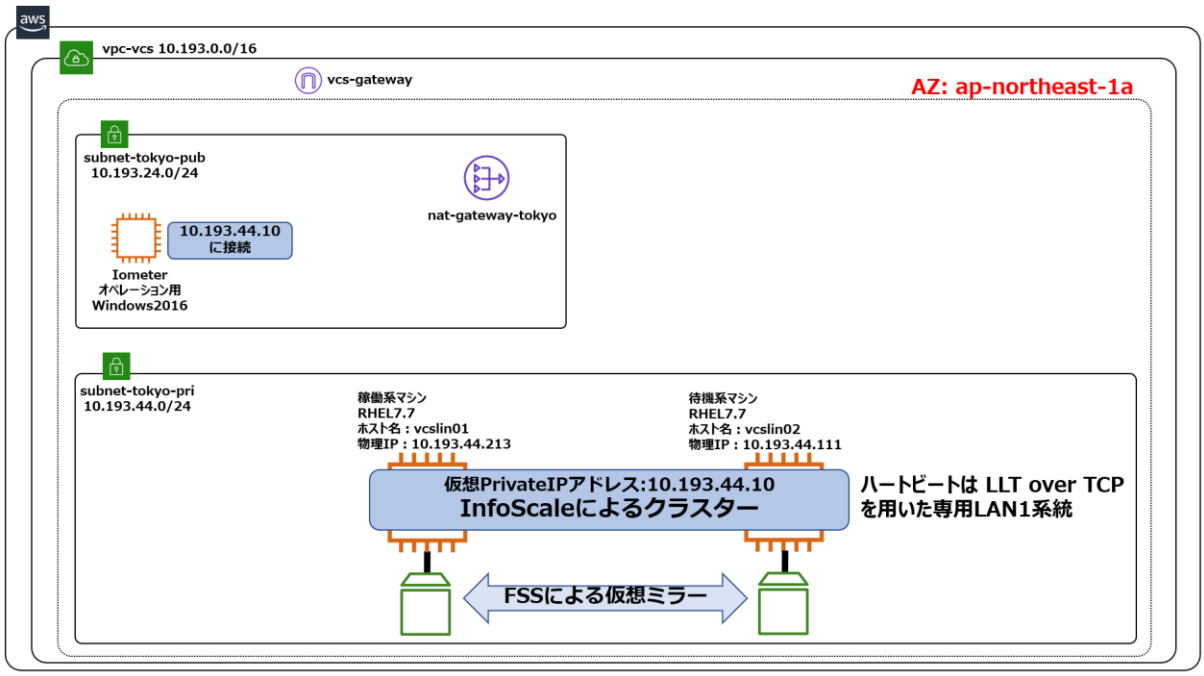
構成 1 シングルノード構成

本書における基準となるべき構成です。AWS 上の非クラスター構成のインスタンスに InfoScale を導入して EBS を管理対象にすると、EBS の拡張・追加時のファイルシステム拡張がサービス停止を伴わず行える等のメリットがあります（詳しくは、https://www.veritas.com/support/en_US/doc/InfoScale7.4.1_RHEL_on_AWS_FSS_storage_maintenance 及び https://www.veritas.com/support/en_US/doc/InfoScale7.4.1_Win_on_AWS_VVR_storage_maintenance をご参照ください）。AWS 上の非クラスター構成のインスタンスに InfoScale を導入した場合に実現できる I/O 性能の基準としてご活用ください。また、この構成で得られた I/O パフォーマンスに対して、他の 4 つの構成がどれだけ劣るかを比べる事で、クラスター構成毎の I/O パフォーマンスに関する留意点が明らかになります。



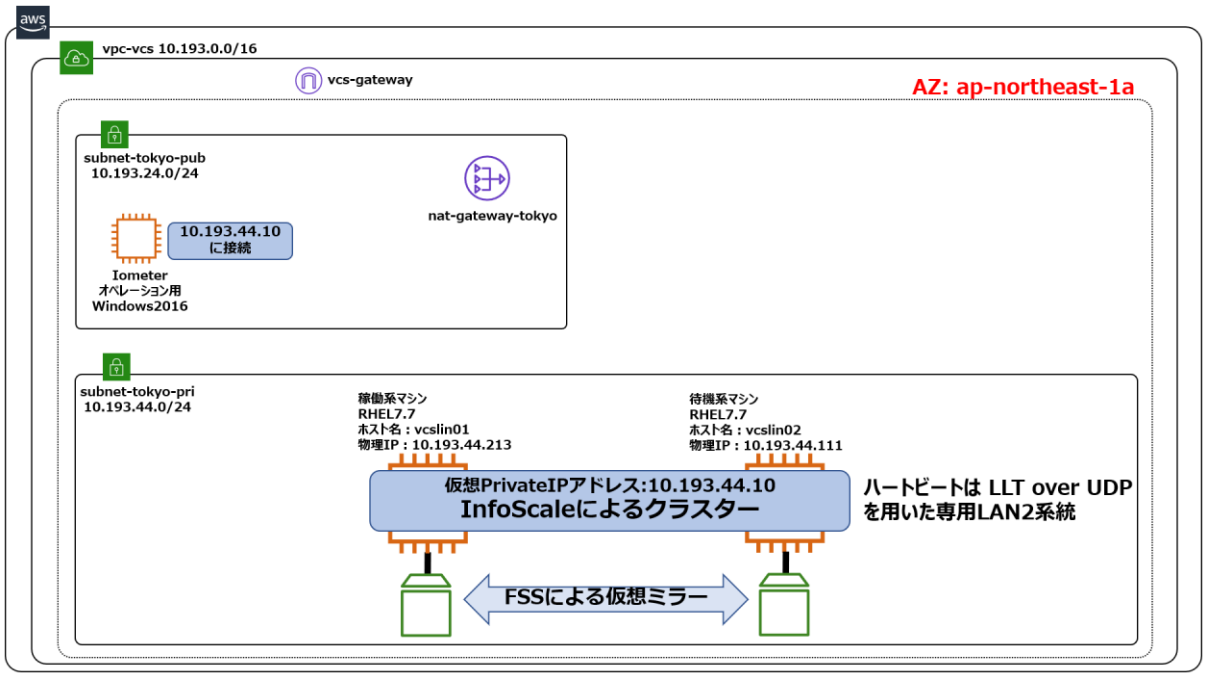
構成 2 共有ディスク型クラスター構成

構成 1 に対してどれだけ I/O 性能が劣化するかを考察する事で、クラスター化に伴うオーバーヘッドの度合いを把握する事ができます。構成 1 と構成 2 では、1 つのインスタンスが 1 つの EBS に専用アクセスする事は変わりませんので、理論上はクラスター化のオーバーヘッドのみが顕著化します。尚、上記構成の構築方法については https://www.veritas.com/content/support/en_US/doc/InfoScale7.4.1_RHEL_on_AWS_deploy_EBS_PrivateIP_JP をご参照ください。



構成 3 FSS を用いた AZ 内クラスター構成（ハートビートは TCP1 系統）

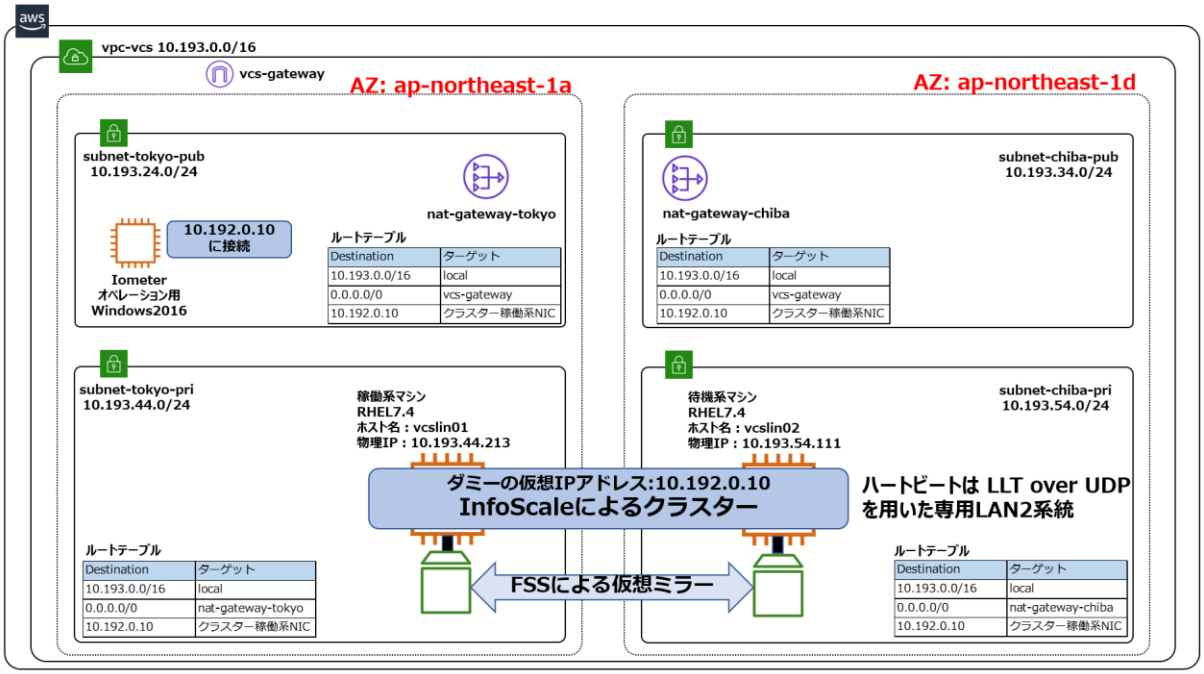
構成 2 に対してどれだけ I/O 性能が劣化するかを考察する事で、FSS による仮想ミラーに伴うオーバーヘッドの度合いを把握する事ができます。構成 2 と構成 3 の相違点は仮想ミラーを行っているかどうかのみですので、理論上は仮想ミラーのオーバーヘッドのみが顕著化します。尚、上記構成の構築方法については https://www.veritas.com/content/support/en_US/doc/InfoScale7.4.1_RHEL_on_AWS_deploy_FSS_PrivateIP_JP をご参照ください。



構成 4 FSS を用いた AZ 内クラスター構成（ハートビートは UDP2 系統）

構成 3 と I/O 性能を比較する事で、FSS による仮想ミラーを実現するハートビート LAN の構成が I/O 性能に与える影響を把握する事ができます。構成 3 と構成 4 の相違点はハートビート LAN が TCP/IP を 1 系統持っているか、UDP/IP を 2 系統持っているかです。AWS の NIC の性能を上回るような I/O 負荷をかけた場合は、2 系統で負荷分散できる構成 4 が勝り、そうでない場合はオーバーヘッドの小さい構成 3 が勝る事が予想できます。尚、上記構成の構築方法については

https://www.veritas.com/content/support/en_US/doc/InfoScale7.4.1_RHEL_on_AWS_deploy_FSS_PrivateIP_JP をご参照ください。



構成 5 FSS を用いた AZ 跨ぎクラスター構成（ハートビートは UDP2 系統）

構成 4 と I/O 性能を比較する事で、AZ を跨ぐことによるネットワーク遅延が I/O 性能に与える影響を把握する事ができます。構成 4 と構成 5 の相違点は AZ を跨ぐか跨がないかのみです。FSS による仮想ミラーの性能、特に IOPS は使用するハートビートネットワークの Latency に依存しますので、構成 4 が勝る事が予想できます。尚、上記構成の構築方法については https://www.veritas.com/support/en_US/doc/InfoScale7.4.1_RHEL_on_AWS_deploy_FSS_OverlayIP をご参照ください。

使用する EBS の種類

本書では、検証に使用する EBS は「gp2」を使用します。gp2 には「バーストバケット」という機能があり、バーストチケットに余裕があれば一定の IOPS を保証します。一般に EBS の IOPS は容量に比例する為、容量の小さな EBS を使用する場合は十分な IOPS が得られない事がありますが、gp2 を使用する事でこのデメリットを回避する事ができます。ただし、バーストチケットを使い切らないよう注意が必要です。gp2 やバーストバケットに関しては、AWS の情報（https://docs.aws.amazon.com/ja_jp/AWSEC2/latest/UserGuide/ebs-volume-types.html 等）を参照してください。

マシン環境

本書で使用した検証環境は以下の通りです

OS：RHEL7.7

InfoScale 7.4.2

インスタンスタイプ：t2.large（2Core, 8Gbyte メモリ）

EBS：gp2 の Standard SSD 5Gbyte

3. 使用するパフォーマンス計測ツール

本書では、パフォーマンス計測ツールとして、フリーのツール：IOMETER を使用します。

IOMETER について

IOMETER には 2 つの機能があります。1 つ目は負荷生成器(I/O 処理を実行してシステムに負荷をかける)としての機能です。2 つ目は、測定ツール(I/O 処理のパフォーマンスとシステムに対する影響を検査して記録する)の機能です。また、2 つのコンポーネント：Iometer と Dynamo で構成されます。

Iometer は、制御プログラムです。Iometer の GUI を使用して、負荷の設定、動作パラメーターの設定、テストの開始および停止を行います。Iometer は Dynamo に処理を指示し、処理結果を収集し、出力ファイルにまとめます。一度に実行できる Iometer のタスクは 1 つのみ、本書では windows のオペレーション用マシン上で実行しています。

Dynamo は、負荷生成器です。ユーザーインタフェースはありません。Iometer の指示により、Dynamo は I/O 処理を実行し、パフォーマンス情報を記録した後、データを Iometer に返します。

IOMETER のインストールとセットアップ

まず、Iometer は、Windows 上でのみ実行されます。Linux には、Iometer がインストールされている Windows と連携して負荷を掛けるために、Dynamo をインストールします。本書では、Linux 上のパフォーマンステストを行う為、オペレーション用の Windows インスタンスに Iometer をインストールします。インストールに際して、Iometer.org から Iometer をダウンロードします。インストール後のセットアップに関しても前述のサイト等をご参照ください。https://www.dell.com/support/article/ja-jp/how10228/iometer%E3%82%92%E4%BD%BF%E7%94%A8%E3%81%97%E3%81%A6poweredge%E3%82%B5%E3%83%BC%E3%83%90%E3%83%BC%E3%81%AE%E3%83%91%E3%83%95%E3%82%A9%E3%83%BC%E3%83%9E%E3%83%B3%E3%82%B9%E3%82%92%E3%83%86%E3%82%B9%E3%83%88%E3%81%99%E3%82%8B%E6%96%B9%E6%B395?lang=ja にも親切にまとめられています。

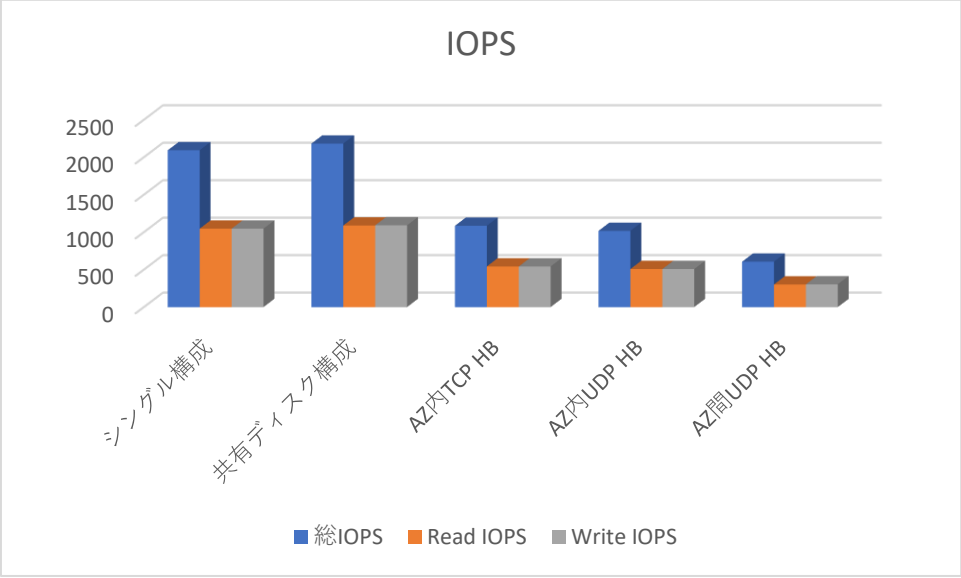
4. テスト結果

I/O サイズ 4Kbyte、Read/Write 比率 5 対 5、ランダムアクセスでの結果

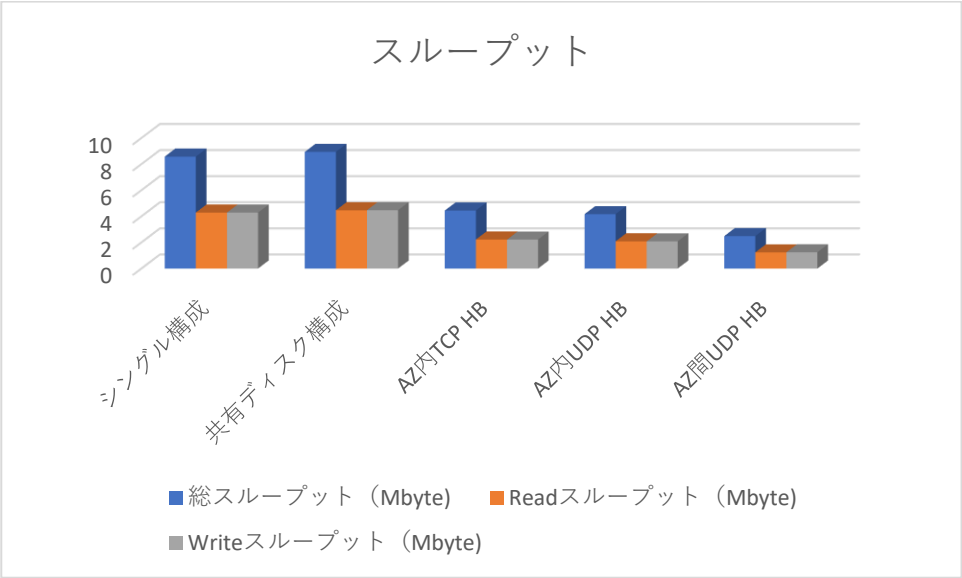
今回のテストでは、以下の 9 種類のパターンでテストを行いました。

・I/O サイズ 4Kbyte Read/Write 比率 10 対 0、ランダムアクセス
・I/O サイズ 4Kbyte Read/Write 比率 5 対 5、ランダムアクセス
・I/O サイズ 4Kbyte Read/Write 比率 0 対 10、ランダムアクセス
・I/O サイズ 4Kbyte Read/Write 比率 10 対 0、シーケンシャル
・I/O サイズ 4Kbyte Read/Write 比率 5 対 5、シーケンシャル
・I/O サイズ 4Kbyte Read/Write 比率 0 対 10、シーケンシャル
・I/O サイズ 64Kbyte Read/Write 比率 10 対 0、シーケンシャル
・I/O サイズ 64Kbyte Read/Write 比率 5 対 5、シーケンシャル
・I/O サイズ 64Kbyte Read/Write 比率 0 対 10、シーケンシャル

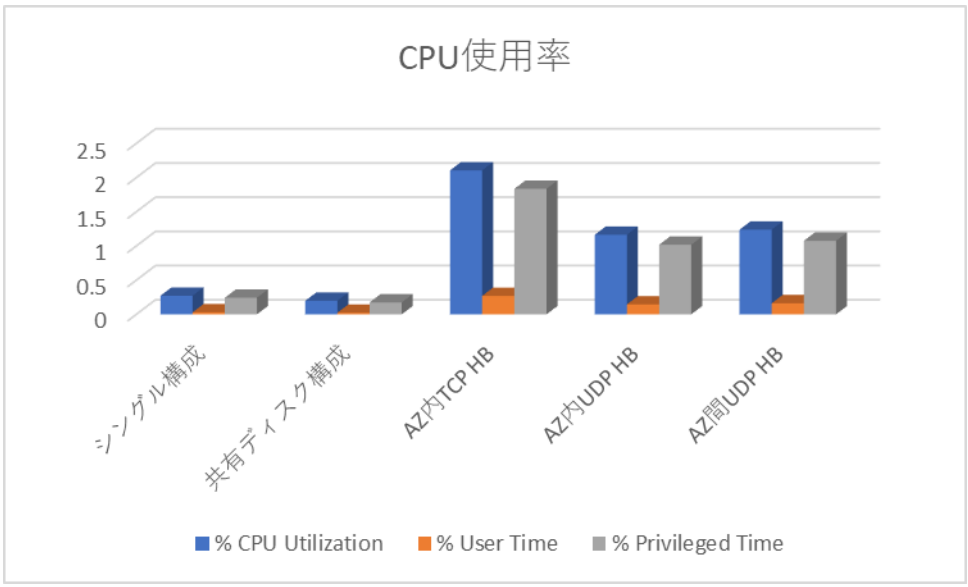
各構成の比較においては、9 種類のいずれでも概ね同じ傾向となりました。本書では、最も代表的な「I/O サイズ 4Kbyte、Read/Write 比率 5 対 5、ランダムアクセス」についての結果を紹介し、考察を述べます。



IOPS に関しては想定通り、シングル構成と共有ディスククラスター構成がほぼ同じ結果となりました。共有ディスククラスター構成がわずかに上回ったのは誤差と思われます。この 2 つとも 2000IOPS 程度の性能を発揮しました。バーストバケットが保証する IOPS が 3000 であった事と、IOMETER が掛けられる負荷を考えると、想定通りの結果と言えます。AZ 内で FSS による仮想ミラーを行った 2 つの構成は、先の 2 つの構成に比べて半分程度の IOPS です。これは、仮想ミラーによるオーバーヘッドと考えられます。最後に、AZ を跨いで FSS による仮想ミラーを行った構成では、AZ 跨ぎに起因するハートビート遅延により、さらに性能劣化が見られました。



スループットに関しては、IOPS と全く同じ比較結果となりました。これは、スループットは IOPS に I/O サイズを乗算したものだからです。具体的なスループット値も、例えばシングルノード構成に着目した場合、前ページの IOPS が 2000 だったので、それに I/O サイズである 4Kbyte を乗算すると 8Mbyte になり、上記のグラフと一致します。



CPU 使用率に関しては、IOPS やスループットとは全く異なる比較結果となりましたが、これも想定通りです。I/O に関連して特別な処理を一切行わないシングルノード構成と共有ディスク構成は、非常に低い結果となりました。一方、FSS による仮想ミラーを行う残りの 3 つの構成は、最大で 2%程度のオーバーヘッドが発生しています。グラフでは突出しているように見えますが、システム全体から見れば 2%は無視しても良い値です。尚、仮想ミラーの同期処理を行う為のネットワーク通信において様々なチェックを行っている分、TCP/IP を用いた構成は若干 CPU 使用率が高くなっています。

考察

シングルノード構成

AWS 上のシングルノード（非クラスター構成）の RHEL インスタンス上に、ストレージの管理性向上を理由に InfoScale を導入する場合、InfoScale の導入による CPU オーバーヘッドは誤差の範囲であり、I/O 性能もネイティブ環境と大差ないと言えます。

共有ディスク型のクラスター構成

InfoScale を用いた共有ディスク型のクラスター構成の I/O 性能や CPU オーバーヘッドは、非クラスター構成と大差ない値でした。従って、AZ 内でのクラスター構成を検討する際は、パフォーマンスの観点では共有ディスク型のクラスターを選択される事をお勧めします。

仮想ミラーを用いたクラスター構成

FSS による仮想ミラーを用いた 3 つのパターンのいずれも、シングルノードや共有ディスク型と比べて I/O 性能や CPU オーバーヘッドで劣る結果となりました。従って、仮想ミラーを行う必然性が無い場合（AZ 内でのクラスター）は、パフォーマンスの観点では仮想ミラー型クラスターはお勧めできません。しかし、AZ を跨いでのクラスターでは共有ディスク構成が使用できない為、仮想ミラーが有力な選択肢の 1 つになります。仮想ミラーの他にレプリケーションを使用する事も可能ですが、話を分かりやすくするため、あえて本書ではレプリケーションには言及しません。

つまり、AZ を跨いでのクラスターを想定し、仮想ミラーを用いたクラスターの I/O 性能を向上させるチューニングが必要という事です。チューニングについては、次の章で詳しく説明します。

5. 仮想ミラーのパフォーマンスを向上させるチューニング

仮想ミラー型クラスターの I/O パフォーマンスを向上させるには、ハートビートのチューニングが効果的です。なぜなら、仮想ミラー型のクラスターは、個々のクラスターノードのローカルディスクの同期をハートビート経由で行うからです。ハートビートのチューニングは、以下の 3 つの方法があります。

1．MTU サイズ拡張

2．RHEL の通信バッファサイズ拡張

3．InfoScale のハートビートドライバーである LLT のフロー制御閾値の拡張

それぞれ向き不向きがありますので、個別に説明します。

MTU サイズ拡張によるパフォーマンス向上

ローカルディスクの同期をハートビート経由で行う際、MTU サイズが大きければデータ転送の単位が大きくなるので、転送効率が良くなります。ただし、データの書き込みサイズが小さい場合は、逆に転送効率が悪くなるので注意が必要です。以下のようなケースでチューニングが有効です。

1．バッチ系の処理

2．シーケンシャルアクセス

3．書込みが多い

例えば、NIC の MTU が 1500 バイトで、ファイルシステムのブロックサイズが 8K である場合、MTU サイズを 8K より大きくする事で、パフォーマンス向上が期待できます。

以下は、NIC：eth1 の MTU サイズを 9000 バイトに変更する例です。

```
# ifconfig -a | grep mtu | grep eth1
eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500

# ifconfig eth1 mtu 9000

# ifconfig -a | grep mtu | grep eth1
eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 9000
```

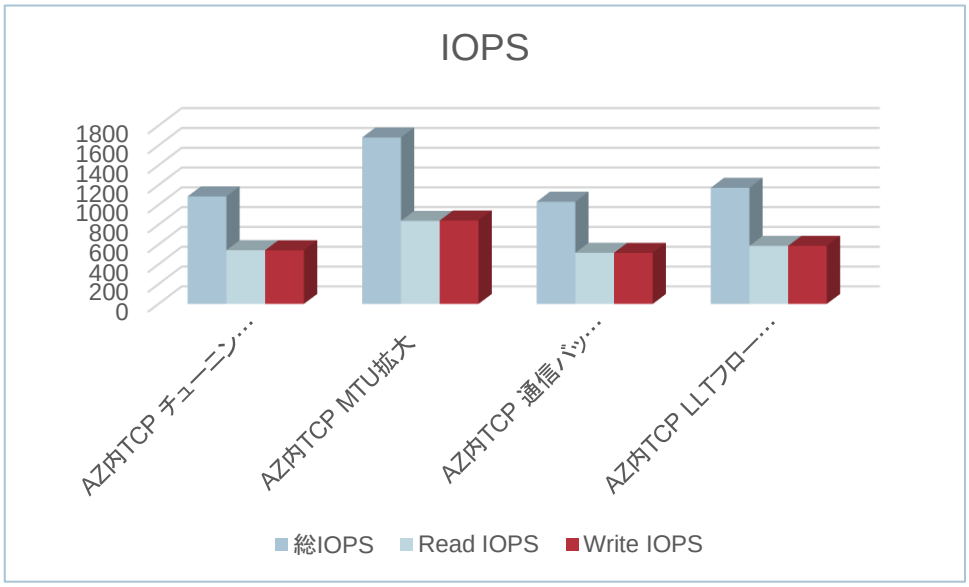
以下は、AZ 内で TCP/IP をハートビートに用いたクラスターにおいて、I/O サイズ 4Kbyte、Read/Write 比率 5 対 5、シーケンシャルアクセスでのパフォーマンス比較です。比較したのは、以下 4 パターンです。

1．チューニング無し

2．MTU サイズを 1500 バイトから 9000 バイトに拡張

3．RHEL の通信バッファサイズをデフォルトの 4 倍に拡張

4．LLT フロー制御の閾値をデフォルトの 2 倍に拡張



このパターンでは、4Kbyte のデータが連続的にハートビートに流れるので、MTU サイズの拡張が最も効果が高い結果となりました。

RHEL の通信バッファサイズ拡張によるパフォーマンス向上

ローカルディスクの同期をハートビート経由で行う際、RHEL の通信バッファサイズが大きければデータ転送の効率が向上するパフォーマンスが良くなります。ただし、バッファサイズが使用できるメモリ量を上回らないようにする必要があります。以下のようなケースでチューニングが有効です。

1．バッチ系の処理

2．クラスターのハートビートに UDP を選択している

3．書込みが多く且つ I/O ブロックサイズが大きい

RHEL の通信バッファとチューニング方法についての詳細は、Red Hat 社のドキュメント（例えば https://access.redhat.com/documentation/ja-jp/red_hat_enterprise_linux/6/html/performance_tuning_guide/s-network-dont-adjust-defaults ）を参照してください。

以下は、各種バッファサイズを 8Mbyte に拡張する場合の例です。このチューニングを有効にするには Reboot が必要です。

```
# cat /etc/sysctl.conf
net.core.rmem_default=8388608
net.core.wmem_default=8388608
net.core.rmem_max=8388608
net.core.wmem_max=8388608
net.core.optmem_max=8388608
net.ipv4.tcp_rmem="4096 87380 8388608"
net.ipv4.tcp_wmem="4096 65536 8388608"
```

以下は、Reboot 後に net.core.rmem_default の値を確認する例です

```
# cat /proc/sys/net/core/rmem_default
8388608
```

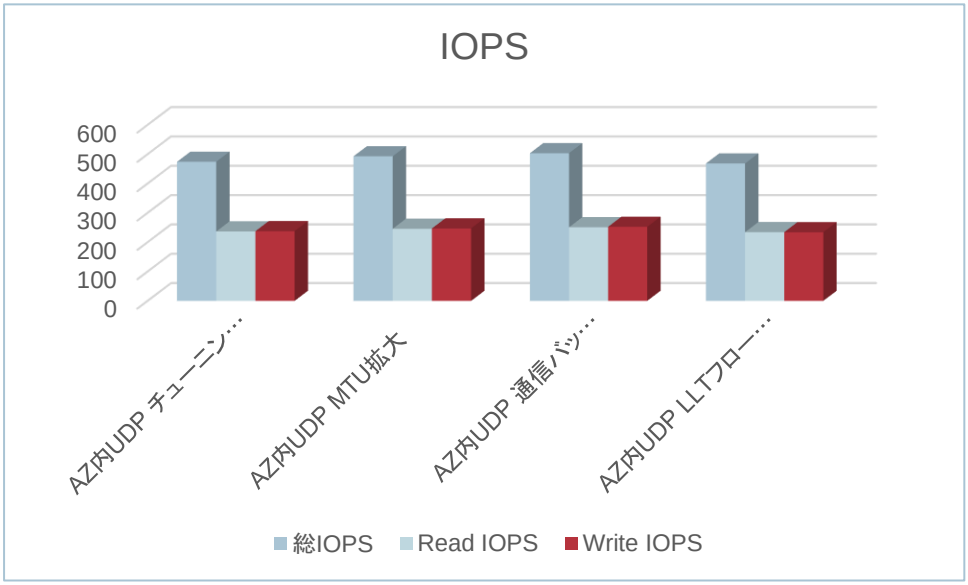
以下は、AZ 内で UDP/IP をハートビートに用いたクラスターにおいて、I/O サイズ 64Kbyte、Read/Write 比率 5 対 5、シーケンシャルアクセスでのパフォーマンス比較です。比較したのは、以下 4 パターンです。

1．チューニング無し

2．MTU サイズを 1500 バイトから 9000 バイトに拡張

3．RHEL の通信バッファサイズをデフォルトの 4 倍に拡張

4．LLT フロー制御の閾値をデフォルトの 2 倍に拡張



このパターンでは、どのチューニングもあまり差はありませんが、通信バッファサイズの拡張が他よりも若干効果が高い結果となりました。

LLT フロー制御閾値の拡張によるパフォーマンス向上

InfoScale のハートビートは、LLT という独自のプロトコルで行われています。LLT は送信キューに大量のデータが溜まる事を防止するためにフロー制御を行っています。フロー制御には閾値があり、キューに溜まったデータの量がハイウォーター値を超えると、溜まったデータの量がローウォーター値を下回るまで、キューの受付を停止します。従って、バースト的にデータ転送が発生する場合は、フロー制御によってデータ転送効率が低下します。逆に言うと、フロー制御閾値（ハイウォーター値）を拡張する事で、データ転送の効率が向上しパフォーマンスが良くなります。ただし、キューのサイズが使用できるメモリ量を上回らないようにする必要があります。以下のようなケースでチューニングが有効です。

1．AZ 跨ぎなど、ネットワークの Latency が大きい

2．書込みが多い

LLT フロー制御閾値とチューニング方法についての詳細は、InfoScale のマニュアル https://sort.veritas.com/DocPortal/pdf/142044775-142044783-1 の 1155 ページを参照してください。

以下は、LLT フロー制御閾値の変更例です。このチューニングは、即時有効になります。

```
#  lltconfig -F query
Current LLT flow control values (in packets):
   lowwater  = 800
   highwater = 1000
   rportlowwater  = 800
   rporthighwater = 1000
   largepktlen    = 1024
   window    = 50
   ackval    = 10
   sws       = 40
   linkburst = 32
# lltconfig -F highwater:2000
# lltconfig -F rporthighwater:2000
# lltconfig -F query
Current LLT flow control values (in packets):
   lowwater  = 800
   highwater = 2000
   rportlowwater  = 800
   rporthighwater = 2000
   largepktlen    = 1024
   window    = 50
   ackval    = 10
   sws       = 40
   linkburst = 32
```

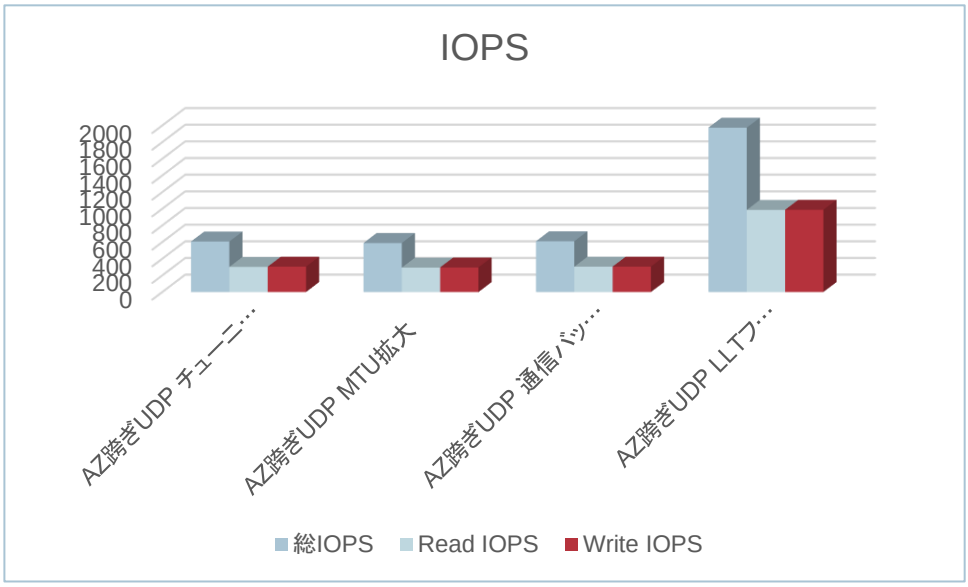
以下は、AZ を跨いだ UDP/IP をハートビートに用いたクラスターにおいて、I/O サイズ 4Kbyte、Read/Write 比率 5 対 5、シーケンシャルアクセスでのパフォーマンス比較です。比較したのは、以下 4 パターンです。

1．チューニング無し

2．MTU サイズを 1500 バイトから 9000 バイトに拡張

3．RHEL の通信バッファサイズをデフォルトの 4 倍に拡張

4．LLT フロー制御の閾値をデフォルトの 2 倍に拡張



このパターンでは、ハートビートの Latency が大きいためフロー制御が頻発します。それを抑制するためのチューニングが最も効果が高い結果となりました。AZ を跨ぐクラスターで FSS による仮想ミラーを使用しパフォーマンスを向上させたい場合は、このチューニングを行う事をお勧めします。

ベリタステクノロジーズについて

Veritas Technologies はエンタープライズデータ管理のグローバルリーダーです。複雑化した IT 環境においてデータ管理の簡素化を実現するために、世界の先進企業 50,000 社以上、Fortune 500 企業の 90 パーセントが、ベリタスのソリューションを導入しています。ベリタスのエンタープライズ・データサービス・プラットフォームは、お客様のデータ活用を推進するため、データ保護の自動化とデータリカバリを実現して、ビジネスに不可欠なアプリケーションの可用性を確保し、複雑化するデータ規制対応に必要なインサイトを提供します。ベリタスのソリューションは信頼性とスケーラビリティに優れ、500 以上のデータソースと 50 のクラウドを含む 150 以上のストレージ環境に対応しています。

VERITAS

The truth in information.

ベリタステクノロジーズ合同会社

https://www.veritas.com/ja/jp

〒107-0052 東京都港区赤坂 1-11-44 赤坂インターシティ 4F

ベリタスセールスインフォメーションセンター（法人のお客様向け製品購入に関する相談窓口）

■電話受付時間：10:00 ~ 12:00, 13:00 ~ 17:00 （土、日、祝日、年末年始を除く）

■電話番号：0120-907-000 （IP 電話からは 03-4531-1799）