# HP パフォーマンス・エンジニアリング・ベスト・プラクティス・シリーズ

パフォーマンス・エンジニアおよびパフォーマンス・マネージャ向け

# パフォーマンス監視ベスト・プラクティス

ドキュメントリリース日：2014 年 12 月（英語版）

# ご注意

## 保証

HP 製品，またはサービスの保証は，当該製品，およびサービスに付随する明示的な保証文によってのみ規定されるものとします。ここでの記載で追加保証を意図するものは一切ありません。ここに含まれる技術的，編集上の誤り，または欠如について，HP はいかなる責任も負いません。

ここに記載する情報は，予告なしに変更されることがあります。

## 権利の制限

機密性のあるコンピュータ・ソフトウェアです。これらを所有，使用，または複製するには，HP からの有効な使用許諾が必要です。商用コンピュータ・ソフトウェア，コンピュータ・ソフトウェアに関する文書類，および商用アイテムの技術データは，FAR12.211 および 12.212 の規定に従い，ベンダーの標準商用ライセンスに基づいて米国政府に使用許諾が付与されます。

## 著作権について



## 商標について

Adobe® は，Adobe Systems Incorporated の商標です。

Java™ は，Sun Microsystems, Inc. の米国商標です。

Microsoft®，Windows® は，Microsoft Corporation の米国登録商標です。

# ドキュメントの更新情報

このガイドの表紙には，次の識別情報が記載されています。

- ソフトウェアのバージョン番号は，ソフトウェアのバージョンを示します。
- ドキュメント・リリース日は，ドキュメントが更新されるたびに変更されます。
- ソフトウェア・リリース日は，このバージョンのソフトウェアのリリース期日を表します。

最新の更新のチェック，またはご使用のドキュメントが最新版かどうかのご確認には，次のサイトをご利用ください。

**http://support.openview.hp.com/selfsolve/manuals**

このサイトを利用するには，HP Passport への登録とサインインが必要です。HP Passport ID の取得登録は，次の Web サイトから行なうことができます。

**http://h20229.www2.hp.com/passport-registration.html**（英語サイト）

または，HP Passport のログイン・ページの **[New users - please register]** リンクをクリックします。

適切な製品サポート・サービスをお申し込みいただいたお客様は，最新版をご入手いただけます。詳細は，HP の営業担当にお問い合わせください。

# サポート

次の HP ソフトウェア・サポート Web サイトを参照してください。

**http://support.openview.hp.com**

HP ソフトウェアが提供する製品，サービス，サポートに関する詳細情報をご覧いただけます。

HP ソフトウェア・サポート・オンラインでは，セルフ・ソルブ機能を提供しています。お客様の業務の管理に必要な対話型の技術支援ツールに素早く効率的にアクセスいただけます。HP ソフトウェア・サポート Web サイトのサポート範囲は次のとおりです。

- 関心のある技術情報の検索
- サポート・ケースとエンハンスメント要求の登録とトラッキング
- ソフトウェア・パッチのダウンロード
- サポート契約の管理
- HP サポート窓口の検索
- 利用可能なサービスに関する情報の閲覧
- 他のソフトウェア・カスタマとの意見交換
- ソフトウェア・トレーニングの検索と登録

一部を除き，サポートのご利用には，HP Passport ユーザーとしてご登録の上，ログインしていただく必要があります。また，多くのサポートのご利用には，サポート契約が必要です。HP Passport ID を登録するには，以下の Web サイトにアクセスしてください。

**http://h20229.www2.hp.com/passport-registration.html**（英語サイト）

アクセス・レベルに関する詳細は，以下の Web サイトを参照してください。

**http://support.openview.hp.com/access_level.jsp**

# 目次

目次

# はじめに

パフォーマンス監視ベストプラクティスへようこそ。

本書では，パフォーマンス・テスト監視をさまざまな環境で実装する際の概念，ガイドライン，ベストプラクティスを紹介します。

**本章の内容**

# HP パフォーマンス監視について

HP は，自動パフォーマンス・テストにおいて業界をリードしています。パフォーマンス・テストとは，アプリケーション，アップグレード，パッチのデプロイメントに伴うリスクを軽減するために製品，人的リソース，プロセスを活用するための機能です。特に，運用環境の作業負荷を事前にデプロイしたシステムに適用し，システム・パフォーマンスとエンドユーザの作業環境を測定することを主な目的とします。パフォーマンス・テストを綿密に計画することによって，次のような点を解明できます。

- アプリケーションの応答時間は，対象ユーザが期待するレベルを満たしているか。
- ユーザのパフォーマンス・テストにアプリケーションが対応でき，想定以上の結果を達成できるか。
- アプリケーションは，業務で求められるトランザクション数の処理に対応できるか。
- アプリケーションは，ユーザが想定または想定しないパフォーマンス・テストで安定稼働できるか。
- 運用開始日からユーザが満足するパフォーマンスを発揮できるか。

上記のようなチェックを行うために，自動パフォーマンス・テストでは業務上の条件の変化が及ぼす影響を数値化し，デプロイメントのリスクを明確に把握します。自動パフォーマンス・テストを効果的に行うことによって，豊富なデータに基づいてリリースの可否を判断し，システム・ダウンタイムや障害を回避することができます。

HP は，自動パフォーマンス・テストの分野で HP LoadRunner と HP Performance Center という 2 つの製品を提供しています。この 2 つは異なる市場部門を対象にした製品ですが，いずれも実証済みのプロトコルや監視機能などを基盤としています。

**HP LoadRunner** は，ユーザの設定に基づいて，ピーク時の条件下でパフォーマンス・テストを実施します。LoadRunner によるパフォーマンス・テストでは，多数の仮想ユーザ（Vuser）をネットワーク上に分散します。仮想ユーザは UNIX プラットフォームと Windows プラットフォームに対応しています。仮想ユーザを使用することによって，最小限のハードウェア・リソースで実際のユーザ環境をエミュレートでき，一貫性があり，再現可能，測定可能な方法でテスト対象アプリケーション（AUT）を実行することができます。

**HP Performance Center** は Application Lifecycle Management（ALM）を構成する製品であり，ユーザ組織全体をカバーするクロス・エンタープライズ対応のパフォーマンス・テスト・ツールとして，ユーザ組織のインフラストラクチャにインストールされます。

- 地理的に離れた場所でパフォーマンス・テストを実行したい場合でも，Performance Center を使用すれば，複数のパフォーマンス・テスト・プロジェクトを同時に実行でき，テスト担当者が実際に足を運ぶ必要はありません。
- Performance Center は，パフォーマンスのテストに対する内部のニーズをすべて管理します。
- Performance Center には，リソースの割り当てやスケジュール設定など，大規模なパフォーマンス・テスト・プロジェクトで必要になる機能がすべて含まれており，Web 経由での集中管理が可能です。
- Performance Center は，テスト・プロセスの合理化，リソース・コストの削減，操作効率の向上に役立ちます。
- Performance Center により，パフォーマンス・ボトルネックを特定できます。
- Performance Center により，テスト対象アプリケーションが対応可能なユーザ数を特定できます（このユーザ数は，アプリケーションのパフォーマンスが低下し始める**分岐点**を示します）。この情報を元に，アプリケーションのテスト結果を向上するために必要な方策を検討できます。

本書では，パフォーマンス監視チームのニーズ対応と，監視機能の設定およびデプロイメントにかかる時間の短縮を目的に，パフォーマンス・テスト・ガイドラインを掲載しています。また，監視機能の事前設定として，標準設定のメトリックス，しきい値（該当する場合），プロアクティブ・テスト（該当する場合）も掲載しています。このようなガイドラインや設定値はすべて，HP のオペレーティング・システム管理者，HP のプロフェッショナル・サービス組織，技術文書，業界の専門家による書籍などから収集したベストプラクティス・データと専門知識に基づく調査をベースにしたものです。このようなガイドラインを参考にシステム・パフォーマンスを監視することにより，システム障害につながるパフォーマンス・ボトルネックを特定することができます。

本書の目的は，Performance Center の知識があまりないユーザや IT 組織にも，使いやすく包括的なパフォーマンス監視ガイドラインを提供することにあります。

# 本書の構成

『HP パフォーマンス監視ベスト・プラクティス』は，次の項で構成されています。

**第 I 部　概要**

パフォーマンス監視とソリューションの概要

**第 II 部　オペレーティング・システム**

Window および UNIX オペレーティング・システムの監視に関するベスト・プラクティス

**第 III 部　ランタイム・プラットフォーム**

Java および .NET ランタイム・プラットフォームの監視に関するベスト・プラクティス

**第 IV 部　Web サーバの監視**

Apache および IIS Web サーバの監視に関するベスト・プラクティス

**第 V 部　アプリケーション・サーバの監視**

WebLogic および WebSphere アプリケーション・サーバの監視に関するベスト・プラクティス

**第 VI 部　データベース・リソースの監視**

Oracle および MS SQL Server データベース・リソースの監視に関するベスト・プラクティス

**第 VII 部　仮想化テクノロジ**

Microsoft Hyper-V および VMWare ハイパーバイザ・プラットフォームの監視に関するベスト・プラクティス

## 対象読者

本書は，次のユーザを対象としています。

- パフォーマンス・エンジニア
- パフォーマンス CoE マネージャ
- QA マネージャ
- QA エンジニア

## その他のオンライン・リソース

**［トラブルシューティング & ナレッジ ベース］**から，セルフ・ソルブ技術情報を検索できる HP ソフトウェア・サポート Web サイトのトラブルシューティング・ページにアクセスできます。［**ヘルプ**］＞［**トラブルシューティング & ナレッジ ベース**］を選択します。この Web サイトの URL は http://support.openview.hp.com/troubleshooting.jsp です。

**HP ソフトウェア・サポート**：HP ソフトウェアのサポート Web サイトにアクセスします。このサイトでは，セルフ・ソルブ技術情報を閲覧できます。また，ユーザ・ディスカッション・フォーラムへの投稿や検索，サポート依頼の送信，パッチや更新されたドキュメントのダウンロードなども行えます。［**ヘルプ**］＞［**HP Software サポート**］を選択します。この Web サイトの URL は，http://support.openview.hp.com です。

ほとんどのサポート・ページでは，HP Passport ユーザとして登録してログインすることを求められます。また，多くはサポート契約が必要です。

アクセス・レベルの詳細については，

http://support.openview.hp.com/access_level.jsp

HP Passport ユーザ ID の登録は，次の場所で行います。

http://h20229.www2.hp.com/passport-registration.html（英語サイト）

**HP ソフトウェア Web サイト**から，HP ソフトウェア Web サイトにアクセスします。このサイトでは，HP ソフトウェア製品に関する最新情報を提供します。新しいソフトウェアのリリース，セミナー，展示会，カスタマー・サポートなどの情報も含まれています。［**ヘルプ**］＞［**HP ホームページ**］を選択します。この Web サイトの URL は，http://welcome.hp.com/country/jp/ja/prodserv/software.html です。

# 第I部

## 概要

# 第1章

# パフォーマンス監視について

パフォーマンスの監視は，テスト対象アプリケーションのパフォーマンスを測定するパフォーマンス・テスト機能の 1 つです。

さらにパフォーマンス監視は，システムの品質を示す属性（スケーラビリティ，信頼性，リソース使用率など）の検証でも役立ちます。

**本章の内容**

# パフォーマンス監視の概要

パフォーマンス監視では，パフォーマンス・テストでアプリケーションを実行する方法について最新の情報を提供します。複数のパフォーマンス・テストを実行し，そのデータを分析することによってベースラインを定義することができます。ベースラインとは，一般的な稼働条件下で許容できるレベルのパフォーマンスを示す測定値のセットを示します。問題が発生した時に，ベースラインを参照ポイントとして使用することによって，問題を簡単に特定できます。

また，システム障害のトラブルシューティングでは，パフォーマンス・データから，問題発生時にシステム・リソースがどのように動作していたかがわかるので，原因特定に役立ちます。

さらには，アプリケーションのパフォーマンス監視データを元に，将来的な拡張を予測し，システム構成の変更が及ぼす影響への対処を計画することができます。

パフォーマンス監視では，さまざまな作業負荷条件下（パフォーマンス・テスト，作業負荷，単独ユーザ操作など）でのアプリケーションの動作を示すメトリックスを収集することによって，ボトルネックを特定し，パフォーマンス目標を達成しているかどうかを検証できます。さらにこのメトリックスは，パフォーマンス目標として設定されているメトリックスと比較することができます。このようなメトリックスの例として，応答時間，スループット，リソース使用率（CPU，メモリ，ディスク I/O，ネットワーク帯域幅など）があります。ただし，メトリックスの特徴をよく理解しておかないと，パフォーマンス・データの分析から正しい結論を引き出したり，ボトルネックを正しく特定することはできません。したがって，正しい分析に必要な専門知識を身につけることをお勧めします。

アプリケーション開発者と IT 組織は，アプリケーションのパフォーマンスを最適化するための設定とチューニングという課題に常に直面しています。アプリケーションの実行速度が遅い**原因**を究明する作業には，技能と技術を融合したスキルが必要です。ただし，どのようなレベルの技能や技術があったとしても，問題の診断と解決を行うためにはまず最初にパフォーマンス・データを収集しなければなりません。

# パフォーマンスに関する用語

監視フェーズでは，パフォーマンス・テストの数値データが収集されます。ここでは，パフォーマンス監視で使用される重要な用語について説明します。

システムの動作を示す測定値の中でも，最も重要なのが**帯域幅**と**スループット**です。帯域幅は処理を実行「可能な」速度を示す測定値であるのに対して，スループットは「実際の」処理速度を示します。

スループットは，テスト対象システムで稼働するユーザ数によって変動し，通常は 1 秒あたりの要求の数で表されます。たとえば，同時ユーザ数が多くなるとスループットが低下するケースや，作業負荷が大きくても一定のスループットを発揮するが，キュー内の要求が増えるとスループットが低下するケースなどがあります。**使用率**は，各種システム・リソースがビジー状態になる割合を示します。

特定のタスクの実行にかかる時間を示す測定値には，**キュー時間**，**サービス時間**，**応答時間**があります。

## サービス時間とキュー時間

**サービス時間**は，特定のユーザ作業要求の処理にかかる時間を示します。

作業要求を受け取ったリソースがビジー状態であると，すぐには処理を開始できないため，要求はキューに入ります。この要求の処理が開始されるまでキュー内で待機する時間は，**キュー時間**の遅延として測定されます。

## 応答時間

**応答時間**は最も重要なメトリックスであり，本書ではサービス時間とキュー時間の合計を指します。応答時間は，次に示すように，サーバでの応答時間とクライアントでの応答時間に分けることができます。

- **サーバで測定される待ち時間**: サーバが要求の実行を完了するまでにかかる時間です。要求がネットワーク経由で応答する際の，クライアント/サーバ間の遅延時間は含まれていません。
- **クライアントで測定される待ち時間**：クライアント側で測定された待ち時間であり，要求のキュー時間，サーバが要求の処理を完了するまでにかかる時間，ネットワーク遅延時間が含まれます。アクティビティとそのニーズを適切に組み合わせるためには，アプリケーションの使用率を詳細に把握する必要があります。

## 作業負荷プロファイル，容量，スケーラビリティ

**作業負荷プロファイル**も，パフォーマンス監視の結果に影響を与える要素の 1 つです。これは，テスト対象アプリケーションでさまざまな操作を行うユーザを組み合わせたユーザ・ミックスです。

**容量**は，使用率が最大の状態で各リソースが処理可能な作業量を示すのに対して，**スケーラビリティ**は，マシンまたはシステムのスループットを，サービスを要求するユーザ数の合計で示します。

# パフォーマンスに影響を与える要因

ソフトウェア開発技術は着実に向上していますが，100% 完璧なソフトウェアの開発は不可能だということはよく知られています。そしてアプリケーションのパフォーマンスは，パフォーマンス目標との比較で把握するしかありません。

パフォーマンスの問題は，クライアント/サーバや Web アプリケーションといったシステムのタイプに関わらず，あらゆるタイプのシステムに影響を与えます。したがって，システム・パフォーマンスにどのような要因が影響を与えるのかを理解してから，パフォーマンスの問題解決にあたる必要があります。

一般的に，パフォーマンスに影響を与える要因は，**プロジェクト管理**に起因する要因と**技術的な**要因に分類されます。

## パフォーマンスに影響を与えるプロジェクト管理要因

最近のソフトウェア開発ライフサイクル（SDLC）では，ますます激化する競争を背景に，主要なフェーズが時間的な制約を受けています。これにより，プロジェクト管理では次のような問題が発生しています。

- コーディングにかけられる時間が短縮した結果，パフォーマンスがあまり重視されなくなり，品質低下につながっています。
- このように短期間で完成させるアプローチでは，十分な情報が収集されず，パフォーマンス目標を達成できない可能性があります
- 製品のデプロイメントの後に内部設計の矛盾が発見されることがあります（不要なオブジェクトや画面操作）。
- コーディング規則の違反が多くなり，コードが最適化されないため，リソースが無駄に消費されます。
- プロジェクト専用の設計になるので，将来的なプロジェクトにモジュールを再利用できなくなります。
- スケーラビリティを考慮したモジュール設計が行われません。
- ユーザのパフォーマンス・テストが急増すると，システムがダウンする可能性があります。

## パフォーマンスに影響を与える技術的な要因

プロジェクト管理に起因する問題がアプリケーションの出力に大きな影響を与えるのに対して，技術的な問題はアプリケーション全体のパフォーマンスに重大な影響を与えます。このような問題は，誤ったテクノロジ・プラットフォームを選択したことに起因することがあります。テクノロジ・プラットフォームの中には一部の用途に特化した設計されているものがあるため，それ以外の条件下では十分なパフォーマンスを発揮できない可能性があるからです。

ただしほとんどの問題は，パフォーマンスを考慮しない方法で開発を行うことが原因で発生します。多くの開発者は，開発フェーズでコードの最適化を行っていません。このようなコードは，**メモリ**や**プロセッサ**など貴重なシステム・リソースを無駄に使用している可能性があります。また，次のような深刻なパフォーマンス・ボトルネックを引き起こすことがあります。

- メモリ・リーク
- 配列の範囲エラー
- 非効率なバッファ処理
- 処理サイクルの数が過剰
- HTTP トランザクションが多数発生
- メモリとディスク間のファイル転送が過剰
- 非効率なセッション・ステート管理
- 同時ユーザの増大によるスレッド競合
- ピーク・パフォーマンス・テストでのアーキテクチャ・サイジングが不十分
- 非効率な SQL ステートメント
- データベース・テーブルのインデックスが不適切
- サーバ構成が不適切

コードがパッケージ化されてしまうと，このような問題の追跡は難しくなり，特殊なツールや手法が必要になります。

また，**セキュリティ**もパフォーマンスに影響を与える技術的要因の 1 つです。セキュリティ機能（SSL，秘密 / 公開キーなど）を追加すると負荷が大きくなることから，一般的に，アプリケーションのパフォーマンスとセキュリティにはトレードオフの関係があります。

**ネットワーク**関連の問題，特に Web アプリケーションについても考慮する必要があります。ネットワーク関連の問題は，次のような要因から発生します。

- ネットワーク・インフラストラクチャが古いまたは最適化されていない
- Web サイト接続が遅いため，ネットワーク・トラフィックや応答時間に影響する
- バランスよく考えられたパフォーマンス・テストをサーバで実行しなかったためパフォーマンスが低下する

## パフォーマンス目標

パフォーマンス・テスト対象のシステムの監視を成功させるためには，パフォーマンス・プロジェクトの内容に合ったアプローチと機能を使ってパフォーマンスを監視する必要があります。この目的のために，パフォーマンス・テスト・ライフサイクル（PTLC）の最初に実行する手順が，**パフォーマンス目標**の設定です。この作業では，パフォーマンス・テスト・プロセスで収集したデータと，製品の品質を判断または改善する際に使用する期待値を使用します。ただし，パフォーマンス目標は必ずしも数値である必要はなく，他のパフォーマンス基準に直接関連付ける必要もありません。

パフォーマンス目標には，次のような特徴があります。

- **契約による合意**：一般的なパフォーマンス目標は，ビジネス・ユーザとテストを担当するエンティティとの間で正式に合意されます。
  - **必須条件**：法規制，サービス・レベル契約，確定されているビジネス・ニーズなど，交渉の余地のない絶対的な条件です。
  - **交渉可能な条件**：製品リリースには望ましいが，状況によっては変更可能な条件です。エンド・ユーザ中心の条件であるのが一般的です。

- **精度**：パフォーマンス目標の数値を言葉で表現します。
  - **限定値**：目標値を限定された値で示します。たとえば，「CPU 使用率 50%」などがあります。
  - **概数**：目標値を範囲または単一の制限値で示します。たとえば，「プロセスあたりのメモリ使用率が 50MB を超えない」，「トラザクション X の 90% 以上の応答時間が 3 秒以下である」などがあります。
- **境界**：多くの場合，パフォーマンス目標は，テスト対象アプリケーションに関連する値で設定されます。
  - **ターゲット値**：一連の条件下で，リソースに期待される値です。応答時間，スループット，リソース使用率などで表すのが一般的です。
  - **しきい値**：リソースで許容される境界値を示します。応答時間，スループット（1 秒あたりのトランザクション数），リソース使用率などで表すのが一般的です。

パフォーマンス目標とサービス属性は，ビジネス要件を元に定義されます。測定によって収集された監視対象メトリックスは，パフォーマンス目標を満たしているか，どの程度乖離しているかを示します。

## パフォーマンス監視ガイドライン

ここでは，パフォーマンスを監視する上で参考になる一般的なガイドラインを紹介します。

- 標準的なサンプリング間隔から監視を始めます。具体的な問題が発生している場合や，疑いのあるボトルネックを特定できる場合には，サンプリング間隔を短くします。
- サンプリング間隔に基づいて，監視セッションの長さを決定します。短い間隔でサンプリングを行う場合には，監視セッションを短くしてください。
- 収集データを管理可能な量に抑えるためにも，監視オブジェクトの数とサンプリング頻度のバランスを考慮してください。

- テスト対象アプリケーションの特性に関連した監視カウンタのみを選択します。これにより，テスト・シナリオを包括的にカバーできるだけでなく，類似した監視を重複してデプロイしてしまうことがなくなります。
- カウンタの数が多すぎると，分析処理に過度な負荷がかかるだけでなく，パフォーマンス・オーバーヘッドを招きます。
- システム構成（仮想メモリのサイズなど）が正しいかどうかのチェックを必ず行います。このチェックは監視手順には含まれていませんが，テスト結果に大きな影響を及ぼすことがあります。
- リモート・マシン向けのポリシーを決定します。方法としては，各リモート・マシンで監視サービスを定期的に実行して結果を収集し，完了後に結果をまとめて管理者に送信する方法と，メトリックスを継続的に収集し，ネットワーク経由で管理者に送信する方法のいずれかになります。テスト対象アプリケーションとパフォーマンス目標に基づいて，ポリシーを選択します。
- ハードウェアやオペレーティング・システムのベンダが「汎用的な」推奨値（平均 CPU 使用率は 80% を超えない，ディスク・キューの長さは 2 未満とする，など）を提示している場合は，これを考慮してテストやアプリケーションのしきい値を設定します。

  「汎用的な」推奨値は，必ず満たさなければならない条件ではなく，監視結果とパフォーマンス・テストの応答時間の比較検討の対象となるメトリックスとして活用してください。
- アプリケーションのアクティビティとその目標を最も明確に監視できるパラメータを選択します。データ量が多すぎると，分析プロセスに大きな負荷がかかります。
- 監視目標を達成するには，システムやアプリケーションに組み込まれたオブジェクトやカウンタのみでなく，アプリケーション固有のログ，スクリプト，XML ファイルなどを監視する必要があります。
- 基本的な監視機能（HP SiteScope など）を，数を限定して継続的に実行する一方で，これよりも詳細な監視をパフォーマンス・テスト・シナリオ向けに定義し，テスト中に実行する方法をお勧めします。

パフォーマンス・テスト中のみでなく，パフォーマンス・テストの前後にもメトリックスを測定して「ローカル・ベースライン」を作成します。これにより，パフォーマンス・テストの完了後，テスト対象アプリケーションがベースラインに戻ることを確認できます。

## 誤った考え方

パフォーマンス監視全体の目的は分析用の測定データを収集することであり，最終的な目的はボトルネックの原因を特定することにあります。

ただし，次に示すような誤解があると，それが原因で目標を達成できなくなるケースや，負担やコストが増大してしまうケースがあるので注意してください。

- **基本的なインフラストラクチャを監視すれば十分である。**
  システム・メトリックス（CPU，メモリ，ディスク）の監視は重要ですが，それだけでは，ユーザやアプリケーションが稼働する実際の環境でパフォーマンスに問題が発生しているかどうかを把握できません。現在発生しているパフォーマンスの問題の多くは，個々のハードウェア・コンポーネントではなく，アプリケーション・コンポーネントに起因しています。したがって，アプリケーション・パフォーマンスの全体像を正確に把握するためには，システムの監視だけでは不十分です。

- **アプリケーションのプロセスやサービスを監視すれば十分である。**
  パッケージ・アプリケーション，J2EE，.NET，カスタマイズされた SOA アプリケーションなど，今日のアプリケーションは複雑で，複数のシステムやさまざまなテクノロジがベースになっています。アプリケーションの稼働状態を完全に把握するには，コンポーネントを詳細に監視および診断することによって，サービス間で行われる複雑な対話を理解する必要があります。HP Diagnostics は，エンド・ユーザ・ビジネス・プロセスから診断を開始し，アプリケーション・コンポーネントやシステム・レイヤにドリルダウンすることによって，ビジネスに大きな影響を与える問題の短期解決や，サービス・レベル契約の達成において威力を発揮します。

➤ **システムやアプリケーションで使用可能なメトリックスをすべて監視するのが最善策。**

収集データの量が多すぎると分析作業で大きな負荷が発生します。また，パフォーマンスにも影響が発生するので，本当の問題を切り分けることができなくなる可能性があります。つまり，すべてのメトリックスを完全にカバーする必要はなく，これは理想的な監視方法でもありません。80 対 20 の法則（20% のシステム・コンポーネントやアプリケーション・コンポーネントが障害の 80% に起因するという一般法則）は，パフォーマンス監視にもあてはまります。したがって，ビジネス・クリティカルな機能に関連するシステムとそれ以外のシステムを区別することが重要です。

➤ **同じメトリックスですべてのテストを実行できる。**

ほとんどのパフォーマンス・テストで共通のメトリックスもありますが，効果的なパフォーマンス監視を実行するためには，実行するテストのタイプに基づいてさメトリックスを組み合わせる必要があります。

➤ **一般的な環境では，Web サーバを監視すれば十分である。**

最近のアプリケーションは複雑なので，アーキテクチャを理解していないとパフォーマンスの実態を把握することはできません。Web アプリケーションの標準的なデプロイメントには，Web サーバ，アプリケーション・サーバ，データベース・サーバが含まれ，ほとんどの場合，それぞれが複数の物理マシンや物理サイトに分散しています。さらに，SOA の普及に伴って，エンド・ユーザとの対話に関わるインフラストラクチャやサービスが増大しています。したがって，関連するサーバ，特にデータベース・サーバをすべて監視することが非常に重要です。また，クライアント・ワークステーションの監視も必要になることがあります。

# ボトルネックとチューニング

アプリケーションがパフォーマンス目標を達成するためには，パフォーマンスを継続的に監視する必要があります。監視によって収集したパフォーマンス・データは，運用環境に近い条件下でパフォーマンスの問題を診断するときに役立ちます。このデータから，ボトルネックの存在がわかることがあります。ボトルネックとは，1 つのコンポーネントによってシステム全体のパフォーマンスや容量が極端に制限される状態を指します。

ボトルネックの正式な定義は，システムのクリティカル・パス上に存在し，スループットが最低になるポイントです。クライアント・サーバ・システム，特に Web ベース・システムでは，CPU，メモリ，データベース，ネットワーク・リンクなど，低速になる可能性のあるポイントが多数存在します。オペレーティング・システムに関連するカウンタを監視することで特定できるボトルネックもあれば，アプリケーションをインストルメント化しないと特定できないものもあります。

HP が提供する HP Diagnostics for J2EE/.Net には，次のような機能があります。

- 運用環境の問題をプロアクティブに検出します。
- システム層やアプリケーション層ごとの切り分けを短時間で実行します。
- 根本原因となっているアプリケーション・コンポーネントを特定します。

開発環境や QA 環境では問題がなくても，運用環境へ移行するとスケーラビリティやパフォーマンスに問題が発生することがあります。したがって，アプリケーションが稼働するインフラストラクチャがどのような影響を及ぼすか，パフォーマンス・テストの実行中に多くのアプリケーション・コンポーネントがどのように動作するかを理解することが必要です。システム診断の点から考えれば，アプリケーション・アーキテクチャの層とアプリケーション・コンポーネントごとに問題を切り分ける機能と，J2EE/.Net のパフォーマンスの問題，J2EE/.NET 環境，実際のロジックにまで段階的にドリルダウンすることで根本原因を究明する機能が重要な役割を果たします。

一方でビジネスの視点から考えれば，システム・リソースをフル活用することが目標になります。CPU，大量のメモリ，ディスク容量には購入コストがかかっているので，最大限に活用する必要があるからです。したがって，ボトルネックの別の定義として，リソースがフル活用された状態であり，なおかつ待機中のプロセス/スレッドが存在する状態を指すこともできます。

分散環境は，次のような理由から特にボトルネックの影響を大きく受けてしまいます。

- アプリケーション・コンポーネントごとに複数のオペレーティング・システムが稼働している。
- ネットワーク構成がコンポーネントごとに異なる。
- ファイアウォールなどのセキュリティ機能が実装されている。
- データベースの誤動作。スキーマ設計，インデックス，ストレージのパーティション設定の不備が原因でデータベースが誤動作すると，システム全体の応答時間が大幅に低下します。
- 非効率なスレッド管理。これにより，同時実行可能な処理数が低下します。
- 未検証の接続数が増大する。

- スレッドのプール・サイズ管理が非効率。これが原因で，スレッド数が短時間で増大します。
- データベース接続のプール・サイズの設定に誤りがある。
- 最適化されていない SQL ステートメントが頻繁に使用される。
- 大量のトランザクションを処理するためにはメモリ･チューニングが必要になるにも関わらず，物理メモリや共有メモリのチューニングが行われていない。

これまでに説明したように，パフォーマンス監視の結果を元に，ボトルネックの検出と解消，アプリケーションのチューニングを行うことが理想的です。

80 対 20 の法則が当てはまる別のケースとして，どのようなアプリケーションでも 20% の操作がリソースの 80% を消費しているという事実があります。つまり，よく実行される操作ほどボトルネックになる可能性も高くなります。したがって，20% にあたるこのコードの品質を向上すれば，全体的なパフォーマンスを大幅に改善できます。

パフォーマンスのチューニングは，技能と技術を融合したスキルであり，設計フェーズ，コンパイル時，アセンブリ・レベル，実行時に行われます。実行時間，メモリ使用率，ディスク容量，帯域幅，消費電力などのリソースにはトレードオフの関係があるものも含まれるので，同時に最適化できるのは 1 つか 2 つに限られます。たとえば，キャッシングを拡張（要求の実行時間）すると，メモリ使用量が増大したり，マルチプロセッサ環境ではソース・コードが複雑になるなどの影響が発生します。

# 第2章

# HP 監視ソリューション

HP は，あらゆる監視ニーズに対応できる豊富な監視ソリューション・ポートフォリオを提供しています。パフォーマンス検証製品である HP LoadRunner と HP Performance Center は，HP Sitescope および HP Diagnostics という 2 つのソリューションとの統合を通じて，パフォーマンスの監視とボトルネック分析を行う完全かつ包括的なソリューションを提供します。

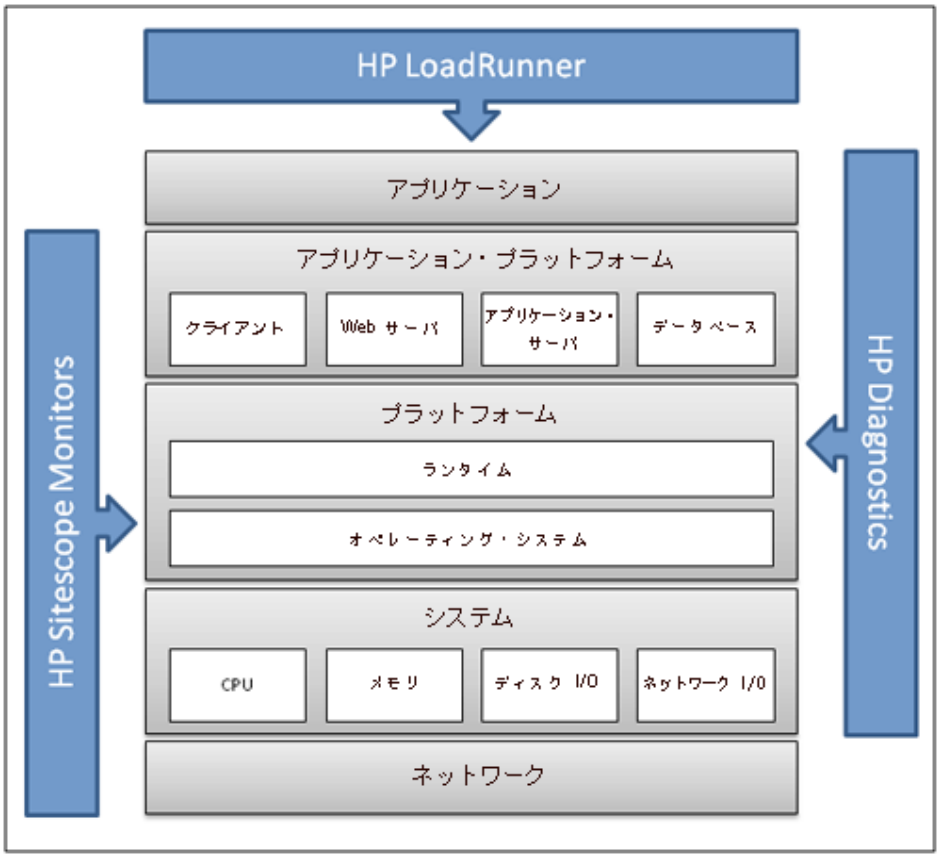
**本章の内容**

# 概要

LoadRunner と Performance Center は，HP Sitescope および HP Diagnostics と統合することによって，包括的で完全な監視ソリューションとなります。このソリューションでは，次に示す各製品の優れた機能を活用することができます。

- **LoadRunner および Performance Center**：一般的な作業負荷のシミュレーションやユーザ・アクションをトランザクションの観点から監視することにより，パフォーマンスを検証します。
- **HP Sitescope**：テスト対象システムの各層を監視し，ボトルネック分析に焦点をあてたデータを収集します。
- **HP Diagnostics**：トランザクション応答時間をアプリケーション層ごとに測定することでパフォーマンス・ボトルネックを特定し，ボトルネック解消の参考になるデータを提供します。

次の図は，テスト対象システムの各層で提供される HP 監視ソリューションを示しています。



具体的なアプローチとしては，監視のタイプに合わせてカウンタを選択する必要があります。メトリックスは次のタイプに分類できます。

- **アプリケーション**：アプリケーション・メトリックスには，カスタマイズ可能なパフォーマンス・カウンタが含まれます。
- **プラットフォーム**：Microsoft Windows および JVM 環境の J2EE.NET 共通言語ランタイム（CLR）に関連するメトリックスです。オペレーティング・システムもプラットフォームとみなされます。
- **システム**：プロセス，メモリ，ディスク I/O，ネットワーク I/O に関連するメトリックスです。
- **ネットワーク**：ネットワーク帯域幅の使用率や遅延に関連するメトリックスです。

検証を目的とするテストでは，LoadRunner と Sitescope を使用して AUT を監視することで，トランザクション応答時間やリソース使用率で発生する可能性のあるボトルネックを特定することをお勧めします。ボトルネックが検出された場合には，HP Diagnostics を使って狭い範囲で集中テストを行い，問題を切り分けることによってボトルネック解消の参考になるデータを開発チームに提供します。

最適化を目的とするテストでは，最初から HP Diagnostics を使用することをお勧めします。これにより，最適化の可能性のあるポイントを短期間で特定することができます。このアプローチは，負荷テスト，アプリケーションの小規模サブシステムに対するテスト実行，ボリューム・テストなどに最適です。

# HP LoadRunner

LoadRunner と Performance Center にはネイティブの監視機能が付属しているので，パフォーマンス・テストをすぐに実行できます。

次のような機能があります。

- **LoadRunner データ・ポイント監視**：Web ベース・アプリケーションについて，VuGen スクリプトが生成するトランザクション監視機能や，データ・ポイント（1 秒あたりのヒット数やスループットなど）の自動生成機能を実行します。
- **テスト対象システムの監視**：システム・リソースなどアプリケーション関連のメトリックス，Web サーバ，データベース，ネットワーク・メトリックスが含まれます。

LoadRunner のトランザクション監視は基本的な監視機能ですが，ユーザの操作環境を包括的に示しているため，パフォーマンス・テストでは重要な役割を果たす機能です。ビジネスの視点からトランザクションを検証できるので，テストやボトルネック分析の対象を絞り込むことができます。LoadRunner のサービス・レベル・アグリーメント機能を使って，実際のパフォーマンスを測定し，パフォーマンス目標と比較することをお勧めします。次の図は LoadRunner スクリプトの例です。Web リンクのマウス・クリックを測定するトランザクションが示されています。

```
ManageResourcePool()
{
    web_reg_find("TEXT=Select a Resource Pool",LAST);
    lr_start_transaction("TM_ResourcePool_T01_ClickManageResourcePools");
    web_link("Manage Resource Pools",
        "Text=Manage Resource Pools",
        "Snapshot=t4.inf",
        LAST);
    lr_end_transaction("TM_ResourcePool_T01_ClickManageResourcePools", LR_AUTO);
    lr_think_time( 10 );
    web_reg_find("TEXT=Resource Requests",LAST);
    lr_start_transaction("TM_ResourcePool_T02_ClickOnResourcePool");
    web_link("ResPool_{UserNum}",
        "Text=ResPool_{UserNum}",
        "Snapshot=t5.inf",
        LAST);
    lr_end_transaction("TM_ResourcePool_T02_ClickOnResourcePool", LR_AUTO);
    .
    .
    .
```

## トランザクション・カウンタ

トランザクション・カウンタはすべて，個々のトランザクションの数値と集計値（合計値）が示されます。

| カウンタ | 説明 |
|---|---|
| トランザクション応答時間 | 作業負荷の変化に伴って，応答時間も変動します。<br>平均応答時間，最大値，パーセンタイルなど。 |
| 秒ごとのトランザクション | 1 秒あたりに生成されたトランザクションの数です。 |
| トランザクションの成功率 | 成功，失敗，停止したトランザクションの数です。 |

## Web リソース関連のカウンタ

LoadRunner には，データ・ポイント・ベースの監視機能として Web ベース・アプリケーションに関連したカウンタがあります。これは，シミュレートした作業負荷にアプリケーションが対処できるかどうかを評価する上で，重要なカウンタです。

- 秒ごとのヒット数
- スループット
- 秒毎の HTTP 応答数
- 秒ごとにダウンロードされたページ数
- 接続数
- 秒ごとの SSL 接続

LoadRunner には，VuGen スクリプトからユーザ定義のデータ・ポイントを生成する機能があります。これは，ユーザ環境専用にカスタマイズした監視機能を短時間で作成できる非常に強力なツールです。操作には VuGen の **lr_user_data_point** 関数を使用します。各種データ・ソースから取得したメトリックス値は，LoadRunner Controller または Performance Center オンライン・グラフに表示するだけでなく，LoadRunner Analysis でオフライン調査や他の測定データとの比較にも使用できます。

次の図は，JBoss カスタム・モニタを示しています。この VuGen スクリプトは，JBoss パフォーマンス統計ページのデータを相関処理し，結果を Controller または Performance Center 実行ページのユーザ定義データ・ポイント・グラフで表示します。


さらに LoadRunner と Performance Center では，製品に組み込まれているネイティブ監視機能や SiteScope との統合を通じて，システム・リソース使用率，データベース，Web サーバ，アプリケーション・サーバなどの監視も実行できます。

# HP Sitescope

LoadRunner と Performance Center は SiteScope と連携することができます。SiteScope は業界トップの監視ソリューションであり, スタンドアロン製品としてだけでなく, HP Business Availability Center や本書でこれまで紹介した幅広いパフォーマンス・テスト・ソリューションと組み合わせて監視モジュールとしても使用できます。

SiteScope はエージェントを使用しない監視ソリューションであり, サーバ, オペレーティング・システム，ネットワーク・デバイス，アプリケーション，アプリケーション・コンポーネントを監視することによって，IT 分散環境での可用性とパフォーマンスを確保します。SiteScope は Web ベースのインフラストラクチャ監視ソリューションなので，軽量で自由にカスタマイズでき，運用環境システムにデータ収集エージェントをインストールする必要がありません。

SiteScope では，インフラストラクチャの稼働状態の検証に必要なリアルタイム情報の取得，問題の通知，潜在的なボトルネックの解消が可能です。また，SiteScope に付属するテンプレートは，監視機能の標準化や，監視機能のデプロイメントにかかる時間の短縮に役立ちます。さらに，警告タイプを設定することで，さまざまな方法でイベント情報を通知および記録できます。警告テンプレートは，ユーザ・ニーズに合わせたカスタマイズが可能です。

Performance Center のネイティブ監視機能は組織の平均的なニーズのほとんどに対応できますが，SiteScope には幅広い監視機能と事前にパッケージ化されたテンプレートが付属し，あらゆる監視ニーズに対応できます。SiteScope には，オペレーティング・システムの測定，アプリケーション・サーバのメトリックス，各種 UNIX フレーバやファイルの検査など，あらゆるニーズに対応できる機能が搭載されています。

SiteScope は, 業界初のエージェントレス監視ソリューションです。業界で実証されたエージェントレスな監視アーキテクチャを採用した SiteScope では，エージェントベースの監視ソリューションに比べて次の点で総運用コストを低減できます。

- インフラストラクチャ・コンポーネントに関する詳細なパフォーマンス・データを収集
- すべての監視コンポーネントを中央サーバに集約することにより，時間とメンテナンス・コストを低減
- エージェントの動作が不安定になることでシステム・パフォーマンスに影響を与える可能性を排除

# HP Diagnostics

HP Diagnostics は，アプリケーションで発生するパフォーマンスの問題を切り分け，アプリケーションのパフォーマンス・ボトルネックの平均復旧時間（MTTR）を短縮します。また，問題解決に役立つ情報を提供します。

HP Diagnostics は，LoadRunner および Performance Center との連携を通じて，J2EE，.NET，エンタープライズ・リソース・プランニング（ERP），カスタマ・リレーションシップ・マネジメント（CRM）などの複雑なアプリケーションのテストにも対応し，アプリケーション・ライフサイクル全体で威力を発揮します。

HP Diagnostics には，次のような機能があります。

- ライフサイクルの早い段階で問題点を検出して解決
- 運用環境へ移行する前にアプリケーションでよく発生する問題を検出し，品質を向上
- アプリケーションを運用環境に移行するかどうかの判断をサポートする具体的なデータを収集
- アプリケーションが運用環境へ移行した後，役割ベースの管理および監視を通じて，問題解決にかかる時間を短縮

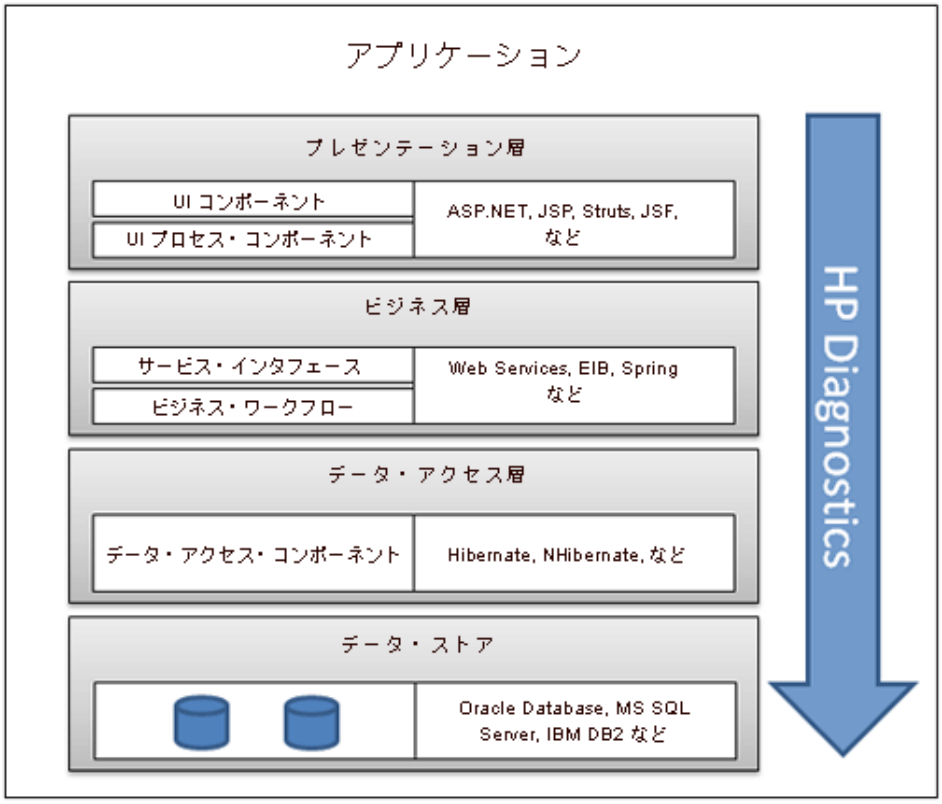
パフォーマンス・テストの実行中，HP Diagnostics は J2EE，.NET，ERP，CRM の各ビジネス・プロセスをインフラストラクチャのすべての階層を対象にクライアント側から追跡します。さらに，各トランザクションの応答時間を，階層ごとやコンポーネントごとに把握します。

HP Diagnostics には，次のような特長があります。

- 個々の階層，コンポーネント，メモリ，SQL ステートメントがビジネス・プロセスの全体的なパフォーマンスにどのような影響を与えているのかを，わかりやすく表示します。パフォーマンス・テストの実行中または完了後，アプリケーションのスケーラビリティの問題と参考データをアプリケーション・チームに通知します。
- ビジネスの視点から問題を効率的に選別して検出することで，ビジネス・プロセスに影響を与える問題を集中的に対処できます。

➤ テスト対象のビジネス・プロセスに関連するコンポーネントを簡単に特定できます。J2EE，ERP，CRM の各アプリケーションは無数のコンポーネントを使用している可能性があり，その中から特定のコンポーネントを見つけ出すのは難しい作業です。HP Diagnostics は，所定のトランザクションが実行されると「アクティブ」になるコンポーネントを自動的に検出し，分析用のデータを収集します。ビジネス・プロセスで使用されないコンポーネントは除外されるので，システム構成を調べなくてもすぐに分析作業に取りかかることができます。

次の図は，HP Diagnostics によってインストルメント化されるアプリケーション層の例です。



HP Diagnostics の主な機能と利点：

- 低速なエンド・ユーザ・トランザクションから，ボトルネックが発生しているコンポーネント，メソッド，SQL ステートメントにドリルダウンする機能は，メモリや例外などよく発生する問題の解決で威力を発揮します。
- ビジネス・プロセスに関与するコンポーネントをすべて自動的に検出し，追跡します。
- アプリケーション・ライフサイクル全体でアプリケーションを改善に把握できるので，アプリケーション品質を改善した状態で運用環境に移行できます。
- J2EE，.NET，ERP，CRM（Siebel，Oracle，PeopleSoft，SAP など）の環境で平均復旧時間（MTTR）を短縮します。
- HP HP Business Availability Center，LoadRunner，Performance Center との完全な統合が可能です。

# 第II部

## オペレーティング・システム

# 第3章

# Windows の監視

Performance Center では，Windows プラットフォームで稼働するアプリケーションのパフォーマンス・テストを行う包括的な監視ソリューションが提供されています。

**本章の内容**



## 概要

IT 組織で使用されているアプリケーションは大部分が Windows ベースなので, Performance Center の Windows オペレーティング・システムのパフォーマンス・カウンタを使用することによって，テスト対象アプリケーションの動作を追跡できます。

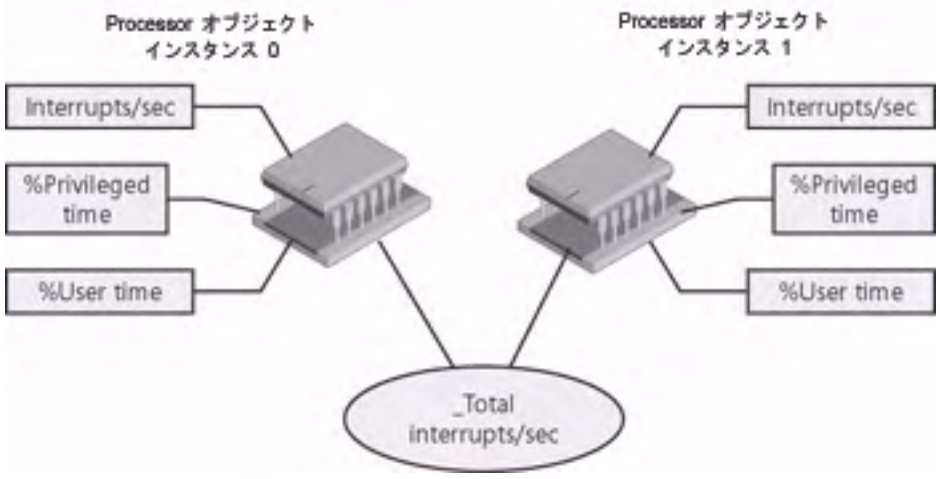
本章では，重要な監視機能について，メトリックスとしきい値の標準設定（該当する場合）を紹介します。本章の内容は，HP のオペレーティング・システム管理者，HP のプロフェッショナル・サービス組織，技術文書，業界の専門家による書籍などから収集したベストプラクティス・データと専門知識に基づく調査をベースにしたものです。

Windows 2000 以降の新しい Windows プラットフォームでは，パフォーマンス関連情報の収集，表示，再利用を簡単に実行できる各種機能が搭載されています。このような機能は，さまざまなサンプリング技術を使って定期的にパフォーマンス監視データを生成することにより，パフォーマンスの問題の診断に非常に役立つ情報を提供します。また，処理を効率化する設計が採用されているので，運用環境への影響を最小限にとどめながら継続的な実行が可能です。

# アーキテクチャ

## オブジェクト

関連のあるパフォーマンス統計は，**オブジェクト**ごとにまとめられます。たとえば，プロセッサの全体的な使用率に関連する測定値には **Interrupts/sec** や **% User Time** があり，これは Processor オブジェクトに含まれます。パフォーマンス・オブジェクトには 1 つ以上の**インスタンス**が含まれ，各インスタンスには一意の名前が割り当てられています。たとえば，プロセッサが複数搭載されているマシンにはインスタンスが複数存在し，それぞれがプロセッサの測定値セットを示します。プロセッサのパフォーマンス・カウンタは，Processor オブジェクトに含まれる名前付きのインスタンスに関連付けられています。次の図で示すように，インスタンス名から，そのインスタンスが関連付けられているカウンタ群を識別できます。



## カウンタ

パフォーマンス・**カウンタ**は，それぞれの時間間隔で測定されたパフォーマンス統計を数値で示したものです。パフォーマンス・カウンタは，パスごとに一意に特定され，通常は次の構文で指定します。

```
¥¥Computer_name¥Object(Parent/Instance#Index)¥Counter
```

**Computer_name** のパス指定はオプションです。

関連付けられるインスタンスが 1 つしかないオブジェクト（System や Memory など）は，次の構文で指定します。

```
¥Object¥Counter
```

## カウンタのタイプ

カウンタには，それぞれタイプがあります。カウンタ・タイプからパフォーマンス統計の収集方法がわかるので，タイプを把握しておくと便利です。

次に，重要なカウンタ・タイプをいくつか紹介します。

- **瞬間値を示すカウンタ**：最新の測定値を表示します。
- **時間間隔を示すカウンタ**：ある時間間隔におけるアクティビティの変化量を表示します。
- **経過時間を示すカウンタ**：時間間隔に渡って収集され，集計されません。
- **平均値を示すカウンタ**：所定の時間間隔に収集された値の平均値を表示します。

# Processor - 最も重要なカウンタ

プログラムの実行スレッドは，プロセッサ（CPU）リソースを消費します。スレッドは，ユーザ・モードのプロセスまたはオペレーティング・システム・カーネルに含まれています。パフォーマンス・カウンタを使用することにより，スレッドなどの実行可能単位がどの程度の CPU プロセス時間を消費しているのかを測定できます。また，測定したプロセッサ使用率から，どのアプリケーションがどの程度の CPU 時間を消費しているかがわかります。

| カウンタ | 説明 |
|---|---|
| **% Processor Time カウンタ** | アイドルでない状態のスレッドを実行するプロセッサ時間の割合（パーセンテージ）。 |
| **% Privileged Time カウンタ** | プロセス・スレッドが特権モードでコードを実行した時間の割合（パーセンテージ）。 |
| **% Interrupt Time カウンタ** | プロセッサが，サンプル期間中にハードウェア割り込みの受信と処理に費やした時間。 |
| **Processor Queue Length カウンタ** | プロセッサ・キュー内で待機しているスレッドの数。 |
| **Context Switches カウンタ** | コンピュータ上にあるすべてのプロセッサについて，スレッド間の切り替えが発生した割合の合計。 |
| **System Up Time カウンタ** | システム全体の可用性。 |

### % Processor Time カウンタ

| | |
|---|---|
| **正式名** | Processor(_Total)¥% Processor Time カウンタ |
| **カウンタのタイプ** | サンプル間隔（ビジー状態の %） |
| **説明** | サンプル間隔内での全体的なプロセッサ使用率の平均値です。アイドル・スレッドの実行時間以外では，プロセッサは実質的な作業負荷を処理するビジー状態だとみなされます。 |
| **使用方法** | 全体的なプロセッサ使用率を示す主要な指標です。ビジー率は 0 ～100% の範囲で示されます。Processor オブジェクトの _Total インスタンスは，プロセッサ使用率インスタンスすべての**平均**値を示します。 |
| **パフォーマンス** | プロセッサが潜在的なボトルネックかどうかを判断する際の主要な指標です。 |
| **操作** | 使用率が 100% の状態を維持する場合，ランナウェイ・プロセスである可能性があります。Process(n)¥% Processor Time カウンタを使って，ランナウェイ・プロセス・スレッドが無限ループに陥っていないか調査してください。 |
| **しきい値** | 応答型の作業負荷の場合，使用率が 80～90% を超える状態が続く場合には注意が必要です。スループット型の作業負荷の場合，容量に制限がある場合を除いて，使用率が高い状態が長時間続いても問題になることはほとんどありません。 |
| **関連するカウンタ** | ➤ Processor(_Total)¥% Privileged Time（49ページを参照してください）<br>➤ Processor(_Total)¥% User Time<br>➤ Processor(n)¥% Processor Time<br>➤ Process(n)¥% Processor Time Thread(n/Index#)¥% Processor Time |

**注**：マシン上で使用率が高いプロセッサが存在しても，対処が必要な問題が発生しているとは限りません。これ以外にも，プロセッサ関連のカウンタ（**% Privileged Time** や **Processor Queue Length** など）の中に直線的に増大しているものがある場合には，CPU 使用率を調査することをお勧めします。

## % Privileged Time カウンタ

| | |
|---|---|
| **正式名** | Processor(_Total)¥% Privileged Time カウンタ |
| **カウンタのタイプ** | サンプル間隔（ビジー状態の %） |
| **説明** | サンプル間隔中に，特権モードまたはカーネル・モードでの処理に費やされた全体的なプロセッサ使用率の平均値です。オペレーティング・システム関数は，すべて特権モードで実行されます。特権モードには，デバイスの入出力操作を開始するデバイス・ドライバ・コードや，割り込み処理を完了するための遅延プロシージャ呼び出しがあります。 |
| **使用方法** | Processor オブジェクトの _Total インスタンスは，プロセッサ使用率インスタンスすべての**平均**値を示します。全体的な **% Processor Time** に対する **% Privileged Time** の比率（特権モード比率）は，作業負荷によって変動します。 |
| **パフォーマンス** | デバイス・ドライバ関数などのオペレーティング・システム関数が，潜在的なプロセッサ・ボトルネックになっているかどうかを判断する際の補助的な指標です。 |
| **操作** | ランナウェイ・プロセス・スレッドが無限ループに陥っている場合，プロセスのステータスから，この問題によってシステム・モジュールが影響を受けているかどうかを特定できます。 |
| **しきい値** | 75% を超える状態が続く場合，ボトルネックが発生していることを示します。 |
| **関連するカウンタ** | ➤ Processor(_Total)¥% Interrupt Time<br>➤ Processor(_Total)¥% DPC Time<br>➤ Process(n)¥% Privileged Time |

**注：** 特権モード比率に推奨値はありません。ただし，一定の作業負荷で急激な変化がみられた場合には，原因を調査することをお勧めします。

## % Interrupt Time カウンタ

| | |
|---|---|
| **正式名** | Processor(_Total)¥% Interrupt Time カウンタ |
| **カウンタのタイプ** | サンプル間隔（ビジー状態の %） |
| **説明** | サンプル間隔中に, 割り込みモードでの処理に費やされた全体的なプロセッサ使用率の平均値です。割り込みモードで実行されるのは，デバイス・ドライバ関数である割り込みサービス・ルーチン（ISR）のみです。 |
| **使用方法** | Processor オブジェクトの _Total インスタンスは，プロセッサ使用率インスタンスすべての**平均**値を示します。ISR が処理する割り込み処理は，優先度が最も高くなります。割り込み処理はシステム関数であり，関連付けられたプロセスはありません。% Interrupt Time の値が大きい場合，デバイスの誤動作を示しますが，デバイスを特定することはできません。カーネル・デバッガである Kernrate を使用すると, ディスパッチの頻度が高い ISR を特定できます。 |
| **パフォーマンス** | このカウンタは, ハードウェア割り込みの受信と処理に費やされたプロセッサ時間の割合をパーセンテージで示します。この値は，ネットワーク・アダプタなど割り込みを生成するデバイスの動作を間接的に示しています。カウンタの値が急増する場合は，ハードウェア障害が発生している可能性があります。 |
| **操作** | デバイスの誤動作が原因で潜在的なプロセッサ・ボトルネックが発生しているかどうかを判断する際の補助的な指標です。 |
| **しきい値** | プロセッサによって異なります。 |
| **関連するカウンタ** | ➤ Processor(_Total)¥Interrupts/sec<br>➤ Processor(_Total)¥% DPC Time<br>➤ Processor(_Total)¥% Privileged Time |

## Processor Queue Length カウンタ

| | |
|---|---|
| **正式名** | System¥Processor Queue Length カウンタ |
| **カウンタのタイプ** | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| **説明** | プロセッサの Ready Queue 内で遅延状態であるとみなされ，実行まで待機しているスレッドの数です。プロセッサの Ready Queue で待機するスレッドは，プロセッサがアイドルになった時点で，優先度の高い順に処理されます。 |
| **使用方法** | 自発的に待機状態になるプログラム・スレッドは多数あります。したがって，アクティブ・スレッドのサブセットが，プロセッサ・キューの実質的な上限になります。 |
| **パフォーマンス** | プロセッサが潜在的なボトルネックかどうかを判断する際の補助的な指標です。 |
| **操作** | 容量の制限が原因でアプリケーションに過度な遅延が発生している可能性があることを示します。 |
| **しきい値** | 使用率が非常に高いプロセッサを 1 基のみ搭載するマシンでは，Processor Queue Length が 5 を超える状態が続く場合，プロセッサが処理可能なレベルを上回る作業負荷が発生している可能性があります。Ready Queue length が 10 を超え，プロセッサ使用率も飽和状態に近い場合，プロセッサが過負荷状態であることが明確です。マルチプロセッサ環境では，Processor Queue Length の値を物理プロセッサの数で割ってください。プロセッサ関係を使って非対称マルチプロセッサ構成を行っている環境で Processor Queue Length の値が大きい場合は，設定のバランスがとれていないことを示します。 |
| **関連するカウンタ** | Thread(parent-process¥Index#)¥Thread State |

## Context Switches カウンタ

| | |
|---|---|
| **正式名** | System¥Context Switches/sec カウンタ |
| **カウンタのタイプ** | サンプル間隔での差異（1 秒あたりの速度） |
| **説明** | 実行中のスレッドが別のスレッドに切り替わると，コンテキスト・スイッチが発生します。Windows ではマルチスレッド操作がサポートされているので，コンテキスト・スイッチは正常な動作です。ユーザ・モードのスレッドが権限を持つオペレーティング・システム関数を呼び出すと，ユーザ・モード・スレッドと，呼び出された関数を特権モードで実行するカーネル・モード・スレッド間でコンテキスト・スイッチが発生します。 |
| **使用方法** | コンテキスト・スイッチは通常のシステム機能であり，作業負荷の結果としてコンテキスト・スイッチが発生します。コンテキスト・スイッチの回数が多くても，通常は問題が発生しているとは限らず，CPU 容量が上限に達したことを示すものでもありません。一般的に，コンテキスト・スイッチの発生回数をチューニングする方法はほとんどありません。過去の標準的なレベルを大きく上回った場合には，デバイスの誤動作などの問題が発生している可能性があります。Context Switches/sec の値と Processor(_Total)¥Interrupts/sec カウンタには相関関係があるので，比較により検討してください。 |
| **パフォーマンス** | コンテキスト・スイッチの回数が多いと，アプリケーション設計やスケーラビリティに潜在的な問題があることを示します。 |
| **操作** | コンテキスト・スイッチは，スレッドが実行中に，それよりも優先度の高いスレッドがプロセッサを横取りする場合や，優先度の高いスレッドがブロックする場合に発生します。同じ優先度のスレッドが多数発生すると，コンテキスト・スイッチの回数が多くなります。これは，多数のスレッドがシステム上のプロセッサを奪い合っている状態を示します。また，プロセッサ使用率があまり高くなく，コンテキスト・スイッチの回数が非常に低い場合には，スレッドがブロックされている可能性があります。 |

| | |
|---|---|
| **しきい値** | 重要なサーバ・マシンを対象に，過去の標準的なレベルから大きく外れる場合は警告を通知するように設定します。一般的には，プロセッサあたりのコンテキスト・スイッチが 1 秒あたり 5,000 回未満の場合には問題ありません。これが 15,000 回を**超える**場合には，ボトルネックが発生しています。 |
| **関連するカウンタ** | Thread¥Context Switches/sec |

### System Up Time カウンタ

| | |
|---|---|
| **正式名** | System¥System Up Time カウンタ |
| **カウンタのタイプ** | 経過時間 |
| **説明** | コンピュータを前回再起動してから経過した稼働時間を秒単位で示します。 |
| **使用方法** | システムの可用性を示す主要な指標です。 |
| **パフォーマンス** | なし |
| **操作** | システムの可用性を報告します。 |
| **しきい値** | なし |
| **関連するカウンタ** | Process(n)¥Elapsed Time |

---

**注：**パフォーマンスを測定する前に，サーバおよびサーバ・アプリケーションが稼働していて使用可能な状態であることを確認してください。

---

# Memory - 最も重要なカウンタ

Windows では，物理メモリと仮想メモリを管理します。RAM 容量が不足すると，ディスクへのページングが過剰に発生してディスク帯域幅が大量に消費されるため，ディスク・パフォーマンスの低下が報告されます。したがって，ディスクへのページング率は，メモリ・パフォーマンスを示す重要な指標となります。32 ビット・システムでは，仮想メモリの容量には 4 GB の制限があり，2 GB のプライベート領域と 2 GB の共有領域に分けられています。物理メモリの容量が大きくても仮想メモリ不足を補うことはできません。アプリケーションが使用した割り当てメモリが解放されないために**メモリ・リーク**が発生すると，致命的なクラッシュにつながる恐れがあります。

使用可能な RAM 容量が不足している場合には，割り当てられた物理メモリの使用状況と，問題のあるプロセスの常駐ページ（ワーキング・セット）のサイズを把握することが重要です。

| カウンタ | 説明 |
|---|---|
| **Available Bytes カウンタ** | コンピュータ上で実行中のプロセスが使用可能な物理メモリの容量。 |
| **Working Set カウンタ** | 各プロセスの常駐ページの数。 |
| **Pages/sec カウンタ** | ハード・ページ・フォルトを解決するために，ディスクに対して読み書きされるページの数。 |
| **Page Reads/sec カウンタ** | プロセスのワーキング・セットが物理メモリより大きすぎるため，ディスクへのページングが発生。 |
| **Pool Nonpaged Bytes カウンタ** | ディスクに書き込むことができないが，割り当てられている限り物理メモリ内に存在するオブジェクト用のシステム・メモリ（オペレーティング・システムが使用する物理メモリ）領域のサイズ。 |
| **Paged Pool Bytes カウンタ** | メモリ・リーク。 |
| **Paged Pool Failures カウンタ** | ページ・プールからの割り当てに失敗した回数。 |
| **Cache Bytes カウンタ** | 静的なファイル・キャッシュのサイズ。 |

| カウンタ | 説明 |
|---|---|
| System Cache Resident Bytes カウンタ | システム・ファイル・キャッシュに割り当てられた常駐ページの数。 |
| Committed Bytes カウンタ | 応答時間の悪化につながる過剰なページング。 |

## Available Bytes カウンタ

| | |
|---|---|
| 正式名 | Memory¥Available Bytes カウンタ |
| カウンタのタイプ | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| 説明 | プロセスの常駐ページのセットを示します。ページ・フォルトが発生しない状態で，プロセスが使用可能な RAM 内の割り当てページの数です。 |
| 使用方法 | ゼロ・メモリ，空きメモリ，スタンバイ・メモリの領域の合計です。空きメモリは使用可能なメモリです。ゼロ・メモリはゼロで埋められたメモリのページであり, 前のプロセスが使用したデータを後のプロセスが参照できないようにします。スタンバイ・メモリは，ディスクへのルート上にあるプロセスのワーキング・セット（物理メモリ）から削除されているが，再呼び出しが可能なメモリです。 |
| パフォーマンス | メモリが不足している場合，Process(n)¥Working Set を使って RAM の使用容量をプロセスごとに把握できます。 |
| 操作 | なし |
| しきい値 | インストールされている RAM で 20 ～ 25% 未満の状態が続く場合，メモリが不足していることを示します。 |
| 関連するカウンタ | ➤ Memory¥Available Byte<br>➤ Memory¥Committed Bytes<br>➤ Process(n)¥Private Bytes<br>➤ Process(n)¥Virtual Bytes<br>➤ Process(n)¥Pool Paged Bytes |

## Working Set カウンタ

| | |
|---|---|
| **正式名** | Process(*)¥Working Set カウンタ |
| **カウンタのタイプ** | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| **説明** | プロセスの常駐ページのセットを示します。ページ・フォルトが発生しない状態で，プロセスが使用可能な RAM 内の割り当てページの数です。 |
| **使用方法** | Process(n)¥Working Set は，アクティブなプロセスによる現在の RAM 使用率を追跡します。IIS，Exchange，SQL Server など一部のサーバ・アプリケーションは，専用のプロセス・ワーキング・セットを管理します。Process オブジェクトの Process(_Total)¥Working Set を監視することにより，すべてのプロセスのアドレス領域全体で RAM がどのように割り当てられているかを把握できます。 |
| **パフォーマンス** | メモリが不足している場合，Process(n)¥Working Set を使って RAM の使用容量をプロセスごとに把握できます。 |
| **操作** | なし |
| **しきい値** | 継続的に 10% 以上増加する場合は，物理メモリに制限があることを示します。 |
| **関連するカウンタ** | ➤ Memory¥Available Bytes<br>➤ Memory¥Committed Bytes<br>➤ Process(n)¥Private Bytes<br>➤ Process(n)¥Virtual Bytes<br>➤ Process(n)¥Pool Paged Bytes |

## Pages/sec カウンタ

| | |
|---|---|
| **正式名** | Memory¥Pages/sec カウンタ |
| **カウンタのタイプ** | サンプル間隔での差異（1 秒あたりの速度） |
| **説明** | サンプル間隔中に，ディスクへのページングが発生した回数を示します。Pages/sec は，Page Reads/sec と Page Writes/sec の合計です。 |
| **使用方法** | Page Reads/sec カウンタは，ハード・ページ・フォルトを示します。実行中のスレッドは，プロセスのワーキング・セット内に存在しない仮想メモリ内のページを参照します。このページは，遷移中のトリミングされたページではなく，メモリ内にまだ存在しています。ディスクからページを取得する I/O 操作を行うと，その間スレッドには遅延が発生します。オペレーティング・システムは，ページをディスクから RAM 内の使用可能なページにコピーしてから，スレッドを再度ディスパッチします。 |
| **パフォーマンス** | 実メモリが潜在的なボトルネックかどうかを判断する際の主要な指標です。 |
| **操作** | 過剰なページングが応答時間の悪化につながる可能性を示します。 |
| **しきい値** | ディスクへのページングあたりの Pages/sec が 50 を超える場合には注意が必要です。 |
| **関連するカウンタ** | ➤ Memory¥Available Bytes<br>➤ Memory¥Committed Bytes<br>➤ Process(n)¥Working Set |

---

**注：**通常の場合，RAM を追加することによって過剰なページングを解消できます。ディスク帯域幅には上限があります。ページング操作に使用される RAM 容量は，他のアプリケーション・ファイル操作に使用できなくなります。

---

## Page Reads/sec カウンタ

| | |
|---|---|
| **正式名** | Memory¥Page Reads/sec |
| **カウンタのタイプ** | サンプル間隔での差異（1 秒あたりの速度） |
| **説明** | プロセスのワーキング・セットが物理メモリより大きすぎるため，ディスクへのページングが発生していることを示します。このカウンタは読み取り操作の回数を示し，1 回の操作で取得されたページ数には関係しません。値が大きいほど，メモリ・ボトルネックが発生している可能性が高くなります。 |
| **使用方法** | ページ読み取り操作の回数が少なく，Physical Disk¥% Disk Time と Physical Disk¥Avg. Disk Queue Length の値が大きい場合，ディスク・ボトルネックが発生している可能性があります。ページ読み取り回数が減ってもキューの長さが大きくならない場合は，メモリが不足しています。 |
| **パフォーマンス** | 実メモリが潜在的なボトルネックかどうかを判断する際の主要な指標です。 |
| **操作** | 過剰なページングが応答時間の悪化につながる可能性を示します。 |
| **しきい値** | 常に 5 を超える場合は，読み取り要求でページ・フォルトが多発していることを示します。 |
| **関連するカウンタ** | ➤ Memory¥Pages/sec<br>➤ Memory¥Page Writes/sec |

## Pool Nonpaged Bytes カウンタ

| | |
|---|---|
| **正式名** | Memory¥Pool Nonpaged Bytes カウンタ |
| **カウンタのタイプ** | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| **説明** | 非ページ・プールから割り当てられたページは，常に RAM 内に存在します。 |
| **使用方法** | すべての TCP 接続のステータス情報は，非ページ・プールに格納されます。この値をページ・サイズで割ると，割り当て済みのページ数を取得できます。 |
| **パフォーマンス** | メモリ不足の場合，Pool Nonpaged Bytes から，システム関数が使用しているページングできない RAM の容量がわかります。 |
| **操作** | なし |
| **しきい値** | Memory¥Pool Nonpaged Bytes が, システム起動時の値から 10% 以上増加している場合には注意が必要です。この場合，重大なメモリ・リークが発生していることを示します。 |
| **関連するカウンタ** | ➤ Pool Paged Bytes<br>➤ Pool Paged Resident Bytes<br>➤ System Cache Resident Bytes<br>➤ System Code Resident Bytes<br>➤ System Driver Resident Bytes<br>➤ Process(_Total)¥Working Set |

## Paged Pool Bytes カウンタ

| | |
|---|---|
| **正式名** | Memory¥Paged Pool Bytes カウンタ |
| **カウンタのタイプ** | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| **説明** | システムのページ・プール内でコミットされた仮想メモリ・ページの数です。システム関数は，ページ・プールからページ・アウトできる仮想メモリ・ページを割り当てます。プロセスによって呼び出されたシステム関数も，仮想メモリ・ページをページ・プールから割り当てます。 |
| **使用方法** | Memory¥Paged Pool Bytes は，システム・ページ・プール内で割り当てられている仮想メモリの容量を示します。Memory¥Paged Pool Resident Bytes は，RAM 内に存在するページ・プールの現在のページ数を示します。これ以外の残りのページはページ・アウトです。 |
| **パフォーマンス** | なし |
| **操作** | メモリ・リークが発生しているプロセスを特定する際の主要な指標です。 |
| **しきい値** | プロセスの Process(n)¥Paged Pool Bytes の増加が 10% を超える場合，メモリ・リークが発生している可能性があります。 |
| **関連するカウンタ** | ➤ Memory¥Commit Limit<br>➤ Memory¥% Committed Bytes in Use<br>➤ Process(n)¥Pool Paged Bytes<br>➤ Process(n)¥Virtual Bytes |

---

**注 :** 標準に準拠しない一部プロセスにおいて，システムのページ・プールでメモリ・リークが発生している可能性があります。Process(n)¥Paged Pool Bytes カウンタを使用すると，このようにメモリ・リークが発生しやすいアプリケーションを特定できます。

---

## Paged Pool Failures カウンタ

| | |
|---|---|
| **正式名** | Server¥Paged Pool Failures カウンタ |
| **カウンタのタイプ** | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| **説明** | サーバ・サービスの初期化以降，ページ・プールの割り当てに失敗した累計回数です。 |
| **使用方法** | ファイル・サーバ・サービスには，ページ・プールから仮想メモリ・ページを割り当てる関数が多数存在します。メモリ・リークによってページ・プールが消費されると，ファイル・サーバ・サービスはページ・プールから仮想メモリを割り当てることができなくなります。仮想メモリ割り当ての呼び出しが失敗すると，ファイル・サーバ・サービスは正常に失敗を処理し，失敗を報告します。仮想メモリの割り当ての失敗を正常に処理できないアプリケーションやシステム関数が多数存在するため，このカウンタは，割り当ての失敗が原因で発生したメモリ・リークを特定できる唯一のカウンタです。 |
| **パフォーマンス** | なし |
| **操作** | ページ・プール内の仮想メモリの不足を特定する際の主要な指標です。 |
| **しきい値** | 値がゼロ以外の場合，ボトルネックを示しています。 |
| **関連するカウンタ** | ➤ Memory¥Pool Paged Bytes<br>➤ Memory¥Commit Limit<br>➤ Memory¥% Committed Bytes in Use<br>➤ Server¥Pool Paged Bytes<br>➤ Process(n)¥Pool Paged Bytes |

## Cache Bytes カウンタ

| | |
|---|---|
| **正式名** | Memory¥Cache Bytes カウンタ |
| **カウンタのタイプ** | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| **説明** | システム・ワーキング・セット内の常駐ページのセットを示します。ページ・フォルトが発生しない状態で，カーネル・スレッドが使用可能な RAM 内の割り当てページの数です。 |
| **使用方法** | システム・ワーキング・セットは，他のワーキング・セットと同様に，ページ置換の対象になります。 |
| **パフォーマンス** | メモリ不足の場合，Cache Bytes から，システム関数が使用しているページング可能な RAM 容量がわかります。 |
| **操作** | なし |
| **しきい値** | なし |
| **関連するカウンタ** | ➤ Pool Nonpaged Bytes<br>➤ Pool Paged Resident Bytes<br>➤ System Cache Resident Bytes<br>➤ System Code Resident Bytes<br>➤ System Driver Resident Bytes<br>➤ Process(_Total)¥Working Set |

## System Cache Resident Bytes カウンタ

| | |
|---|---|
| **正式名** | Memory¥System Cache Resident Bytes カウンタ |
| **カウンタのタイプ** | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| **説明** | システム・ファイル・キャッシュに割り当てられた常駐ページの数を示します。このカウンタは，現在 RAM 内に常駐するファイル・キャッシュの仮想メモリのページ数を追跡します。 |
| **使用方法** | ファイルおよびプリント・サーバ上は，System Cache Resident Bytes が示す RAM 消費容量が非常に大きな値になることがあります。この値はシステムのワーキング・セット（Cache Bytes）の一部であり，Available Bytes が小さい場合にはページ・トリミングの対象になります。 |
| **パフォーマンス** | システム・ファイル・キャッシュが非効率だと，キャッシュを多用するサーバ･アプリケーションのパフォーマンスに影響が発生します。このようなアプリケーションには，サーバ，リダイレクタ，NTFS，IIS があります。 |
| **操作** | メモリ・リークが発生しているプロセスを特定する際の主要な指標です。 |
| **しきい値** | なし |
| **関連するカウンタ** | Memory¥Cache Bytes |

## Committed Bytes カウンタ

| | |
|---|---|
| **正式名** | Memory¥Committed Bytes カウンタ |
| **カウンタのタイプ** | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| **説明** | コミットされた仮想メモリ・ページの数です。コミットされたページは，RAM 内の物理ページまたはページング・ファイル上のスロットによって予約されます。 |
| **使用方法** | Committed Bytes は，プロセスのアドレス空間が割り当てた仮想メモリの合計を示します。Committed Bytes:RAM の比率が 1 より大きい場合は仮想メモリが RAM より大きいことを示すので，何らかのメモリ管理が必要になります。Committed Bytes:RAM の比率が 1.5 を超える場合，一般的に，ページング・ディスクの帯域幅の上限までディスクへのページングが増大します。 |
| **パフォーマンス** | Committed Bytes:RAM の比率は，実メモリの不足を示す補助的な指標です。 |
| **操作** | 過剰なページングが応答時間の悪化につながる可能性を示します。 |
| **しきい値** | Committed Bytes:RAM の比率が 1.5 を超える場合，実メモリがボトルネックになっていることを示しています。 |
| **関連するカウンタ** | ➤ Memory¥Pages/sec<br>➤ Memory¥Commit Limit<br>➤ Memory¥% Committed Bytes in Use<br>➤ Memory¥Pool Paged Bytes<br>➤ Process(n)¥Private Bytes Process(n)¥Virtual Bytes |

**注：**Committed Bytes:RAM の比率が 1.5 に近い値または上回った場合には，メモリを追加してください。

# I/O - 最も重要なカウンタ

Windows では，I/O マネージャが物理および論理ディスク操作を管理します。**論理ディスク**は，一意のドライブ文字で示される 1 つのファイル・システムです。**物理ディスク**はストレージ・デバイスを内部的に示したものであり，SCSI，RAID，SATA などがあります。アレイ・コントローラや RAID など複雑なストレージ・システムでは，それを構成する物理ディスク・ハードウェアの属性をオペレーティング・システムが直接認識できない場合があります。このような属性には，ディスクの数，ディスクの速度，シーク時間，回転速度，ビット密度，最適化機能（オンボードのメモリ・バッファなど）があり，パフォーマンスに大きな影響を与える可能性があります。また，メモリ・バッファやコマンド・キューなど高度な機能が搭載されていると，パフォーマンスが 25～50% 向上することがあります。

ディスク・パフォーマンスは，特にディスク・ページングが発生すると急激に低下する傾向があるので，プロアクティブな対策を講じることが重要です。

| カウンタ | 説明 |
|---|---|
| **Avg. Disk secs/transfer カウンタ** | 物理ディスクの潜在的なボトルネック。 |
| **% Idle Time カウンタ** | 物理ディスクの使用率。 |
| **Disk Transfers/sec カウンタ** | 物理ディスクが潜在的なボトルネックになっているかどうかの指標。 |
| **Avg. Disk Queue Length カウンタ** | ディスクの潜在的なボトルネック（他のカウンタと組み合わせて使用する必要がある）。 |
| **Split IO/sec カウンタ** | 最適化の指標。 |
| **Free Megabytes カウンタ** | 論理ディスク容量の使用率。 |

## Avg. Disk secs/transfer カウンタ

| | |
|---|---|
| **正式名** | Physical Disk(n)¥Avg. Disk secs/transfer カウンタ |
| **カウンタのタイプ** | 平均値 |
| **説明** | サンプル間隔内で, 物理ディスク要求の全体的な応答時間の平均値を示します。Avg. Disk secs/transfer には，デバイスのサービス時間とキュー時間の両方が含まれます。 |
| **使用方法** | 物理ディスクの I/O パフォーマンスを示す主要な指標です。パフォーマンスはディスク構成によって左右され, ディスク構成はオペレーティング・システムで認識されます。シーク時間，回転速度，記録密度，インタフェース速度などのパフォーマンス属性は，ディスクごとに異なります。高額な高性能ディスクでは，パフォーマンスが 50% 向上することがあります。 |
| **パフォーマンス** | ディスクが潜在的なボトルネックかどうかを判断する際の主要な指標です。 |
| **操作** | ディスク応答時間の悪化は, アプリケーションの応答時間の悪化につながります。 |
| **しきい値** | 基盤となるディスク・ハードウェアによって異なりますが，一般的に 18 ミリ秒以下が適正値です。 |
| **関連するカウンタ** | ➤ Physical Disk(n)¥Disk Transfers/sec<br>➤ Physical Disk(n)¥% Idle Time<br>➤ Physical Disk(n)¥Current Disk Queue Length |

---

**注：**このカウンタは，過度なディスク断片化，ディスク速度の低下，ディスク障害を示す場合があります。**Physical Disk¥Avg. Disk sec/Transfer** カウンタの値と **Memory¥Pages/sec** カウンタの値を掛けた値が 0.1 を超える場合，ページングがディスク・アクセス時間の 10% を超えていることを示しているので，RAM の追加が必要です。

---

### % Idle Time カウンタ

| | |
|---|---|
| **正式名** | Physical Disk(n)¥% Idle Time カウンタ |
| **カウンタのタイプ** | サンプル間隔（ビジー状態の %） |
| **説明** | サンプル間隔中に，ディスクがアイドル状態になっている時間の割合をパーセンテージで示します。100 から % Idle Time の値を差し引いた値がディスク使用率です。 |
| **使用方法** | ディスク使用率は，次の式で計算できます。<br>Physical Disk(n)¥Disk utilization ＝ 100% － Physical Disk(n)¥% Idle Time<br>ディスク・アレイの場合，ディスク使用率をアレイ内のディスク数で割ると，個々のディスクの使用率を概算できます。要求が個別にディスクに到着することを前提とすると，ディスク使用率が 100% に近付くほどキュー時間が急増します。 |
| **パフォーマンス** | 物理ディスクが過負荷状態かどうか，潜在的なボトルネックになっているかどうかを判断する際の主要な指標です。 |
| **操作** | キュー時間が増大するとディスク応答時間が悪化し，これがアプリケーションの応答時間の悪化につながります。 |
| **しきい値** | % Idle Time が 20% 未満の場合には注意が必要です。 |
| **関連するカウンタ** | ➤ Physical Disk(n)¥Avg<br>➤ Disk secs/Transfer<br>➤ Physical Disk(n)¥Disk Transfers/sec<br>➤ Physical Disk(n)¥Current Disk Queue Length |

---

**注：**ディスク使用率，ディスク・サービス時間，ディスク・キュー時間を計算することによって，ディスク・サブシステムのパフォーマンス低下やディスクの過負荷状態が発生しているかどうかを判断できます。

---

## Disk Transfers/sec カウンタ

| | |
|---|---|
| **正式名** | Physical Disk(n)¥Disk Transfers/sec カウンタ |
| **カウンタのタイプ** | サンプル間隔での差異（1 秒あたりの速度） |
| **説明** | サンプル間隔中に完了した物理ディスク要求の数を示します。 |
| **使用方法** | 物理ディスクの I/O 操作を示す主要な指標です。また，ディスク到着回数とも呼ばれます。この値は，次に示すように読み取りと書き込みに分類されます。<br>Physical Disk(n)¥Disk Transfers/sec ＝ Physical Disk(n)¥Disk Reads/sec ＋ Physical Disk(n)¥Disk Writes/sec<br>使用率の法則から，% Idle Time を元にディスク・サービス時間を計算できます。 |
| **パフォーマンス** | ディスクが潜在的なボトルネックかどうかを判断する際の主要な指標です。 |
| **操作** | ディスク応答時間の悪化は，アプリケーションの応答時間の悪化につながります。 |
| **しきい値** | 基盤となるディスク・ハードウェアによって変動します。 |
| **関連するカウンタ** | ➤ Physical Disk(n)¥Disk Transfers/sec<br>➤ Physical Disk(n)¥% Idle Time<br>➤ Physical Disk(n)¥Current Disk Queue Length |

## Avg. Disk Queue Length カウンタ

| | |
|---|---|
| **正式名** | Physical Disk(n)¥Avg. Disk Queue Length カウンタ |
| **カウンタのタイプ** | 複合カウンタ |
| **説明** | サービスの実行中またはディスク・サービスの待機中の物理ディスク要求の推定平均値です。 |
| **使用方法** | 物理ディスク I/O のキューを示す補助的な指標であり，監視には注意が必要です。Avg. Disk Queue Length カウンタの値は，基盤となる物理ディスクの特性をよく理解した上で判断する必要があります。ホスト・オペレーティング・システムでは単一の物理ディスクとして認識されていても，実際には複数の物理ディスクのグループに 1 つの LUN が割り当てられている場合があります。また，アレイ・コントローラを使って，複数の物理ディスクから仮想 LUN が構成されていることもあります。アレイ・コントローラを使用すると，アレイ内の複数のディスクを同時に稼働することが可能になります。このような物理ディスクは，単一のサーバとしてみなすことはできません。<br>% Disk Read Time，% Disk Time，% Disk Write Time は同じ式から計算できますが，100% が上限となる点が異なります。 |
| **パフォーマンス** | ディスクが潜在的なボトルネックかどうかを判断する際の補助的な指標です。 |
| **操作** | なし |
| **しきい値** | 適正値は，スピンドルの数に 2 を加えた値を超えない値です。 |
| **関連するカウンタ** | ➤ Physical Disk(n)¥% Idle Time<br>➤ Physical Disk(n)¥Avg. Disk secs/Transfer<br>➤ Physical Disk(n)¥Disk Transfers/sec<br>➤ Physical Disk(n)¥Current Disk Queue Length<br>➤ Physical Disk(n)¥% Disk Time |

## Split IO/sec カウンタ

| | |
|---|---|
| **正式名** | Physical Disk(n)¥Split IO/sec カウンタ |
| **カウンタのタイプ** | サンプル間隔での差異（1 秒あたりの速度） |
| **説明** | サンプル間隔中, 複数のディスク要求に分割された物理ディスク要求の数を示します。I/O 分割が発生すると，I/O マネージャの測定層はオリジナルの I/O 要求と分割された I/O 要求の両方を分割 I/O としてカウントするため，分割 I/O の数値は I/O マネージャが開始した I/O 操作を正確に示した数値になります。 |
| **使用方法** | 物理ディスクの断片化を示す主要な指標です。<br>要求されたデータのサイズが大きすぎて 1 回の I/O では処理できない場合にも，I/O の分割が発生します。分割 I/O の処理には通常よりも長い時間がかかるので，Physical Disk(n)¥Avg. Disk secs/Transfer と組み合わせて監視することをお勧めします。 |
| **パフォーマンス** | ディスクの最適化ソフトウェアの実行頻度を判断する際の補助的な指標です。 |
| **操作** | ディスク応答時間の悪化は, アプリケーションの応答時間の悪化につながります。 |
| **しきい値** | Split I/O/sec の値が Disk Transfers/sec の値の 20% を超える場合には注意が必要です。 |
| **関連するカウンタ** | ➤ Physical Disk(n)¥Disk Transfers/sec<br>➤ Physical Disk(n)¥Avg. Disk secs/Transfer<br>➤ Physical Disk(n)¥% Idle Time |

---

**注：**ランダムなディスク要求よりもシーケンシャルな要求の方が処理速度は格段に高くなるので，ディスクの Split IO/sec の値が増大した時点でディスクを最適化するか，定期的に最適化することにより，ディスク・パフォーマンスを向上できます。

---

## Free Megabytes カウンタ

| | |
|---|---|
| **正式名** | Logical Disk(n)¥Free Megabytes カウンタ |
| **カウンタのタイプ** | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| **説明** | 論理ディスク上の未割り当て領域の容量をメガバイト単位で示します。<br>非常に容量の大きなファイル・システムでは，Free Megabytes カウンタの計算に時間がかかるため，I/O マネージャの測定レイヤは約 5 分間隔でカウンタの値を計算します。 |
| **使用方法** | 使用済みの論理ディスク容量を示す主要な指標です。 |
| **パフォーマンス** | なし |
| **操作** | 一般的に，ファイル・システム容量の不足は致命的な状況を招きます。 |
| **しきい値** | このカウンタの値または Logical Disk(n)¥% Free Space の値が 10% 未満になった場合は注意が必要です。 |
| **関連するカウンタ** | Logical Disk(n)¥% Free Space |

# Network - 最も重要なカウンタ

Windows では，ネットワーク・トラフィックは最も低いレベルのハードウェア・インタフェースと，それよりも高いレベルのネットワーク・プロトコル（TCP/IP など）で測定されます。ネットワーク・インタフェースの統計データは，ネットワーク・インタフェース・ドライバ層に組み込まれているソフトウェアによって収集されます。このソフトウェアは，送受信されるパケットの数をカウントします。Network Interface オブジェクトでは，インストールされているネットワーク・インタフェース・チップまたはカードごとに複数のインスタンスが生成されます。また，サポート対象プロトコル（TCP，UDP，NetBEUI，NWLink IPX，NWLink NetBIOS，NWLink SPX など）ごとに，これよりも高レベルのカウンタ（Protocol_Object¥Segments Received/sec や Protocol_Object¥Segments Sent/sec など）が提供されます。

| カウンタ | 説明 |
|---|---|
| **Bytes Total/sec カウンタ** | スループットの合計。 |
| **Server Bytes Total/sec** | ネットワークに関する全体的なサーバ使用率。 |
| **Datagrams/sec カウンタ** | IP プロトコルの作業負荷。 |
| **Connections Established カウンタ** | TCP プロトコル接続の成功率。 |
| **Segments Received/sec カウンタ** | 受信した TCP データ・セグメントの数。 |
| **% Interrupt Time カウンタ** | ネットワーク・カードなどのハードウェア・デバイスの割り込みに費やされたプロセッサ時間。 |

## Bytes Total/sec カウンタ

| | |
|---|---|
| **正式名** | Network Interface(n)¥Bytes Total/sec カウンタ |
| **カウンタのタイプ** | サンプル間隔での差異（1 秒あたりの速度） |
| **説明** | サンプル間隔において，このインタフェースで 1 秒あたりに送受信したバイト数の合計です。これは，このインタフェースのスループット（バイト単位）を示します。 |
| **使用方法** | ネットワーク・インタフェースのトラフィックを示す主要な指標です。ネットワーク・インタフェースの使用率は，次の式で計算します。<br>Network Interface(n)¥% Busy ＝ Network Interface(n)¥Bytes Total/sec ÷ Network Interface(n)¥Current Bandwidth<br>交換接続リンクで達成可能な最大帯域幅は，Current Bandwidth カウンタの 90～95% に近い値が適性値です。 |
| **パフォーマンス** | ネットワークが潜在的なボトルネックかどうかを判断する際の主要な指標です。 |
| **しきい値** | Total Bytes/sec の値が回線容量の 80% を超えた場合は注意が必要です。 |
| **関連するカウンタ** | ➤ Network Interface(n)¥Bytes Received/sec<br>➤ Network Interface(n)¥Bytes Sent/sec<br>➤ Network Interface(n)¥Packets Received/sec<br>➤ Network Interface(n)¥Packets Sent/sec<br>➤ Network Interface(n)¥Current Bandwidth |

**注：**このカウンタは，特定のネットワーク・アダプタのトラフィックが飽和状態にあるか，ネットワーク・アダプタの追加が必要かどうかを判断する際の指標になります。

## Server Bytes Total/sec カウンタ

| | |
|---|---|
| **正式名** | Server¥Bytes Total/sec カウンタ |
| **カウンタのタイプ** | サンプル間隔での差異（1 秒あたりの速度） |
| **説明** | サーバがネットワークから送受信したバイト数です。この値は，サーバがどの程度ビジー状態であるかを全体的に示します。 |
| **使用方法** | このカウンタは，ネットワークを介して送受信されたバイト数を示します。値が大きい場合，ネットワーク帯域幅がボトルネックになっていることを示します。すべてのサーバの Bytes Total/sec を合計した値が，ネットワークの最大転送速度にほぼ近い場合には，ネットワークのセグメント化が必要です。 |
| **パフォーマンス** | ネットワークが潜在的なボトルネックかどうかを判断する際の主要な指標です。 |
| **しきい値** | ネットワーク容量の 50% 以下が適正値です。 |
| **関連するカウンタ** | Network Interface(n)¥Bytes Received/sec |

## Datagrams/sec カウンタ

| | |
|---|---|
| **正式名** | IPvn¥Datagrams/sec カウンタ |
| **カウンタのタイプ** | サンプル間隔での差異（1 秒あたりの速度） |
| **説明** | サンプル間隔内で，1 秒あたりに送受信された IP データグラムの合計数です。 |
| **使用方法** | IP トラフィックを示す主要な指標です。 |
| **パフォーマンス** | ネットワークが潜在的なボトルネックかどうかを判断する際の補助的な指標です。 |
| **操作** | IP トラフィックの急激な増加は，ネットワークへの不正侵入が発生している可能性を示します。 |
| **しきい値** | 想定外に 10% 超の増加がみられる場合は，過負荷状態またはセキュリティ違反が発生している可能性を示します。 |
| **関連するカウンタ** | ➤ IPvn¥Datagrams Received/sec<br>➤ IPvn¥Datagrams Sent/sec<br>➤ Network Interface(n)¥Packets/sec |

## Connections Established カウンタ

| | |
|---|---|
| **正式名** | TCPvn¥Connections Established カウンタ |
| **カウンタのタイプ** | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| **説明** | 測定期間が終わった時点で，ESTABLISHED 状態になっている TCP 接続の合計数です。 |
| **使用方法** | TCP セッション接続の動作を示す主要な指標です。<br>確立可能な TCP 接続の数は，非ページ・プールのサイズによって制限されます。非ページ・プールが不足すると，新しく接続を確立できなくなります。 |
| **パフォーマンス** | ネットワークが潜在的なボトルネックかどうかを判断する際の補助的な指標です。 |
| **操作** | TCP 接続の値に急激なスパイクが発生した場合, サービス拒否攻撃が発生している可能性があります。 |
| **しきい値** | 想定外に 10% 超の増加がみられる場合は，過負荷状態またはセキュリティ違反が発生している可能性を示します。 |
| **関連するカウンタ** | ➤ TCPPvn¥Segments Received/sec<br>➤ TCPPvn¥Segments Sent/sec<br>➤ Network Interface(n)¥Packets/sec<br>➤ Memory¥Nonpaged Pool Bytes |

## Segments Received/sec カウンタ

| | |
|---|---|
| **正式名** | TCPvn¥Segments Received/sec カウンタ |
| **カウンタのタイプ** | サンプル間隔での差異（1 秒あたりの速度） |
| **説明** | サンプル間隔内で，確立された接続を介して受信された TCP セグメント数の平均値です。 |
| **使用方法** | TCP ネットワークの負荷を示す主要な指標です。<br>次の式を使って，受信セグメント数の平均値を接続ごとに計算します。<br>TCPvn¥Segments Received/sec ÷ TCPPvn¥Connections Established/sec<br>この値から，将来的なユーザ数の増大に伴う作業負荷を予測することができます。 |
| **パフォーマンス** | ネットワークが潜在的なボトルネックかどうかを判断する際の補助的な指標です。 |
| **操作** | 受信した TCP 要求での急激なスパイクは，ネットワークへの不正侵入が発生している可能性を示します。 |
| **しきい値** | 想定外に 10% 超の増加がみられる場合は，過負荷状態またはセキュリティ違反が発生している可能性を示します。 |
| **関連するカウンタ** | ➤ TCPPvn¥Connections Established/sec<br>➤ TCPPvn¥Segments Sent/sec<br>➤ IPvn¥Datagrams Received/sec<br>➤ Network Interface(n)¥Packets/sec |

## % Interrupt Time カウンタ

| | |
|---|---|
| **正式名** | Processor(_Total)¥% Interrupt Time カウンタ |
| **カウンタのタイプ** | サンプル間隔（ビジー状態の %） |
| **説明** | サンプル間隔中に, 割り込みモードでの処理に費やされた全体的なプロセッサ使用率の平均値です。割り込みモードで実行されるのは，デバイス・ドライバ関数である割り込みサービス・ルーチン（ISR）のみです。 |
| **使用方法** | Processor オブジェクトの _Total インスタンスは，プロセッサ使用率インスタンスすべての**平均**値を示します。ISR が処理する割り込み処理は，優先度が最も高くなります。割り込み処理はシステム関数であり，関連付けられたプロセスはありません。% Interrupt Time の値が大きい場合，デバイスの誤動作を示しますが，デバイスを特定することはできません。カーネル・デバッガである Kernrate を使用すると, ディスパッチの頻度が高い ISR を特定できます。 |
| **パフォーマンス** | このカウンタは, ハードウェア割り込みの受信と処理に費やされたプロセッサ時間の割合をパーセンテージで示します。この値は，ネットワーク・アダプタなど割り込みを生成するデバイスの動作を間接的に示しています。カウンタの値が急増する場合は，ハードウェア障害が発生している可能性があります。 |
| **操作** | デバイスの誤動作が原因で潜在的なプロセッサ・ボトルネックが発生しているかどうかを判断する際の補助的な指標です。 |
| **しきい値** | プロセッサによって異なります。 |
| **関連するカウンタ** | ➤ Processor(_Total)¥Interrupts/sec<br>➤ Processor(_Total)¥% DPC Time<br>➤ Processor(_Total)¥% Privileged Time |

# 第4章

# Unix の監視

HP Performance Center では，各種 Unix プラットフォームで稼働するアプリケーションのパフォーマンス・テストを行う包括的な監視ソリューションが提供されています。

**本章の内容**

# 概要

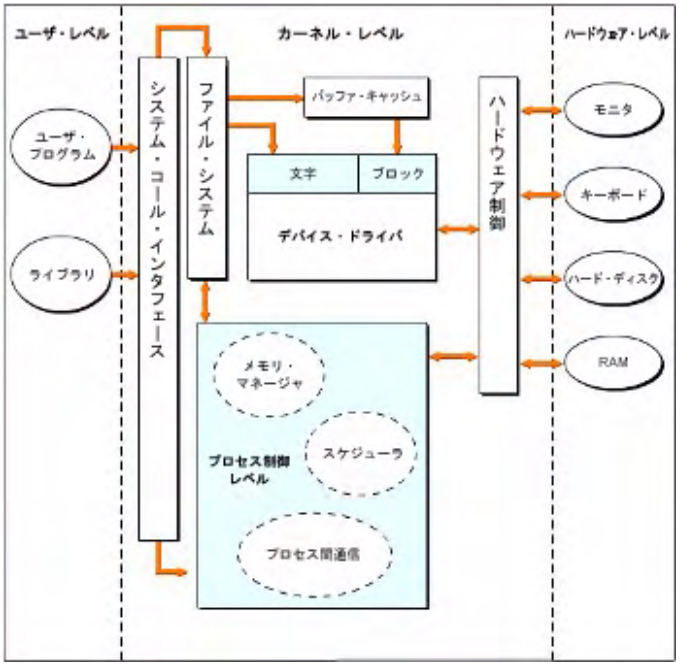
システムとアプリケーションは Windows ベースが圧倒的多数ですが，UNIX プラットフォームを基盤とするレガシー・アプリケーションや最新のアプリケーションも多数存在します。HP/UX，Sun Solaris，IBM AIX など高い実績を誇る UNIX フレーバに加えて，Linux の急速な普及により，安定性と拡張性の高さで定評ある UNIX でのアプリケーションの開発や移植が進みました。また UNIX/Linux は，Apache Web サーバ，WebSphere アプリケーション・サーバ，Oracle データベース・サーバを含めた J2EE ベース・システムの主要なプラットフォームになっています。

したがって，HP LoadRunner および HP Performance Center では，このようなテスト対象アプリケーションの動作を追跡できるように，UNIX オペレーティング・システム向けのパフォーマンス・カウンタが提供されています。

UNIX フレーバはコマンドとオプションに一部相違がありますが，パフォーマンス関連情報の収集，表示，再利用を簡単に実行できる各種機能が搭載されています。このような機能は，さまざまなサンプリング技術を使って定期的にパフォーマンス監視データを生成することにより，パフォーマンスの問題の診断に非常に役立つ情報を提供します。また，処理を効率化する設計が採用されているので，運用環境への影響を最小限にとどめながら継続的な実行が可能です。

# アーキテクチャ

次の図で示すように，UNIX オペレーティング・システムのアーキテクチャは，**ユーザ**，**カーネル**，**ハードウェア**という 3 つのレベルで構成されます。



**カーネル**・レベルは UNIX の中核であり，**ユーザ**・レベルと**ハードウェア**・レベルのインタフェースとして機能します。**カーネル**・レベルは，次に示すように，各種機能を持つプログラム群で構成されます。

- **システム・コール・インタフェース**：システム・コールを処理および実行します。プログラムは, システム･コールを使ってオペレーティング･システムに要求を行います。
- **ファイル・システム**：プロセス制御とシステム・コール間のインタフェースを調整し，文字データとブロック・データの入出力を処理します。データ I/O はデバイス・ドライバが処理します。
- **プロセス制御**：UNIX 内の各種プロセスを調整および制御します。プロセスとは，オペレーティング・システム上で実行中のプログラムです。このプログラムは，ユーザ・プログラムまたはシステム・プログラムのいずれかです。
- **ハードウェア制御**：キーボード，モニタ，ハードディスク，RAM などのハードウェア・デバイスを調整します。
- **デバイス・ドライバ**：ハードディスク，RAM，プリンタなどのシステム・デバイスの I/O を調整します。

**メモリ・マネージャ**は，UNIX アーキテクチャに不可欠な機能です。メモリ・マネージャは，UNIX で実行される各種プロセスに割り当てるメモリ容量を管理します。また，メモリ階層の管理も行います。メモリ階層は次のような構造を持っています。

- **バッファまたはキャッシュ・メモリ**：高速で高額な揮発性のメモリであり，容量は数キロバイト（KB）から数メガバイト（MB）です。
- **プライマリ・メモリまたはランダム・アクセス・メモリ（RAM）**：速度と価格が中程度の揮発性メイン・メモリであり，容量は数 MB から数 GB です。
- **セカンダリ・メモリまたはディスク・ストレージ**：低速で安価な非揮発性のディスクまたはテープ・ストレージであり，容量は GB 単位です。

**ユーザ・プログラム**と**システム・プログラム**はすべて**プロセス**と呼ばれます。プロセスの主な目的は，タスクを実行することです。システムは**プロセス識別番号（PID）**という一意の番号を各プロセスに割り当て，この番号を元にプロセス識別と管理を行います。システムは，PID を使って各プロセスに優先度を割り当てます。生成されたプロセスは，フォアグラウンドまたはバックグラウンドのいずれかで実行されます。バックグラウンドでプロセスを実行することにより，複数のプロセスの同時処理が可能になります。

## パフォーマンス・リソース

UNIX では，次に示す 7 つのリソース・タイプを監視およびチューニングする必要があります。

- CPU
- メモリ
- ディスク容量とアーム
- 通信回線
- I/O 時間
- ネットワーク時間
- アプリケーション・プログラム

### 総実行時間

総実行時間は，ユーザ・レベルでみると，時計の時間で測定されます。プロセス・レベルでは，**time** コマンドを実行して測定されます。これにより，実時間（時計）ユーザ・コード CPU とシステム・コード CPU を取得できます。**ユーザ・コードとシステム・コードの合計が 80% を超える**場合，CPU がボトルネックになっている可能性があります。

総実行時間は，次で構成されます。

- **ユーザ・ステート CPU**：ユーザ・ステートでのプログラム実行に実際に費やされた CPU 時間です。ライブラリ・コールにかかった時間も含まれますが，カーネル自体の処理にかかった時間は除外されます。この値は，コンパイル時の最適化やコードの効率化によって，大きく変動します。

- **システム・ステート CPU**：システム・ステートでのプログラム実行に実際に費やされた CPU 時間です。すべての I/O ルーチンにはカーネル・サービスが必要です。I/O 転送のブロック機能を使用することにより，この値を調整できます。
- **I/O 時間**：I/O 要求の処理にかかった時間です。
- **ネットワーク時間**：データの転送にかかった時間です。
- **仮想メモリ・パフォーマンス**：コンテキスト・スイッチやスワップが含まれます。
- **その他プログラムの実行にかかった時間**：別のアプリケーションが CPU を使用しているために，このアプリケーションを処理していない時間です。

### ツール

ほとんどの UNIX フレーバでは，オペレーティング・システムがプロセス実行中に統計情報を収集します。このような統計情報には，次に示す UNIX ツールでアクセスできます。

- **rstat**：カーネルから取得したパフォーマンス統計を返すサーバ/デーモン
- **netstat**：ネットワーク統計
- **nfsstat**：NTF 統計
- **time/timex**：プロセスの CPU 使用率
- **uptime**：システムの平均負荷
- **ps**：プロセス統計
- **iostat**：I/O 用のツール
- **sar**：バルク・システム・アクティビティ
- **vmstat**：仮想メモリ用ツール
- **prof**：プロセスのプロファイリング
- **trace**：詳細情報の取得

**uptime** は，最も便利なコマンドの 1 つです。スケジューリングの優先度は考慮されていませんが，システムのおおよその平均負荷を調べることができます。**uptime** を実行すると，過去 1 分，5 分，15 分の平均負荷が表示されます。

**uptime** 以外にも便利なコマンドがあります。**sar** コマンドを **-q** オプションを指定して実行すると，実行キューの平均長，実行キュー内で要求が待機している時間の割合，スワップ・キューの平均長，スワップ・キュー内で要求が待機している時間の割合が表示されます。実行キューには，メモリ内に存在していて実行可能なジョブが含まれますが，I/O の待機中またはスリープ状態のジョブは含まれません。実行キューのサイズは，**2** 未満が適正値です。

---

**注 :** UNIX フレーバには，簡単にパフォーマンスを監視できる機能が付属しています。たとえば Sun Solaris は，BSD ツールではなく，広く使用されている **rup** コマンドと **perfmeter** コマンドによって拡張されています。

---

## カウンタのタイプ

カウンタには，それぞれタイプがあります。カウンタ・タイプからパフォーマンス統計の収集方法がわかるので，タイプを把握しておくと便利です。次に，重要なカウンタのタイプを示します。

- **瞬間値を示すカウンタ**：最新の測定値を表示します。
- **時間間隔を示すカウンタ**：ある時間間隔におけるアクティビティの変化量を表示します。
- **経過時間を示すカウンタ**：時間間隔に渡って収集され，集計されません。
- **平均値を示すカウンタ**：所定の時間間隔に収集された値の平均値を表示します。

## HP ツールを使った UNIX の監視

UNIX のパフォーマンス情報は，Windows とは異なり，さまざまなプロセスが統計データを収集します。このようなプロセス（デーモン）には，常時実行されているものと，起動が必要なものがあります。

HP LoadRunner と HP Performance Center には，UNIX 環境向けの監視ソリューションが組み込まれています。このソリューションは **rstatd** デーモンを使用して稼働しますが，ほとんどのバージョンで設定済みの状態で提供されます。**rstatd** デーモンが設定済みかどうかを確認するには，**rup** コマンドを実行してください。**rstatd** を含め，マシン関連の各種統計データが表示されます。このデーモンが収集した統計データを元に，CPU 使用率，コンテキスト・スイッチ率，ディスク使用率など，よく使用されるカウンタを取得できます。詳細なパフォーマンス測定データが必要な場合は，上記の UNIX ツールをお勧めします。

コマンドと引数はフレーバごとに異なるため，HP SiteScope を LoadRunner や Performance Center と組み合わせて使用することをお勧めします。

HP SiteScope は，各種 UNIX フレーバを監視するアダプティブなインフラストラクチャなので，フレーバごとの相違点をなくすことができ，目的別にカウンタを分類します。ただしそのためには，監視対象の UNIX バージョンをサポートするアダプタ・ファイルを設定する必要があります。SiteScope は，UNIX オペレーティング・システムの各種バージョンを実行するサーバからさまざまなシステム・リソース情報を取得するために，アダプタ・ファイルを元にコマンドを記述します。

このコマンドは汎用コマンドですが，特定の UNIX バージョンの機能を拡張することもできます。また，UNIX の幅広いパフォーマンス情報の取得が可能です。次に，コマンドの一部を紹介します。

- **disk**：引数にディスクを指定すると，そのディスクの総容量，空き容量，使用率を返します。
- **disks**：システム上のファイル・システムのリストを返します。
- **memory**：スワップ領域について，使用済みと使用可能な容量を返します。
- **pageFault**：1 秒あたりのページ・フォルトの回数を返します。ページ・フォルトの行が複数発生した場合は加算されます。
- **cpu**：CPU が**待機**状態と**アイドル**状態になる割合を返します。
- **process**：長いプロセス名を持つプロセスのリストを返します。

SiteScope は，目的別（CPU，メモリ，I/O）にカウンタをグループ化し，各グループのインスタンスを対象にパフォーマンス・データを自動収集します。たとえば，CPU 使用率については，合計値とインストールされているプロセッサごとの値を提供し，ネットワークについては，インストールされているネットワーク・インタフェース・カードごとの統計値とネットワーク全体のスループットを提供します。通常は Windows 環境と UNIX 環境では別々の方法でパフォーマンス・テストを実行する必要がありますが，上記のようなアプローチを採用することによって 2 つの環境を論理的に統合できます。1 回のパフォーマンス・テストで対応できる場合もあるため，パフォーマンス・テストの担当者の作業負荷を軽減することができます。

## Processor - 最も重要なカウンタ

すべてのアプリケーションは，実行中にプロセッサ（CPU）リソースを消費します。プロセッサ・リソースに対する要求は**ユーザ状態**の処理と**システム状態**の処理に分けられます。

**ユーザ状態**の処理では，ユーザ状態でのユーザ・プログラムの実行に実際に費やされた CPU 時間が示されます。これにはライブラリ・コールにかかった時間も含まれますが，**カーネル**自体の処理にかかった時間は除外されます。

**システム状態**での処理では，プログラム実行に実際に費やされた CPU 時間が示されます。すべての I/O ルーチンには**カーネル**・サービスが必要です。

全体的な CPU 使用率（すべてのプロセッサの平均値）が **100%** に近付いたり，処理待ちのプロセスが常に存在するなど，CPU ボトルネックが発生しているかどうかは簡単に確認できます。これに比べて，CPU ボトルネックが発生している原因の特定は必ずしも簡単ではありません。したがって，通常のアプリケーションの動作を事前に把握し，これをベースラインにして作業負荷を分析することが非常に重要です。

次に，システム・レベルの監視に関連するカウンタを説明します。これは，特定のプロセスの動作ではなく，全体的なプロセッサ・パラメータに関するカウンタです。

| カウンタ | 説明 |
|---|---|
| **CPU Utilization** | タスクの実行にプロセッサが費やした時間の割合（パーセンテージ）。 |
| **User mode CPU Utilization** | ユーザ・モードでのコード実行にプロセッサが費やした経過時間の割合（パーセンテージ）。 |
| **System mode CPU Utilization** | システム・モードでのコード実行にプロセッサが費やした経過時間の割合（パーセンテージ）。 |
| **Average Load** | 過去 1 分間において，同時に **Ready** 状態になったプロセスの数の平均値。 |
| **Interrupt rate** | プロセッサが，サンプル間隔中にハードウェア割り込みの受信と処理に費やした時間。 |
| **Context switches rate** | プロセス間またはスレッド間の切り替えが発生した割合を，コンピュータ上にあるすべてのプロセッサについて合計した値。 |

### % CPU Utilization

| | |
|---|---|
| **正式名** | CPU Utilization カウンタ |
| **カウンタのタイプ** | サンプル間隔（ビジー状態の %） |
| **説明** | サンプル間隔内での全体的なプロセッサ使用率の平均値です。**アイドル・スレッド**の実行時間以外では，プロセッサは実際の作業負荷を処理するビジー状態だとみなされます。このカウンタの値は，**アイドル＋ユーザ＋システムの各使用率**の合計です（名称はプラットフォームによって異なります）。<br>ほとんどのプラットフォームにはアイドル CPU のカウンタ（関連するカウンタを参照してください）があるので，次の式を使って全体的な CPU 使用率を把握できます。<br>CPU 使用率＝ 100 －アイドル CPU（%） |
| **使用方法** | 全体的なプロセッサ使用率を示す主要な指標です。ビジー率は 0 ～100% の範囲で示されます。 |
| **パフォーマンス** | プロセッサが潜在的なボトルネックかどうかを判断する際の主要な指標です。 |
| **操作** | 使用率が 100% に近い状態が続く場合，ランナウェイ・プロセスである可能性があります。通常は，実行キュー（3 超）または優先度によってブロックされたプロセス（3 超）の値と組み合わせて判断します。<br>さらに詳細な調査が必要な場合は，ユーザ・モードの CPU 使用率カウンタを使って，ユーザ・プロセスまたは**カーネル**処理のどちらが消費しているのかを特定してください。 |
| **しきい値** | 応答型の作業負荷の場合，80 ～ 90% を超える使用率が続く場合には注意が必要です。 |
| **関連するカウンタ** | ➤ CPU Utilization¥%idle<br>➤ CPU Utilization¥%usr<br>➤ CPU Utilization¥%sys（Solaris）<br>➤ Processor¥Idle<br>➤ Processor¥Kernel（Linux）<br>➤ Processor¥%idle<br>➤ Processor¥%usr<br>➤ Processor¥%sys（AIX） |

**注：**マシン上で使用率が高いプロセッサが存在しても，対処が必要な問題が発生しているとは限りません。ただし，CPU アイドル時間が 20% 未満になった場合は調査が必要であり，10% 未満の場合はエラーが発生している可能性があります。

### User mode CPU Utilization

| | |
|---|---|
| **正式名** | User mode CPU Utilization |
| **カウンタのタイプ** | サンプル間隔（ビジー状態の %） |
| **説明** | サンプル間隔内で，ユーザ・モードでの処理に費やされた全体的なプロセッサ使用率の平均値です。つまり，この間，CPU はアプリケーション要求の処理でビジー状態であることを示します。 |
| **使用方法** | オペレーティング・システムがほとんどの時間をカーネル以外の処理に費やしている場合，システムは適正な状態であるといえます。ただし，CPU 時間は適切なプロセスの処理に費やし，重要性の低いアプリケーションを待機させる必要があります。 |
| **パフォーマンス** | なし |
| **操作** | プロセスがユーザ・モードのみで実行され，システム・コールや I/O を要求しない場合，無限ループに陥っている可能性があります。ユーザ・モードのプロセスで I/O 操作が大量に発生している場合は，メモリ・マッピングが実行されています。<br>一部のアプリケーションが CPU 時間をすべて消費し，テスト対象アプリケーションを処理できない状態になっている場合，このテスト対象アプリケーションは優先度でブロックされている可能性があります。 |
| **しきい値** | 50% を超える状態が続く場合，ボトルネックが発生していることを示します。 |
| **関連するカウンタ** | listed in CPU utilization |

### System mode CPU Utilization

| | |
|---|---|
| **正式名** | System mode CPU Utilization |
| **カウンタのタイプ** | サンプル間隔（ビジー状態の %） |
| **説明** | サンプル間隔内で，システム（**カーネル**）・モードでの処理に費やされた全体的なプロセッサ使用率の平均値です。オペレーティング・システム関数は，すべて**カーネル**・モードで実行されます。システム・モードには，デバイスの I/O 操作を開始するデバイス・ドライバ・コードや，割り込み処理を完了するための遅延プロシージャ呼び出しがあります。 |
| **使用方法** | ほとんどの場合，システム・モードでの CPU 利用率が高くなる原因は別にあります。<br>システム・モードの CPU 時間のほとんどはコンテキスト・スイッチで発生し，特に**カーネル**が実行するジョブの数が多すぎることが原因です。また，ディスク I/O やネットワーク帯域幅に問題があり，それが割り込み率（30% 超）の増加につながることも原因の 1 つです。メモリにも注意が必要であり，使用率が上限に達するとスワップが始まり，システムの処理速度が低下します。 |
| **パフォーマンス** | デバイス・ドライバ関数などのオペレーティング・システム関数が，潜在的なプロセッサ・ボトルネックになっているかどうかを判断する際の補助的な指標です。 |
| **操作** | コンテキスト・スイッチや I/O に問題がない場合，システム・コールに問題があります。30% を超える場合は，オペレーティング・システム・ツールを使って詳細な調査を行ってください。 |
| **しきい値** | 50% を超える状態が続く場合，ボトルネックが発生していることを示します。 |
| **関連するカウンタ** | listed in CPU utilization |

### Average Load

| | |
|---|---|
| **正式名** | Average Load or Run Queue |
| **カウンタのタイプ** | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| **説明** | プロセッサの Ready Queue 内で遅延状態であるとみなされ，実行まで待機しているプロセスの数です。プロセッサの Ready Queue で待機するスレッドは，プロセッサがアイドルになった時点で，優先度の高い順に処理されます。CPU ユニットの数は，Run Queue に影響を与えません。 |
| **使用方法** | 自発的に待機状態になるプログラム・スレッドは多数あります。したがって，アクティブ・スレッドのサブセットが，プロセッサ・キューの実質的な上限になります。 |
| **パフォーマンス** | プロセッサが潜在的なボトルネックかどうかを判断する際の補助的な指標です。 |
| **操作** | 容量の制限が原因でアプリケーションに過度な遅延が発生している可能性があることを示します。 |
| **しきい値** | 使用率が非常に高いプロセッサを 1 基のみ搭載するマシンでは，Average Load が 2 を超える状態が続く場合，プロセッサが処理可能なレベルを上回る作業負荷が発生している可能性があります。マルチプロセッサ環境では，Run Queue Length の値を物理プロセッサの数で割ってください。 |
| **関連するカウンタ** | ➤ CPU Utilization<br>➤ Queue length¥runq-sz（Solaris）<br>➤ Queue Statistics¥runq-sz（AIX） |

### Interrupt Rate

| | |
|---|---|
| **正式名** | Interrupt Rate |
| **カウンタのタイプ** | サンプル間隔（ビジー状態の %） |
| **説明** | サンプル間隔中に, 割り込みモードでの処理に費やされた全体的なプロセッサ使用率の平均値です。割り込みモードで実行されるのは，デバイス・ドライバ関数である割り込みサービス・ルーチン（ISR）のみです。 |
| **使用方法** | ISR が処理する割り込み処理は，優先度が最も高くなります。割り込み処理はシステム関数であり, 関連付けられたプロセスはありません。Interrupt Rate の値が大きい場合，デバイスの誤動作を示しますが，デバイスを特定することはできません。 |
| **パフォーマンス** | このカウンタは, ハードウェア割り込みの受信と処理に費やされたプロセッサ時間の割合をパーセンテージで示します。この値は，ネットワーク・アダプタなど割り込みを生成するデバイスの動作を間接的に示しています。カウンタの値が急増する場合は，ハードウェア障害が発生している可能性があります。 |
| **操作** | デバイスの誤動作が原因で潜在的なプロセッサ・ボトルネックが発生しているかどうかを判断する際の補助的な指標です。 |
| **しきい値** | このカウンタの値が 30% を超えた場合，注意が必要です。 |
| **関連するカウンタ** | System mode CPU Utilization |

### Context Switches Rate

| | |
|---|---|
| **正式名** | Context Switches Rate |
| **カウンタのタイプ** | サンプル間隔での差異（1 秒あたりの速度） |
| **説明** | 実行中のスレッドが別のスレッドに切り替わると，コンテキスト・スイッチが発生します。UNIX ではマルチスレッド操作がサポートされているので，コンテキスト・スイッチは正常な動作です。ユーザ・モードのスレッドが権限を持つオペレーティング・システム関数を呼び出すと，ユーザ・モード・スレッドと，呼び出された関数をシステム・モードで実行するカーネル・モード・スレッド間でコンテキスト・スイッチが発生します。 |
| **使用方法** | コンテキスト・スイッチは通常のシステム機能であり，作業負荷の結果としてコンテキスト・スイッチが発生します。コンテキスト・スイッチの回数が多くても，通常は問題が発生しているとは限らず，CPU 容量が上限に達したことを示すものでもありません。さらには，プロセスが CPU を占有できる時間の標準設定値を大きくするなど, チューニングが可能な特殊なシステム構成パラメータがある場合を除いて，コンテキスト・スイッチの発生回数をチューニングする方法はほぼありません。<br>過去の標準的なレベルを大きく上回った場合には, デバイスの誤動作などの問題が発生している可能性があります。 |
| **パフォーマンス** | コンテキスト・スイッチの回数が多いと，アプリケーション設計やスケーラビリティに潜在的な問題があることを示します。 |
| **操作** | コンテキスト・スイッチは，スレッドが実行中に，それよりも優先度の高いスレッドがプロセッサを横取りする場合や, 優先度の高いスレッドがブロックする場合に発生します。多くの場合，シェル・コマンドを使ったログインなど，生成と完了を頻繁に繰り返すプロセスが原因です。これは，多数のスレッドがシステム上のプロセッサを奪い合っている状態を示します。また，プロセッサ使用率があまり高くなく，コンテキスト・スイッチの回数が非常に低い場合には, スレッドがブロックされている可能性があります。 |

| | |
|---|---|
| **しきい値** | なし |
| **関連するカウンタ** | なし |

**注：**パフォーマンスを測定するには，サーバとサーバ・アプリケーションが起動状態であり，使用可能な状態になっている必要があります。

## プロセスの監視

UNIX は強力で非常に柔軟なオペレーティング・システムです。ユーザは，フォアグラウンドまたはバックグラウンドのいずれかで，必要に応じてプロセスを実行できます。フォアグラウンドで実行中のプログラムには，完全な読み取りおよび書き込みアクセスが与えられますが，バックグラウンドのプログラムには読み取りアクセスがありません。

パフォーマンス・カウンタを使用することにより，スレッドなどの実行可能単位がどの程度の CPU プロセス時間を消費しているのかを測定できます。また，測定したプロセッサ使用率から，どのアプリケーションがどの程度の CPU 時間を消費しているかがわかります。

すべての UNIX フレーバに共通の機能はありませんが，HP SiteScope の Process オブジェクトを使用することによって，プロセス/スレッドに関して次に示す統計データを取得できます（使用可能なカウンタはバージョンによって異なります）。

- **CPU**：選択したプロセスの CPU 使用率を，全体的な CPU 使用率に対するパーセンテージで示します。
- **MEMSIZE**：選択したプロセスが消費しているメモリ容量を示します。
- **PID**：オペレーティング・システムで登録されているプロセス ID を示します。
- **THREADS**：選択したプロセスがフォークしたスレッドの数を示します。
- **USER**：ユーザ・セッションの数を示します。

HP SiteScope ではプロセス監視に関して十分な情報を取得できない場合には，次のような組み込みの UNIX コマンドを実行できます。

- **ps :** 現在実行中のプロセスを示す静的リストを表示します。さらに，PID，使用済みのメモリ容量，プロセスの実行に使用したコマンド・ラインなど，プロセスの詳細情報も表示されます。ほとんどの場合，**-aux** 属性を指定して，ユーザ情報と端末プロセス以外の情報も取得することをお勧めします。
- **top :** 現在実行中のすべてのプロセスと，それぞれのメモリ消費量を示すリストを表示します。top コマンドの内容は数秒ごとに自動更新され，コンピュータ上のアクティブなプロセスの情報が表示されます。
- **proc ツール :** プロセスに関してさらに詳細な情報を取得します。このツールを実行するとプロセスの実行が一時中断されるため，実行には注意が必要です。proc ツールは **/var/proc** に格納されており，**pfiles**（アクティブ・プロセス），**pflags**（プロセスのステータス情報とフラグ），**pldd**（各プロセスにリンクされているすべての動的ライブラリ），**pmap**（プロセスのアドレス領域マップ），**psig**（シグナルとスレッド・ハンドラの動作）**prun**（プロセスの実行または開始），**pstack**（スタック・トレース），**pstop**（特定のプロセスの実行を一時停止）があります。

## Memory - 最も重要なカウンタ

UNIX では，物理（常駐）メモリと仮想メモリを管理します。オペレーティング・システムは，実際に使用可能なメモリ容量をアプリケーションには提示しないので，実際よりも大きな容量が使用可能容量として認識される傾向があります。UNIX では**仮想メモリ**という用語が使用されます。仮想メモリは，すべてのデータ用にプログラムが割り当てるメモリであり，具体的には共有メモリ，ヒープ領域，プログラム・テキスト，共有ライブラリ，メモリ・マップ・ファイルが含まれます。システム上のすべてのプロセスに割り当てられる仮想メモリの総容量は，予約されるスワップ領域の容量とほぼ等しくなります。仮想メモリ内にマッピングされたデータは必ずしも物理メモリ内でアクティブ（「常駐」）になるわけではないので，仮想メモリの容量と，実際に割り当てられる物理メモリの容量にはほとんど関連性がありません。プログラムの「メモリ不足」エラーは，一般的に，物理（常駐）メモリが不足しているのではなく，予約可能なスワップ領域（仮想メモリ）が不足していることが原因で発生します。

RAM 容量が不足すると，ディスクへのページングが過剰に発生してディスク帯域幅が大量に消費されるため，ディスク・パフォーマンスの低下が報告されます。したがって，ディスクへのページング率は，メモリ・パフォーマンスを示す重要な指標となります。

メモリ価格は比較的安価であるため，メモリ容量を増やすことによってすべての問題を解決できます。ただし，物理メモリの容量が大きくても仮想メモリ不足を補うことはできません。アプリケーションが使用した割り当てメモリが解放されないためにメモリ・リークが発生すると，致命的なクラッシュにつながる恐れがあります。UNIX システムでデータベース処理など大量のトランザクションを処理するアプリケーションを実行する場合，メモリを増設することでメモリ・キャッシュ内に格納できるデータ容量が増えるので，データベース・パフォーマンスが格段に向上することがあります。

使用可能な RAM 容量が不足している場合には，割り当てられた物理メモリの使用状況と，問題のあるプロセスの常駐ページ（常駐メモリ・セット）のサイズを把握することが重要です。

次の項では，よく使用されるカウンタを紹介しますが，これ以外にも**キャッシュ済み**メモリと**バッファ済み**メモリの使用率にも注意してください（Solaris および **HP-UX** では **%rcache/%wcache** と **bread/s bwrit/s**，Linux では **Cached** と **Buffers**）。これもメモリ消費量に含まれるので，空きメモリ容量が低下したからといってメモリ・リークが発生しているとは限りません。

**ヒント:** メモリ使用量が次の状態になった場合には注意が必要です。

- 一定期間，システム全体でのスワップ使用率が増大を続ける場合
- メモリ消費量は，次の式で計算できます。

**使用メモリ＝メモリ総容量ー（キャッシュ済み＋バッファ済み＋スワップ領域）**

- あるプロセスが原因で予約可能なスワップ領域が増大を続ける場合。このプロセスが原因でメモリ・リークが発生しています。

| カウンタ | 説明 |
|---|---|
| **Percent Used** | コンピュータ上で実行中のプロセスが使用可能な物理メモリの総容量。 |
| **MB Free** | 実行中のプロセスが使用可能なメモリの総容量。 |
| **Paging Rate** | ハード・ページ・フォルトを解決するために，ディスクに対して読み書きされる 1 秒あたりのページ数。 |
| **Page-in Rate** | 物理メモリに読み込まれた 1 秒あたりのページ数。 |
| **Page-out Rate** | ページ・ファイルに書き込まれ，物理メモリから削除された 1 秒あたりのページ数。 |

### Percent Used

| | |
|---|---|
| **正式名** | Percent Used |
| **カウンタのタイプ** | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| **説明** | ページ・フォルトが発生しない状態でアドレス可能な RAM 内の割り当て済みページの容量を, インストールされている総メモリ容量に対する割合（パーセンテージ）で示します。 |
| **使用方法** | メモリ使用量を示す主要な指標です。 |
| **パフォーマンス** | なし |
| **操作** | なし |
| **しきい値** | インストールされている RAM の 80% を超える状態が続く場合, メモリが不足していることを示します。90% に達すると, プロセスを実行できなくなる可能性があるので注意が必要です。 |
| **関連するカウンタ** | Memory¥freemem - in bytes（Solaris） |

### MB Free

| | |
|---|---|
| **正式名** | MB Free |
| **カウンタのタイプ** | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| **説明** | 空き仮想メモリの総容量をメガバイト単位で示します。 |
| **使用方法** | プロセスの実行に使用可能なメモリ容量を示します。 |
| **パフォーマンス** | なし |
| **操作** | なし |
| **しきい値** | なし |
| **関連するカウンタ** | Memory¥swap_free and Memory¥swap_avail - in bytes（Solaris） |

### Paging Rate

| | |
|---|---|
| **正式名** | Paging Rate or Pages/sec |
| **カウンタのタイプ** | サンプル間隔での差異（1 秒あたりの速度） |
| **説明** | サンプル間隔中に, ディスクへのページングが発生した回数を示します。Pages/sec は，Page-in/sec と Page-out/sec の合計です。 |
| **使用方法** | プログラムが仮想アドレスにアクセスしたときに, そのページが物理メモリ内にないと,「ページ・イン」が発生します。UNIX が物理メモリ内に空き領域を作る必要がある場合や，メモリ・マップ・ファイルの書き込みが発生した場合,「ページ・アウト」が発生します。ページ・アウトでは，常駐メモリ・セット全体がディスク・スワップ領域に転送されます。ページ・アウトが発生すると，プロセスは実行キューから削除されるので，CPU 時間は取得できません。 |
| **パフォーマンス** | 実メモリが潜在的なボトルネックかどうかを判断する際の主要な指標です。通常の場合，ページ・インを厳重に監視する必要はありませんが，ページ・アウトを監視することによってメモリ・ボトルネックを特定できることがあります。<br>また，過度に大きなファイル・システムのキャッシュ・バッファは，ページング率が高くなる原因の 1 つです。 |
| **操作** | 過剰なページングが応答時間の悪化につながる可能性を示します。 |
| **しきい値** | スワップ・デバイスあたりの Paging Rate が 50 を超える場合には注意が必要です。 |
| **関連するカウンタ** | ➤ Page-in Rate<br>➤ Page-out Rate |

**注：**

➤ 通常の場合，RAM を追加することによって過剰なページングを解消できます。ディスク帯域幅には上限があります。ページング操作に使用される RAM 容量は，他のアプリケーション・ファイル操作に使用できなくなります。

➤ スワップ・サイズの計算では，アプリケーションが要求すると考えられる容量を「予約可能」なスワップとして確保することをお勧めします。

### Page-in Rate

| | |
|---|---|
| **正式名** | Page-in Rate |
| **カウンタのタイプ** | サンプル間隔での差異（1 秒あたりの速度） |
| **説明** | このカウンタは，プロセスがアクセスするメモリの一部が仮想メモリ内にあり，実行するには物理メモリに読み込む必要があることを示します。このカウンタは読み取り操作の回数を示し，1 回の操作で取得されたページ数には関係しません。値が大きいほど，メモリ・ボトルネックが発生している可能性が高くなります。 |
| **使用方法** | このカウンタは，Page-out カウンタよりも重要度は高くありません。予想外のレベルに増大する場合を除き，特に監視する必要はありません。 |
| **パフォーマンス** | 実メモリが潜在的なボトルネックかどうかを判断する際の補助的な指標です。 |
| **操作** | 過剰なページングが応答時間の悪化につながる可能性を示します。 |
| **しきい値** | なし |
| **関連するカウンタ** | Paging Rate |

### Page-out Rate

| | |
|---|---|
| **正式名** | Page-out Rate |
| **カウンタのタイプ** | サンプル間隔での差異（1 秒あたりの速度） |
| **説明** | このカウンタは，プロセスの常駐メモリ・セットが物理メモリより大きすぎるため，ディスクへのページングが発生していることを示します。このカウンタは読み取り操作の回数を示し，1 回の操作で取得されたページ数には関係しません。値が大きいほど，メモリ・ボトルネックが発生している可能性が高くなります。 |
| **使用方法** | ページアウト操作の回数が少なく，物理メモリの操作回数が多い場合，ディスク・ボトルネックが発生している可能性があります。また，ページアウト回数が減ってもキューの長さが増大しない場合は，メモリが不足しています。 |
| **パフォーマンス** | 実メモリが潜在的なボトルネックかどうかを判断する際の主要な指標です。 |
| **操作** | 過剰なページングが応答時間の悪化につながる可能性を示します。 |
| **しきい値** | スワップ・デバイスあたりの Paging Rate が 50 を超える場合には注意が必要です。 |
| **関連するカウンタ** | Paging Rate |

# I/O - 最も重要なカウンタ

UNIX では，I/O マネージャが物理および論理ディスク操作を管理します。**論理ボリューム**は，一意のドライブ文字で示される 1 つのファイル・システムです。**物理（未加工）ディスク**はストレージ・デバイスを内部的に示したものであり，SCSI，RAID，SATA などがあります。

アレイ・コントローラや RAID など複雑なストレージ・システムでは，それを構成する物理ディスク・ハードウェアの属性をオペレーティング・システムが直接認識できない場合があります。このような属性には，ディスクの数，ディスクの速度，シーク時間，回転速度，ビット密度，最適化機能（オンボードのメモリ・バッファなど）があり，パフォーマンスに大きな影響を与える可能性があります。また，メモリ・バッファやコマンド・キューなど高度な機能が搭載されていると，パフォーマンスが 25～50% 向上することがあります。

ディスク・パフォーマンスは，特にディスク・ページングが発生すると急激に低下する傾向があるので，プロアクティブな対策を講じることが重要です。

---

**注：**

- 一般的に，少数の大容量ディスクよりも，小さい容量のディスクを多数使用する方が柔軟な構成が可能になり，I/O ボトルネックを軽減できる点でメリットがあります。使用率の高い論理ボリュームを複数のディスクと I/O チャネルに分割する方法もお勧めします。
- アプリケーションのディレクトリ・パスを検討する際には，ファイル・システムのルートからの階層数を最小限に抑えるようにしてください。ディレクトリ・ツリーの階層が深すぎると，ファイル・アクセスに伴うルックアップの回数が増え，パフォーマンス低下を招きます。反対に，1 つのディレクトリ内にあるファイル数が多すぎると（数千単位），ファイル・アクセス速度が低下します。

---

I/O 処理が大量に発生するトランザクション・アプリケーションでは，ファイル・システムではなく未加工デバイスを使用する方がパフォーマンスが向上します。この構成は，Oracle を含むほとんどのデータベース・ベンダが推奨しています。ただし，高機能を備えた新しい論理ボリューム管理ツールを使用することにより，ファイル・システム・デバイスでも未加工ボリュームに匹敵するパフォーマンスを発揮できるようになります。いずれの場合も，相互に悪影響が及ばないように，アプリケーションを個別に物理ディスクに割り当てることをお勧めします。

| カウンタ | 説明 |
|---|---|
| **%Used** | マウントされた各ファイル・システムで使用可能な容量の割合。 |
| **Free** | マウントされた各ファイル・システム使用可能なバイト数。 |
| **Disk Rate** | 物理ディスクが潜在的なボトルネックになっているかどうかの指標。 |

### %Used

| | |
|---|---|
| **正式名** | Filesystsems(n)¥%Used |
| **カウンタのタイプ** | サンプル間隔（%） |
| **説明** | 現在のファイル・システムのディスク使用率を，総容量に対するパーセンテージで示します。 |
| **使用方法** | 物理ディスクの I/O パフォーマンスを示す主要な指標です。パフォーマンスはディスク構成によって左右され，ディスク構成はオペレーティング・システムで認識されます。シーク時間，回転速度，記録密度，インタフェース速度などのパフォーマンス属性は，ディスクごとに異なります。高額な高性能ディスクでは，パフォーマンスが 50% 向上することがあります。 |
| **パフォーマンス** | ディスクが潜在的なボトルネックかどうかを判断する際の主要な指標です。 |
| **操作** | ディスク応答時間の悪化は，アプリケーションの応答時間の悪化につながります。 |
| **しきい値** | この値が 90% に近付いたら注意が必要です。95% を超えた場合，エラーが発生していることを示します。 |
| **関連するカウンタ** | ➤ Filesystsems(n)¥Use%（Linux）<br>➤ Filesystsems(n)¥used - in bytes（Solaris） |

### Free

| | |
|---|---|
| **正式名** | Filesystsems(n)¥Free |
| **カウンタのタイプ** | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| **説明** | 論理ディスク上の未割り当て領域の容量をバイト単位で示します。非常に容量の大きなファイル・システムでは，Free Megabytes カウンタの計算に時間がかかるため，I/O マネージャの測定レイヤは約 5 分間隔でカウンタの値を再計算します。<br>ディスク使用率の計画を行う際の主要な指標です。ディスク容量カウンタを使用できない場合（一部の UNIX フレーバでは提供されていません），この値と全体のディスク容量を元に使用率を計算できます。 |
| **使用方法** | 使用済みの論理ディスク容量を示す主要な指標です。 |
| **パフォーマンス** | なし |
| **操作** | 一般的に，ファイル・システム容量の不足は致命的な状況を招きます。 |
| **しきい値** | なし |
| **関連するカウンタ** | ➤ Filesystsems(n)¥Available<br>➤ Filesystsems(n)¥Used（Linux）<br>➤ Filesystsems(n)¥avail<br>➤ Filesystsems(n)¥used<br>➤ Filesystsems(n)¥capacity（Solaris） |

### Disk Rate

| | |
|---|---|
| **正式名** | Filesystsems(n)¥Disk Rate |
| **カウンタのタイプ** | サンプル間隔での差異（1 秒あたりの速度） |
| **説明** | サンプル間隔中に完了した物理ディスク要求の数を示します。 |
| **使用方法** | 物理ディスクの I/O 操作を示す主要な指標です。また，ディスク到着回数とも呼ばれます。 |
| **パフォーマンス** | ディスクが潜在的なボトルネックかどうかを判断する際の主要な指標です。 |
| **操作** | ディスク応答時間の悪化は, アプリケーションの応答時間の悪化につながります。 |
| **しきい値** | 基盤となるディスク・ハードウェアによって変動します。 |
| **関連するカウンタ** | なし |

---

**ヒント**：一般的に，I/O スループットを改善するには次のような方法があります。

- ディスク I/O をできるだけ分散します。使用率 100% のディスクを 1 台稼働するよりも，使用率が 10% のディスク 10 台を稼働する構成をお勧めします。
- ログ処理が過度に発生しないようにします。一部のアプリケーションでは，ログの詳細レベルを設定できます。
- SCSI デバイスをチューニングします。デバイスのキューの最大長を調整できることがあります。通常，これによってハードウェアが過負荷状態になる可能性がありますが，並列性は向上します。

---

**注 :** ディスクについては，次の内容を考慮してください。

- I/O が小さくなるほど，サービス時間も短縮します。I/O が大きくなるほど，サービス時間も長くなります。
- シーケンシャル I/O では，ヘッドの移動が少ないので，ランダム I/O よりも処理が高速になります。
- I/O サイズが大きくなるほど，シーケンシャル I/O で最大スループットを発揮できます。
- ファイル・システム・ブロック，バッファ・チェーン，ファイル・エクステントなどシステム境界をまたいだ処理を行うと，I/O 要求は複数に分割されることがあります。
- 最も使用率の高いディスクがスワップ・デバイスの場合，メモリ・ボトルネックが発生しているにも関わらず，ディスクに問題があるように見えることがあります。このような場合は，まずメモリの問題を解決する必要があります。

## Network - 最も重要なカウンタ

分散アプリケーションやクラウド・アプリケーションの普及に伴い，ネットワーク・パフォーマンスの重要性はこれまでよりもさらに高まっています。ただし UNIX オペレーティング・システムでは，最も低いレベルのハードウェア・インタフェースから，それよりも高いレベルのネットワーク・プロトコル（TCP/IP など）までさまざまなレベルで統計データが収集されますが，データの種類は限られています。ネットワーク・インタフェースの統計データは，ネットワーク・インタフェース・ドライバ層に組み込まれているソフトウェアによって収集されます。このソフトウェアは，送受信されるパケットの数をカウントします。

UNIX 機能を使って収集されるネットワーク統計には，**netstat**，**netperf**，**iozone**，**nfsstat**（NFS の監視）があり，インストールされているネットワーク・インタフェース・チップまたはカードが対象になります。Network Node Manager や SiteScope などの HP 製品は，一定期間に統計データを収集し，パフォーマンス・ボトルネックの本当の原因を究明します。

ネットワーク・ボトルネックの捕捉や分析は難しい作業です。パケット数，衝突回数，エラー回数だけでは，問題の原因を特定することはできません。

- 過剰な衝突が発生している場合にのみ，ネットワーク・ボトルネックが発生している可能性があり，衝突回数が比較的低い場合には正常稼働状態だと考えられます。全二重設定や速度設定のいずれかに矛盾があると衝突が発生しますが，これはエラーの原因になります。この問題を解消すると，衝突回数が低減し，パフォーマンスが改善されます。
- また，パケット数が急増し，さらにネットワーク出力キュー内の要求も増えている場合も，ネットワーク・ボトルネックが発生している可能性があります。ただし，適切な情報に基づいて判断するには，時間経過にともなうパターンを調査する必要があります。
- NFS の使用率が高い場合，**nfsstat** で収集したデータを監視する必要があり，特にサーバ側で監視してください。NFS 統計から，1 つのクライアントに処理が集中していることがわかった場合は，そのクライアント・ホストでツールを実行して問題になっているプロセスを特定します。
- システム・モードで CPU 使用率が高い場合や，1 つのプロセスの割り込み率のみが高く他のプロセスはほとんどアイドル状態の場合は，ネットワーク・ボトルネックが発生している可能性があります。デバイス設定をチェックしてください。ハードウェアが原因になっていることがあります。

| カウンタ | 説明 |
|---|---|
| **Incoming packets rate** | NIC が 1 秒間に受信する Ethernet パケットの数。 |
| **Outgoing packets rate** | NIC が 1 秒間に送信する Ethernet パケットの数。 |
| **Incoming packets error rate** | NIC が受信する Ethernet パケットの中で，1 秒間に発生するエラーの数。 |
| **Outgoing packets error rate** | NIC が送信する Ethernet パケットの中で，1 秒間に発生するエラーの数。 |
| **Collision rate** | ネットワーク衝突の数。 |

### Incoming Packets Rate

| | |
|---|---|
| **正式名** | Incoming Packets Rate |
| **カウンタのタイプ** | サンプル間隔での差異（1 秒あたりの速度） |
| **説明** | サンプル間隔内において，このインタフェースで 1 秒あたりに受信されたバイト数の合計です。 |
| **使用方法** | ネットワーク・インタフェース・トラフィックを示す主要な指標であり，Outgoing Packets Rate と組み合わせて使用します。 |
| **パフォーマンス** | ネットワークが潜在的なボトルネックかどうかを判断する際の主要な指標です。 |
| **しきい値** | Incoming Packets Rate が回線容量の 40% を超えた場合，注意が必要です。 |
| **関連するカウンタ** | Outgoing Packets Rate |

### Outgoing Packets Rate

| | |
|---|---|
| **正式名** | Outgoing Packets Rate |
| **カウンタのタイプ** | サンプル間隔での差異（1 秒あたりの速度） |
| **説明** | サンプル間隔内において，このインタフェースで 1 秒あたりに送信されたバイト数の合計です。 |
| **使用方法** | ネットワーク・インタフェース・トラフィックを示す主要な指標であり，Incoming Packets Rate と組み合わせて使用します。 |
| **パフォーマンス** | ネットワークが潜在的なボトルネックかどうかを判断する際の主要な指標です。 |
| **しきい値** | Outgoing Packets Rate が回線容量の 40% を超えた場合，注意が必要です。 |
| **関連するカウンタ** | Incoming Packets Rate |

---

**注：** この 2 つのカウンタは，このインタフェースでのスループット（バイト数）を示します。ネットワーク・アダプタのトラフィックが飽和状態かどうか，ネットワーク・アダプタの追加が必要かどうかを判断する指標になります。

---

### Incoming Packets Error Rate

| | |
|---|---|
| **正式名** | Incoming Packets Error Rate |
| **カウンタのタイプ** | サンプル間隔での差異（1 秒あたりの速度） |
| **説明** | サンプル間隔内において，このインタフェースで 1 秒あたりに受信したエラー数の合計です。 |
| **使用方法** | ネットワーク・インタフェース・トラフィックを示す補助的な指標であり，Outgoing Packets Error Rate と組み合わせて使用します。 |
| **パフォーマンス** | ネットワークが潜在的なボトルネックかどうかを判断する際の補助的な指標です。通常は，全二重設定と速度設定に矛盾があることが原因で発生します。 |
| **しきい値** | Incoming Packets Error Rate が示す 1 秒あたりのエラー数が 0.025 を超える場合，注意が必要です。 |
| **関連するカウンタ** | Outgoing Packets Error Rate |

### Outgoing Packets Error Rate

| | |
|---|---|
| **正式名** | Outgoing Packets Error Rate |
| **カウンタのタイプ** | サンプル間隔での差異（1 秒あたりの速度） |
| **説明** | サンプル間隔内において，このインタフェースで 1 秒あたりに送信したエラー数の合計です。 |
| **使用方法** | ネットワーク・インタフェース・トラフィックを示す補助的な指標であり，Incoming Packets Error Rate と組み合わせて使用します。 |
| **パフォーマンス** | ネットワークが潜在的なボトルネックかどうかを判断する際の補助的な指標です。通常は，全二重設定と速度設定に矛盾があることが原因で発生します。 |
| **しきい値** | Outgoing Packets Error Rate が示す 1 秒あたりのエラー数が 0.025 を超える場合，注意が必要です。 |
| **関連するカウンタ** | Incoming Packets Error Rate |

**ヒント：**この 2 つのカウンタは，ネットワーク品質をトラッキングします。所定のしきい値を超えた場合は，ネットワーク・ハードウェアを調査してください。

### Collision Rate

| | |
|---|---|
| **正式名** | Collision Rate |
| **カウンタのタイプ** | サンプル間隔での差異（1 秒あたりの速度） |
| **説明** | インタフェースで発生しているエラーの数を 1 秒あたりの数値で示します。 |
| **使用方法** | このカウンタは，ネットワークを介して送受信されたエラー数を示します。値が大きい場合，ネットワーク帯域幅がボトルネックになっていることを示します。一般的な原因としては，ハードウェアの圧縮の問題や，物理的なコネクタ / 終端の問題が考えられます。 |
| **パフォーマンス** | ネットワークが潜在的なボトルネックかどうかを判断する際の主要な指標です。しきい値を超える場合，そのセグメントでネットワークが過負荷状態であることを示すので，ネットワーク・トポロジを再検討することをお勧めします。 |
| **しきい値** | 10% 以下が適正値です。 |
| **関連するカウンタ** | なし |

# 第III部

## ランタイム・プラットフォーム

# 第5章

# ランタイム・プラットフォームの監視

本章では，ランタイム・プラットフォームの監視の概要と，必要になる J2EE と .NET アプリケーション・アーキテクチャについて説明します。

**本章の内容**



## 概要

一般的にアプリケーションは特定のオペレーティング・システム用に開発されます。したがって，オペレーティング・システムに影響を与える要因は，アプリケーションのパフォーマンスにも影響を与えます。それぞれのオペレーティング・システムは，パフォーマンスの監視とチューニングを行う専用のパフォーマンス・パラメータを備えています。

また，アーキテクチャ・レベルでの監視やチューニングもアプリケーションのパフォーマンスに影響を与えます。ただし，アーキテクチャの設計は特定のテクノロジを基盤としているので，パフォーマンスを向上するには，テクノロジ・レベルで監視とチューニングを行う必要があります。以上のような目的をすべて達成するには，監視およびチューニングのさまざまな段階を考慮して適切なガイドラインを作成する必要があります。

汎用や独自仕様などさまざまなテクノロジが存在する中，エンタープライズ・アプリケーションの開発には Java 2 Enterprise Edition（J2EE）またはこれに相当する Microsoft 環境である .NET Framework が使用されます。これによって，従来よりもビジネス・ソリューションの開発期間を短縮でき，高い機能性や堅牢性が実現されています。

このようなソリューションの設計は簡単ではなく，多彩な機能を使いこなす必要もあるため，品質の低いアプリケーションが開発されてしまう可能性も高くなります。開発環境や QA 環境では問題がなくても，運用環境へ移行するとスケーラビリティやパフォーマンスに問題が発生することがあります。

したがって，アプリケーションが稼働するインフラストラクチャがどのような影響を及ぼすか，作業負荷を適用したときに多くのアプリケーション・コンポーネントがどのように動作するかを理解することが必要です。

Web 対応の J2EE および .NET アプリケーションでは，開発期間短縮ニーズの高まりを背景に，デプロイメント・ライフサイクルも短縮しています。その結果，開発，QA，デプロイメント，製品化の各フェーズを区切る境界線が明確でなくなり，各フェーズを担当する IT グループもあいまいになっています。中心となる IT 組織は，十分な知識がないまま何百ものアプリケーションを管理している状態であり, J2EE に関する IT スタッフのスキルも十分とはいえません。

アプリケーションの開発では，設計や使用パターンを細部まで検討し，明確なサービス目標で高いパフォーマンスを発揮するためのプランニングやテストを実施するなど，パフォーマンスとスケーラビリティを十分に考慮する必要がありますが，このような方法で設計されていないアプリケーションが多数存在します。J2EE は高いスケーラビリティを備えてはいますが，不十分な設計を補うことはできません。また，.NET 構成も同様です。バッファ，セッション・タイムアウト，アプリケーションの保護レベル，ログなどの設定が不適切であると, 負荷が大きい環境では .NET アプリケーションに影響を及ぼす可能性があります。

# アーキテクチャ

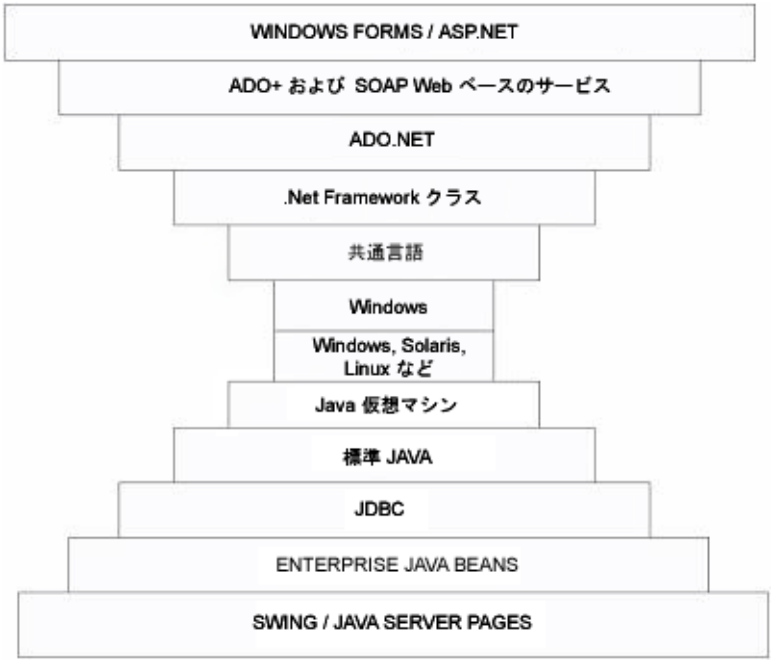
J2EE または .NET アプリケーションを実行する際には，その実行環境であるオペレーティング・システムの各種パラメータを設定することでパフォーマンスを最適化できます。このようなパラメータは，さまざまなカウンタを使用して監視および測定されます。パフォーマンスの視点からオペレーティング・システムをチューニングする作業で役立つカウンタは，テスト結果の分析において非常に重要な役割を果たします。

ほとんどのアプリケーションは Windows と UNIX のいずれかのオペレーティング・システムで実行されるため，次の章ではこの 2 つのオペレーティング・システムに関する重要なカウンタについて説明します。

UNIX で監視とチューニングが必要になる主なリソースには，CPU，メモリ，ディスク容量，通信回線，I/O 時間，ネットワーク時間，アプリケーション・プログラムがあります。UNIX オペレーティング・システムでは，システム・リソースとその使用率を追跡するカウンタがいくつか用意されています。このカウンタは，CPU 使用率，バッファ使用率，ディスク I/O，テープ I/O，端末，システム・コール，コンテキスト・スイッチ，ファイル・アクセス使用率，キュー，プロセス間通信（IPC），ページング，空きメモリ容量とスワップ容量，カーネル・メモリ割り当て（KMA）などの操作を追跡します。詳細については，第4章「Unix の監視」を参照してください。

Windows は，**チューニングを自動で行う**オペレーティング・システムです。つまり，Windows ではほとんどの場合，ハードウェアが正しく設定されていることが前提となり，稼働環境に合わせてパフォーマンスの最適化が自動的に行われます。たとえば，Windows を Web サーバとしてデプロイすると，それ以外のサービスで稼働中でないものは，CPU やメモリなどのシステム・リソースをほとんど使用しない状態になります。ただしパフォーマンスは，他のオペレーティング・システムと同様に，ハードウェア，デバイス・ドライバ，アプリケーション，作業負荷，ネットワークなどさまざまな外的要因の影響を受けます。詳細については，第3章「Windows の監視」を参照してください。

J2EE と .NET はいずれも，アプリケーション開発を始める前にアプリケーション・アーキテクチャを定義しておく必要があります。これは，それぞれが独自のフレームワークでアーキテクチャを定義するためです。ただし，この 2 つのアーキテクチャにはシステムを定義する上で共通の特徴もあるので，ここでは，パフォーマンス・カウンタの監視およびアプリケーションのチューニング向けの共通のガイドラインを作成します。J2EE と Microsoft .NET は，いずれも一般的な標準規格を基盤としています。また，次の図で示すように，複数の階層から成るアーキテクチャが採用されています。アプリケーションは異なる論理層に実装されるので，内部構造（ビジネス・ロジックやデータ管理）には依存しないプレゼンテーションが可能です。



- J2EE と .NET アーキテクチャ・モデルはいずれもオブジェクト指向（OO）のアプローチを採用することで，エンタープライズ・コンピューティングを合理化します。このアプローチでは，強力な OO フレームワーク（クラス・ライブラリ）を使用して，エンタープライズ・コンポーネントの管理，オブジェクトの永続化，トランザクション，Web サービス，非同期通信，疎結合イベント・サービス，メッセージ送受信などのサービスを実行します。
- また，仮想マシン（VM）アーキテクチャも J2EE と .NET に共通の機能です。アプリケーション開発ツールを使用することで，プラットフォーム固有のバイナリ・コードではなく，中間レベルのコードを生成できます。これは，VM がコードをリアルタイムで解釈，つまり Just-In-Time（JIT）コンパイルを実行することを意味します。
- J2EE と .NET の基盤となるアーキテクチャには共通点が多く，階層型の設計が採用されています。

QA フェーズでは，一般的に，パフォーマンス・テストの後に機能テストと回帰テストをまとめて実行します。ソフトウェアをリリースする前に，外部システムとのインタフェースを含めたアプリケーション全体でパフォーマンス・テストを実施する必要があります。

その目的は，実際の使用環境に基づく作業負荷で発揮できるスケーラビリティと容量を予測し，アプリケーションのパフォーマンス上の動作を把握して，ボトルネック解消に役立つデータを収集することにあります。このデータには，J2EE/.NET の各層とメソッドごとのトランザクション遅延時間の内訳や，根本原因の詳細な診断情報が含まれます。

# 第6章

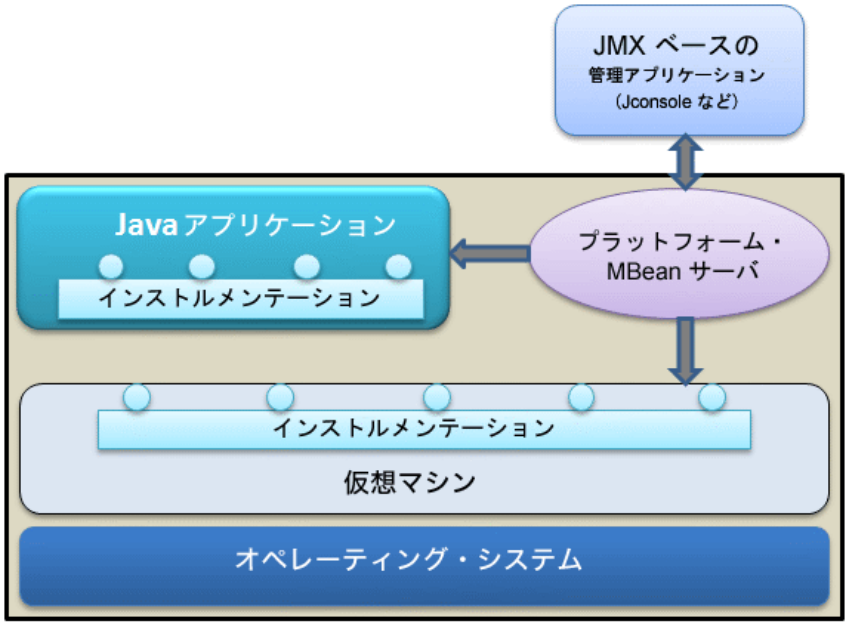
# Java プラットフォームの監視

本章では，Java プラットフォームの監視に関するベスト・プラクティスを紹介します。

**本章の内容**

# 概要

Java 2 プラットフォームは，監視と管理を行う包括的な機能を備えています。Java 仮想マシン（JVM）向けの管理インタフェースを定義する機能だけでなく，すぐに使用できる Java プラットフォームとそこで実行されるアプリケーション向けのリモート監視および管理機能が提供されています。



さらに JDK 5.0 では，Java Monitoring and Management Console ツール（**JConsole**）が付属します。JDK 5.0 は JVM の幅広いインストルメンテーションを使用し，Java プラットフォーム上で実行されるアプリケーションのパフォーマンスやリソース消費に関する情報を Java Management Extension（JMX）テクノロジを使って提供します。JMX は，標準的な方法で Java Runtime Environment とアプリケーションのインストルメンテーションを行います。インストルメンテーションには JMX Managed Bean（MBean）インタフェースを使ってアクセスします。MBean をプラットフォームの MBean サーバに登録しておく必要があります。また，アプリケーションは独自の MBean を作成してプラットフォームの MBean サーバに登録することもできます。これにより，サーバが一元的なリモート・アクセスのポイントになります。JConsole など JMX 準拠のクライアントはプラットフォーム MBean サーバに接続し，JMX テクノロジを使ってアプリケーション（および Java プラットフォーム）の管理を実行できます。各プラットフォーム MBean では，さまざまな属性と操作（メモリ使用率，スレッド CPU 使用率，ガーベジ・コレクションなどの統計）が用意されています。

HP SiteScope では，JMX のサポート機能が組み込まれているので JConsole を使用する必要がなく，オペレーティング・システム・カウンタと Java 固有のアプリケーション統計をまとめて表示することができます。また HP SiteScope では，JConsole で提供されるすべてのカウンタにアクセスできます。

# 最も重要な Java カウンタ

| | カウンタ | 説明 |
|---|---|---|
| 全般 | **Uptime** | JVM の稼働時間。 |
| | **Total compile time** | Just-In-Time（JIT）コンパイルに費やされた時間。 |
| | **Process CPU time** | JVM が使用した CPU 時間の合計。 |
| メモリ | **Current heap size** | 現在ヒープが占有しているサイズ（KB 単位）。 |
| | **Maximum heap size** | ヒープが占有する最大サイズ（KB 単位）。 |
| | **Committed memory** | ヒープ用に割り当てられているメモリ容量の合計。 |
| | **GC time** | ガーベジ・コレクションに費やした累計時間と呼び出し回数。 |
| スレッド | **Live threads** | 稼働中のデーモン・スレッドとデーモン以外のスレッドの現在の数。 |
| | **Peak threads** | JVM を起動して以降のライブ・スレッドの最大数。 |
| | **Daemon threads** | ライブ・デーモン・スレッドの現在の数。 |
| | **Total started threads** | JVM の起動以降に開始されたスレッドの総数（デーモン，デーモン以外，終了したスレッドを含む）。 |
| クラス | **Current classes loaded** | 現在メモリにロードされているクラスの数。 |
| | **Total classes loaded** | JVM の起動以降にメモリにロードされたクラスの総数（ロード後にアンロードされたクラスも含む）。 |
| | **Total classes unloaded** | JVM の起動以降にメモリからアンロードされたクラスの数。 |

## 全般的なカウンタ

本項では，マシン上で稼働する JVM に関して全般的な情報を提供するカウンタを説明します。

### Uptime

| | |
|---|---|
| **正式名** | Uptime |
| **カウンタのタイプ** | 経過時間 |
| **説明** | マシンで JVM を起動してから経過した時間です。 |
| **使用方法** | Java の全体的なステータスを示します。 |
| **パフォーマンス** | 全体的な稼働状態を示す重要な指標です。 |
| **操作** | ガーベジ・コレクションの頻度が低いと，JVM の稼働時間が長くなるほど，開いた状態のスレッド数が増大します。 |
| **しきい値** | なし |

## Total compile time

| | |
|---|---|
| **正式名** | Total compile time |
| **カウンタのタイプ** | 経過時間 |
| **説明** | Just-In-Time（JIT）コンパイルに費やされた時間です。JIT コンパイルのタイミングは，JVM 実装によって異なります。 |
| **使用方法** | JVM は Java をバイトコードに変換するので，ロード時にオブジェクトをコンパイルする必要があります。このカウンタは，JVM の起動以降，このようなコンパイルに費やされた時間の合計を示します。Sun の Hotspot VM は，アダプティブ・コンパイルを採用しています。このコンパイルでは，VM は標準インタープリタを使ってアプリケーションを起動した後，コードを分析してオブジェクトのパフォーマンス・ボトルネック（「ホット・スポット」と呼ばれます）を検出します。 |
| **パフォーマンス** | 新しいオブジェクトを多数追加した場合に，ボトルネックになる可能性があるかどうかを判断する際の補助的な指標です。 |
| **操作** | このカウンタにより，システムが正しい方法でデプロイおよび起動されているかどうかを特定できます。 |
| **しきい値** | なし |

### Process CPU time

| | |
|---|---|
| **正式名** | Process CPU time |
| **カウンタのタイプ** | 経過時間 |
| **説明** | JVM が消費した CPU 時間の合計を示します。 |
| **使用方法** | JVM がオペレーティング・システム全体の動作に与える影響を把握する主要な指標の 1 つです。 |
| **パフォーマンス** | この値から, Java プロセスとその他のプロセスで消費されたプロセッサ時間の割合を計算できます。カウンタの値が急増する場合は，問題が発生している可能性があります。 |
| **操作** | このカウンタから，JVM のスケールアップにどのような変更が必要なのかを特定できます。 |
| **しきい値** | プロセッサによって異なります。 |

## メモリ・カウンタ

本項では，JConsole の［メモリ］タブで通常表示されるカウンタについて説明します。このタブでは，メモリ消費量，メモリ・プール，ガーベジ・コレクションなどの統計が表示されます。

使用可能なメモリ・プールは，使用する JVM によって異なります。Sun Java の標準インストールに付属する HotSpot 仮想マシンでのプールを次に示します。

- **Eden 領域（ヒープ）プール**：このプールから，ほとんどのオブジェクトに対して最初にメモリが割り当てられます。
- **Survivor 領域（ヒープ）プール**：Eden 領域プールのガーベジ・コレクションで残ったオブジェクトが含まれます。
- **Tenured Generation（ヒープ）プール**：Survivor 領域プールである程度の期間存続していたオブジェクトが含まれます。

- **Permanent Generation（非ヒープ）プール**：クラスやメソッド・オブジェクトなど，仮想マシン自体を反映するデータがすべて保持されます。クラス・データ共有を使用する JVM では，このプールは読み取り専用領域と読み取り/書き込みの領域に分割されています。
- **Code Cache（非ヒープ）プール**：HotSpot JVM には「コード・キャッシュ」も含まれます。これには，ネイティブ・コードのコンパイルと保存に使用されるメモリが含まれています。

メモリ・プールには，メモリ不足の検出用に，**使用率のしきい値**と**コレクション使用率のしきい値**の 2 種類が用意されています。一部のメモリ・プールでは，いずれか一方がサポートされていません。

- **使用率のしきい値**：メモリ・プールの管理属性です。小さなオーバーヘッドでメモリ使用率を監視できます。しきい値を正の値に設定することで，メモリ・プールのチェックが可能になります。**ゼロ**に設定すると，しきい値のチェックは無効になります。しきい値は，JVM で標準設定されています。JVM は，ガーベジ・コレクションの実行中や割り当て時など，最も適切なタイミングでメモリ・プールのしきい値をチェックします。現在のメモリ使用率がしきい値を超えていると，**UsageThresholdExceeded** 属性が **true** に設定されます。
- **コレクション使用率のしきい値**：ガーベジ・コレクションの対象となった一部メモリ・プールの管理属性です。メモリ・プールでのガーベジ・コレクションが完了した後，プール内のメモリの一部は，オブジェクトによって使用されている状態で残ります。コレクション使用率のしきい値には，ガーベジ・コレクション後のメモリ使用率をチェックする値を設定します。このメモリ使用率がしきい値を超えていると，**CollectionUsageThresholdExceeded** 属性が **true** に設定されます。

JVM では, 次の 2 種類のメモリを管理します。いずれも JVM の起動時に割り当てられます。

- **ヒープ・メモリ**：JVM は，この実行時データ領域からすべてのクラス・インスタンスと配列にメモリを割り当てます。ヒープは，固定サイズの場合と可変サイズの場合があります。ガーベジ・コレクションは，ヒープ・メモリをオブジェクト用に再利用する自動メモリ管理システムです。
- **非ヒープ・メモリ**：すべてのスレッドが共有するメソッド領域と，JVM の内部処理や最適化に必要なメモリが含まれます。ここには，実行時定数プール，フィールドおよびメソッド・データ，メソッドとコンストラクタのコードなど，クラスごとの構造が格納されています。メソッド領域は論理的にはヒープの一部ですが，実装方法によっては JVM によるガーベジ・コレクションや圧縮の対象にならない場合があります。ヒープ領域と同様に，メソッド領域は固定サイズの場合と可変サイズの場合があります。メソッド領域のメモリは，連続である必要はありません。

  JVM 実装では，メソッド領域に加えて，非ヒープ・メモリの一部である内部処理用および最適化用のメモリが必要になることがあります。たとえば JIT コンパイラでは，パフォーマンスを向上するために，JVM コードから変換したネイティブ・マシン・コードを格納するメモリが必要になります。

## Current heap size

| | |
|---|---|
| **正式名** | Current heap size |
| **カウンタのタイプ** | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| **説明** | 現在使用されているメモリ容量です。 |
| **使用方法** | 使用メモリ容量には, アクセス可能なオブジェクトとアクセスできないオブジェクトの両方が占有するメモリが含まれます。 |
| **パフォーマンス** | メモリが潜在的なボトルネックかどうかを判断する際の重要な指標です。 |
| **操作** | このカウンタの値が時間と伴に増大する場合, 再構成が必要になることがあります。 |
| **しきい値** | 上記の説明を参照してください。 |

### Maximum heap size

| | |
|---|---|
| **正式名** | Maximum heap size |
| **カウンタのタイプ** | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| **説明** | メモリ管理に使用できるメモリの最大容量です。この値は変化することがあり，明示されないこともあります。 |
| **使用方法** | 使用メモリ容量をコミット・メモリを超えるレベルにまで増やそうとすると，使用メモリ容量が最大メモリ容量以下であっても（たとえば，システムの仮想メモリが少ない場合など），メモリ割り当てが失敗することがあります。 |
| **パフォーマンス** | メモリ容量の上限を示します。物理メモリの上限に近付いた場合は注意が必要です。 |
| **操作** | 暗黙的に定義されていない場合，メモリ割り当てが適切に行われなくなる可能性があります。 |
| **しきい値** | 上記の説明を参照してください。 |

## Committed memory

| | |
|---|---|
| **正式名** | Committed memory |
| **カウンタのタイプ** | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| **説明** | ヒープ用に割り当てられているメモリ容量の合計を示します。 |
| **使用方法** | JVM が使用可能な容量として保証されたメモリ容量を示します。コミット・メモリの容量は時間と伴に変化することがあります。JVM はシステムにメモリを解放すると，起動時に割り当てられた初期容量よりもコミット・メモリが小さくなることがあります。 |
| **パフォーマンス** | なし |
| **操作** | なし |
| **しきい値** | 必ず，使用メモリ容量以上になります。 |

## GC time

| | |
|---|---|
| **正式名** | GC (Garbage Collection) time |
| **カウンタのタイプ** | 経過時間 |
| **説明** | ガーベジ・コレクションに費やした累計時間と呼び出し回数です。行が複数ある場合は，それぞれが JVM で使用されるガーベジ・コレクション・アルゴリズムを示しています。 |
| **使用方法** | このカウンタから，参照されなくなったオブジェクトが占有しているメモリを JVM が解放する処理がわかります。<br>一般的に，アクティブな参照を持つオブジェクトを「**生きている**」状態，参照されない（アクセスできない）オブジェクトを「**死んでいる**」状態とみなします。ガーベジ・コレクションは，死んでいる状態のオブジェクトが使用しているメモリを解放するプロセスです。 |
| **パフォーマンス** | GC が使用するアルゴリズムとパラメータは，パフォーマンスに多大な影響を与えることがあります。Sun の HotSpot VM ガーベジ・コレクタは，**世代別**ガーベジ・コレクションを使用します。世代別 GC は，ほとんどのプログラムにみられる次のような現象を考慮して設計されています。<br>➤ 生成されるオブジェクトの多くは寿命が短い（反復子やローカル変数など）<br>➤ 一部のオブジェクト（高レベルの永続的なオブジェクトなど）の寿命は非常に長い<br>世代別 GC ではメモリを複数の世代に分割し，それぞれにメモリ・プールを割り当てます。ある世代が割り当てメモリをすべて消費すると，VM はそのメモリ・プールで部分的なガーベジ・コレクション（**マイナー**・コレクション）を実行し，死んだ状態のオブジェクトが使用していたメモリを解放します。この部分的 GC は，完全な GC よりも大幅に短い時間で実行されます。 |
| **操作** | GC がボトルネックになっている場合，世代のサイズをカスタマイズしてください。GC の詳細出力をチェックし，GC パラメータに対するパフォーマンス・カウンタの感度を確認します。 |
| **しきい値** | 必ず，使用メモリ容量以上になります。 |

**注 :** メモリ設定が最適化されていない場合，ユーザにとって GC の間隔は大きな問題の 1 つになります。JVM のメモリ割り当ての方法や GC の動作は，さまざまな設定の影響を受けます。GC を監視し，チューニングを行う大きな目的は，主要な GC イベントの実行時間を抑えながら，頻度を低くすることにあります。

## スレッド・カウンタ

**スレッド**とは，プログラム内での実行スレッドです。JVM では，アプリケーションは複数のスレッドを同時実行できます。スレッドにはそれぞれ優先度があり，優先度の高い順に実行されます。スレッドは，デーモンと非デーモンがあります。あるスレッドで実行中のコードが新しい Thread オブジェクトを生成する場合，この新しいスレッドに最初に割り当てられる優先度は，オブジェクトを生成したスレッドの優先度になります。また，オブジェクトを生成したスレッドがデーモンの場合は，デーモン・スレッドが生成されます。

JVM の起動時，デーモンでないスレッドが 1 つ生成されます（通常は，指定されたクラスの **main** という名前のメソッドを呼び出します）。JVM は，次の条件が満たされるまで，スレッドの実行を続けます。

- Runtime クラスの exit メソッドが呼び出され，セキュリティ・マネージャが終了操作の実行を許可した場合。
- run メソッドの呼び出しから戻るか，run メソッドを超える例外が発生したことによってデーモンでないスレッドがすべて終了した場合。

### Live threads

| | |
|---|---|
| **正式名** | Live threads |
| **カウンタのタイプ** | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| **説明** | 実行中のデーモン・スレッドとデーモン以外のスレッドの現在の数です。 |
| **使用方法** | なし |
| **パフォーマンス** | スレッドの数が多すぎると，ガーベジ・コレクションが低速になります。 |
| **操作** | なし |
| **しきい値** | このカウンタの値が Peak threads の値に近付いた場合，注意が必要です。 |

### Peak threads

| | |
|---|---|
| **正式名** | Peak threads |
| **カウンタのタイプ** | 累計値 |
| **説明** | JVM の起動以降のライブ・スレッドの最大数です。 |
| **使用方法** | JVM の動作パターンを特定するときに役立ちます。 |
| **パフォーマンス** | なし |
| **操作** | なし |
| **しきい値** | なし |

### Daemon threads

| | |
|---|---|
| **正式名** | Daemon threads |
| **カウンタのタイプ** | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| **説明** | 実行中のデーモン・スレッドの現在の数です。 |
| **使用方法** | |
| **パフォーマンス** | スレッドの数が多すぎると，ガーベジ・コレクションが低速になります。 |
| **操作** | なし |
| **しきい値** | このカウンタの値が Peak threads の値に近付いた場合，注意が必要です。 |

### Total started threads

| | |
|---|---|
| **正式名** | Total started threads |
| **カウンタのタイプ** | 累計値 |
| **説明** | JVM の起動以降に開始されたスレッドの総数（デーモン，デーモン以外，終了したスレッドを含む）です。 |
| **使用方法** | JVM の動作パターンを特定するときに役立ちます。 |
| **パフォーマンス** | なし |
| **操作** | なし |
| **しきい値** | なし |

---

**注 :** スレッド・カウンタは，1 つのカウンタから別のカウンタ値を計算できるので，カウンタのペアを 1 つのみ（Total Started Threads と Live Threads など）監視しておけば十分です。

---

**ヒント：**アプリケーションがデッドロック（ハングアップなど）に陥っているかどうかを確認するには，**findMonitorDeadlockedThreads** を JConsole の［MBeans］タブから呼び出します。

## クラス・カウンタ

本項では，最も重要なクラス・カウンタについて説明します。

### Current classes loaded

| | |
|---|---|
| **正式名** | Current classes loaded |
| **カウンタのタイプ** | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| **説明** | 現在メモリにロードされているクラスの数です。 |
| **使用方法** | |
| **パフォーマンス** | なし |
| **操作** | なし |
| **しきい値** | なし |

### Total classes loaded

| | |
|---|---|
| **正式名** | Total classes loaded |
| **カウンタのタイプ** | 累計値 |
| **説明** | JVM の起動以降，メモリにロードされたクラスの総数（ロード後にアンロードされたクラスも含む）です。 |
| **使用方法** | JVM の動作パターンを特定するときに役立ちます。 |
| **パフォーマンス** | なし |
| **操作** | なし |
| **しきい値** | なし |

## Total classes unloaded

| | |
|---|---|
| **正式名** | Total classes unloaded |
| **カウンタのタイプ** | 累計値 |
| **説明** | JVM の起動以降，メモリからアンロードされたクラスの数です。 |
| **使用方法** | JVM の動作パターンを特定するときに役立ちます。 |
| **パフォーマンス** | ゼロ以外の状態が長く続く場合，ガーベジ・コレクション機能に問題が発生している可能性があります。 |
| **操作** | なし |
| **しきい値** | なし |

# 第7章

# .Net プラットフォームの監視

本章では，.Net プラットフォームの監視に関するベスト・プラクティスを紹介します。

**本章の内容**



## 概要

Microsoft のテクノロジで開発されたアプリケーションのほとんどは，.NET Framework で稼働します。このフレームワークは，アプリケーションの開発と実行の両方に対応したプラットフォームを提供します。また，アプリケーションのパフォーマンスの測定と監視を行うカウンタも備えています。

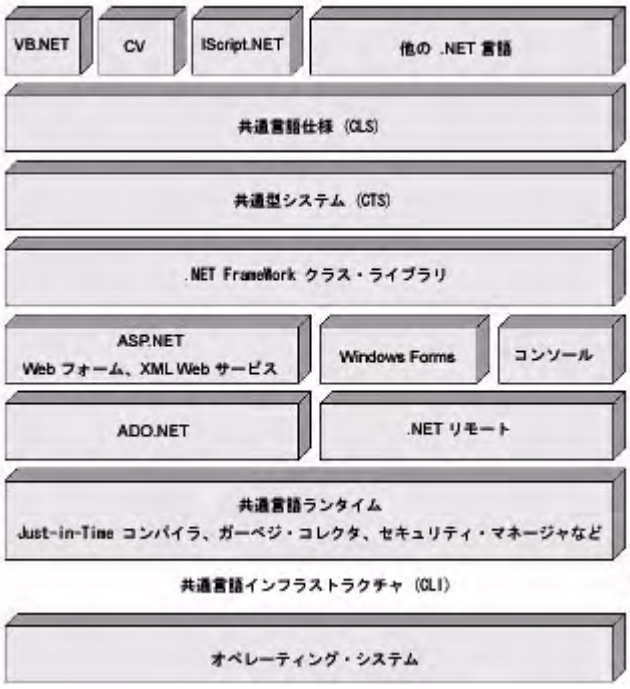
.NET Framework には，主に次の 2 つのコンポーネントがあります。

➤ 共通言語ランタイム

➤ .NET Framework クラス・ライブラリ

共通言語ランタイム（CLR）は，.NET Framework の基盤です。ランタイムとは，実行時のコードを管理するエージェントであり，メモリ管理，スレッド管理，リモート実行などの中核サービスを提供すると同時に，厳格なタイプ・セーフなどコードの正確性を強制する機能によってセキュリティと堅牢性を強化します。実際，このコード管理の概念がランタイムの基本原則となっています。ランタイムを対象とするコードはマネージ・コードと呼ばれ，それ以外はアンマネージ・コードと呼ばれます。.NET Framework の主要なコンポーネントの 1 つであるクラス・ライブラリは，オブジェクト指向の再利用可能な型のコレクションです。

ランタイムでは，パフォーマンス向上を目指す設計が採用されています。共通言語ランタイムは標準ランタイム・サービスマネージ・サービスを多数提供していますが，マネージ・コードの変換は行われません。Just-In-Time（JIT）コンパイルという機能を使ってマネージ・コードはコンパイルされ，実行先のシステムのネイティブ・マシン言語で実行されます。一方，メモリ・マネージャはメモリの断片化を低減し，参照局所性を高めることを通じてパフォーマンスをさらに向上します。



パフォーマンス・カウンタは，カテゴリに分類されています。Windows オペレーティング・システムでは多数のパフォーマンス・カウンタが事前定義されており，プログラムやパフォーマンス・モニタを使って表示することができますが，これと同様に .Net では CLR にパフォーマンス・カウンタが用意されています。このカウンタは次に示す 9 つの主要カテゴリに分類され，アプリケーション・パフォーマンスの監視とチューニングに活用できます。

- **例外**：アプリケーションが発行した例外に関する情報を提供します。
- **メモリ**：ガーベジ・コレクションに関する情報を提供します。
- **ロックおよびスレッド**：アプリケーションが使用するマネージ・ロックおよびスレッドに関する情報を提供します。
- **相互運用**：アプリケーションと COM コンポーネント，COM+ サービス，タイプ・ライブラリに関する情報を提供します。
- **JIT**：Just In Time コンパイラでコンパイルされたコードに関する情報を提供します。
- **読み込み**：読み込まれたアセンブリ，クラス，AppDomain に関する情報を提供します。
- **ネットワーク**：アプリケーションがネットワーク経由で送受信するデータに関する情報を提供します。
- **リモート処理**：アプリケーションが使用するリモート・オブジェクトに関する情報を提供します。
- **セキュリティ**：CLR がアプリケーションで実行するセキュリティ・チェックに関する情報を提供します。

# 最も重要な .Net カウンタ

.Net アプリケーションの監視では，最初の手順として，プロセッサ，メモリ，ネットワーク，I/O デバイスの使用状況を測定するオペレーティング・システム・カウンタ（Windows について説明している章を参照してください）を使うことをお勧めします。次の手順として，.Net パフォーマンス・カウンタを追加し，例外処理からセキュリティ・チェックまで，CLR の操作をあらゆる面から監視します。

| | カウンタ | 説明 |
|---|---|---|
| 例外 | **# of Excep Thrown/sec** | 1 秒あたりに派生したマネージ・コードの例外の数。 |
| | **Throw to Catch Depth/Sec** | スタック・フレームの数。 |
| メモリ | **Large Object Heap Size** | 大きなオブジェクト・ヒープの現在のサイズ（バイト数）。 |
| | **# Bytes in all Heaps** | GC ヒープに割り当てられている現在のメモリ容量（バイト数）。 |
| | **# of Pinned Objects** | 最後の GC で検出されたピン止めオブジェクトの数。 |
| | **% Time in GC** | 前回の GC サイクルで GC の実行に費やされた経過時間の割合（パーセント）。 |
| スレッド | **# of Current Logical Threads** | アプリケーション内にある現在の .NET スレッド・オブジェクトの数。 |
| | **# of Current Physical Threads** | CLR が作成および所有するネイティブ OS スレッドの数。 |
| | **# of Current Recognized Threads** | CLR が現在認識しているスレッドの数。 |
| | **# of Total Recognized Threads** | 開始以降，CLR が認識しているスレッドの総数。 |
| | Contention Rate/Sec | ランタイムのスレッドがマネージ・ロックの取得に失敗した回数。 |

| | カウンタ | 説明 |
|---|---|---|
| 読み込み | Current Assemblies | プロセスに読み込まれているアセンブリの数。 |
| | Rate of Assemblies | 1 秒間にメモリに読み込まれたアセンブリの数。 |
| | Bytes in Loader Heap | クラス・ローダによってコミットされたバイト数。 |
| セキュリティ | Total Runtime Checks | ランタイムのコード・アクセス・セキュリティの実行に費やされた経過時間の割合（パーセンテージ）。 |
| | Stack Walk Depth | 最後に実行したランタイム・コード・アクセス・セキュリティ・チェック時のスタックの深さ。 |

## 例外カウンタ

本項では，.Net アプリケーションがスローした例外に関する情報を提供するカウンタについて説明します。

### # of Excep Thrown/sec

| | |
|---|---|
| **正式名** | .NET CLR Exception/# of Excep Thrown/sec (_Global_) |
| **カウンタのタイプ** | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| **説明** | このカウンタは，1 秒間にスローされた例外の数を表示します。.NET 例外とアンマネージ例外の両方が含まれます。 |
| **使用方法** | このカウンタには，処理済みの例外と未処理の例外の両方が含まれます。例外が発生するケースはまれであり，プログラムの通常の制御フローでは発生しないと想定されます。 |
| **パフォーマンス** | 例外が頻発することが原因で発生するパフォーマンスの問題を特定する指標です。 |
| **操作** | 正常稼働時の値は 0 です。 |
| **しきい値** | 100 を超える場合，エラーとみなされます。 |

### Throw to Catch Depth/Sec

| | |
|---|---|
| **正式名** | .NET CLR Exception/Throw to Catch Depth/Sec |
| **カウンタのタイプ** | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| **説明** | .NET 例外をスローしたフレームから例外を処理したフレームまで，走査したスタック・フレームの数を 1 秒あたりの値で表示します。 |
| **使用方法** | この値は，例外ハンドラに入った時点で 0 にリセットされます。したがって，ネストされた例外では，ハンドラからハンドラへのスタックの深さを示します。 |
| **パフォーマンス** | コードの欠陥が潜在的なボトルネックになっているかどうかを判断する際の補助的な指標です。 |
| **操作** | なし |
| **しきい値** | なし |

## メモリ・カウンタ

本項では，.Net CLR のメモリ管理に関連するカウンタについて説明します。このカウンタは，メモリ消費量，メモリ・プール，ガーベジ・コレクションなどの統計を表示します。

共通言語ランタイムのガーベジ・コレクション（GC）は，アプリケーションに対するメモリの割り当てと解放を管理します。このメモリ管理は自動で実行され，オブジェクトを解放し忘れたことが原因で発生するメモリ・リークや，すでに解放されたオブジェクトのメモリへのアクセスといった問題を解消できます。

.Net アプリケーションを初期化する際，ランタイムはプロセス用に連続したアドレス領域を予約します。この予約済みのアドレス領域を**マネージ・ヒープ**と呼びます。アプリケーションが最初のオブジェクトを作成すると，このマネージ・ヒープのベース・アドレスのメモリが割り当てられます。次に作成したオブジェクトには，ガーベジ・コレクタは最初のオブジェクトに割り当てたすぐ後のアドレス領域を割り当てます。この方法に従って，ガーベジ・コレクタは，使用可能なアドレス領域が存在する限り新しいオブジェクトに領域を割り当てていきます。マネージ・ヒープからメモリを割り当てる方が，**アンマネージ**・メモリからの割り当てよりも高速です。アンマネージ・リソースの場合，GC は実行スタックを追跡できるとは限らないため，明示的なクリーンアップが必要になります。

ガーベジ・コレクタのパフォーマンスを最適化するために，マネージ・ヒープは**Gen 0**，**Gen 1**，**Gen 2** という 3 つの世代に分割されます。ランタイムのガーベジ・コレクタは，新しいオブジェクトを世代 0 に格納します。アプリケーションの寿命の初期の段階で作成され，ガーベジ・コレクションの実行後も残っているオブジェクトは世代 1 に格納され，次に世代 2 へと昇格します。この方法では，ガーベジ・コレクションの実行時に特定の世代のメモリを解放するため，マネージ・ヒープ全体のメモリを解放するより速度が向上します。

## Large Object Heap Size

| | |
|---|---|
| **正式名** | .NET CLR Memory¥Large Object Heap Size |
| **カウンタのタイプ** | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| **説明** | 大きなオブジェクト・ヒープが現在使用しているメモリ容量（バイト数）です。 |
| **使用方法** | ガーベジ・コレクタは，20 KB を超えるオブジェクトを大きなオブジェクトとして認識し，特殊なヒープを直接割り当てます。 |
| **パフォーマンス** | 世代ごとに解放するアルゴリズムが正常稼働している状態に比べて，ヒープ全体を解放する方が時間がかかるので，コードの効率を判断する際の重要な指標になります。一般的に，このようなボトルネックは大きなオブジェクト・ヒープの断片化と呼ばれます。 |
| **操作** | 大きなオブジェクトは上の世代に昇格しません。 |
| **しきい値** | なし |

## # Bytes in all Heaps

| | |
|---|---|
| **正式名** | .NET CLR Memory¥# Bytes in all Heaps |
| **カウンタのタイプ** | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| **説明** | GC ヒープに割り当てられている現在のメモリ容量（バイト数）を示します。 |
| **使用方法** | このカウンタは，Gen 0 Heap Size，Gen 1 Heap Size，Gen 2 Heap Size，Large Object Heap Size の各カウンタの値の合計です。このカウンタは，ガーベジ・コレクション・ヒープに割り当てられている現在のメモリ容量を示します。 |
| **パフォーマンス** | メモリ内の大きなデータ・セットを使用する場合，過剰な数のキャッシュ・エントリ，正規表現や文字列の解析，過剰な数のビュー状態，過剰な数のセッション・オブジェクトは，必要なメモリ容量が増大する原因になります。 |
| **操作** | 一般的な方法として，最初にこのカウンタを監視します。<br>Private Bytes カウンタの値が増大し, # of Bytes in all Heaps カウンタの値に変化がない場合，アンマネージ・メモリが消費されていることを示します。両方のカウンタの値が増大する場合，マネージ・メモリが消費されていることを示します。 |
| **しきい値** | Process¥Private Bytes カウンタの値よりも小さい値が適正値です。 |

## # of Pinned Objects

| | |
|---|---|
| **正式名** | .NET CLR Memory¥# of Pinned Objects |
| **カウンタのタイプ** | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| **説明** | 最後の GC で検出されたピン止めオブジェクトの数を示します。 |
| **使用方法** | ピン止めオブジェクトとは，ガーベジ・コレクタがメモリ内で移動できないオブジェクトです。このカウンタは，ガーベジ・コレクションを実行したヒープ内のピン止めオブジェクトのみを追跡します。したがって，Gen 0 のガーベジ・コレクションでは，世代 0 のヒープにあるピン止めオブジェクトのみを列挙します。 |
| **パフォーマンス** | なし |
| **操作** | なし |
| **しきい値** | なし |

### % Time in GC

| | |
|---|---|
| **正式名** | .NET CLR Memory¥% Time in GC |
| **カウンタのタイプ** | 経過時間 |
| **説明** | 前回の GC サイクルで GC の実行に費やされた経過時間の割合（パーセンテージ）を示します。 |
| **使用方法** | このカウンタは，ガーベジ・コレクタが，アプリケーションに代わって行ったメモリの回収および圧縮作業を示します。 |
| **パフォーマンス** | 大きな文字列をキャッシュに格納すると（負荷の大きな文字列操作など），大きなメモリ領域が残るので，GC によるクリーンアップが必要になります。 |
| **操作** | このカウンタの値が更新されるのは各 GC の終了時のみです。したがって，カウンタの値は平均値ではなく，前回取得された値となります。したがって，カウンタの値にスパイクが発生していても問題ありません。 |
| **しきい値** | 適正な範囲は 5～10% です。 |

## スレッド・カウンタ

**スレッド**とは，プログラム内での実行スレッドです。.NET **論理スレッド・**オブジェクトは，新しい **System.Threading.Thread** コマンドをコード内で発行することによって暗黙的に作成される場合と，アンマネージ・スレッドが管理環境に入るときに明示的に作成される場合があります。また物理スレッドは，CLR によって作成および所有され，.NET スレッド・オブジェクトの基になるスレッドとして動作するネイティブ OS です。.Net アプリケーションは，CLR の外部で作成されたとしても，CLR 内部で 1 回以上実行されたことがあるので CLR によって認識されているスレッドを使用することがあります。

## # of Current Logical Threads

| | |
|---|---|
| **正式名** | .NET CLR LocksAndThreads¥# of Current Logical Threads |
| **カウンタのタイプ** | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| **説明** | アプリケーション内にある現在のマネージ・スレッド・オブジェクトの数を示します。 |
| **使用方法** | 実行中のスレッドと停止したスレッドの両方がカウントされます。 |
| **パフォーマンス** | スレッドの数が多すぎると，ガーベジ・コレクションが低速になります。 |
| **操作** | なし |
| **しきい値** | |

## # of Current Physical Threads

| | |
|---|---|
| **正式名** | .NET CLR LocksAndThreads¥# of Current Physical Threads |
| **カウンタのタイプ** | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| **説明** | マネージ・スレッド・オブジェクトの基になるスレッドとして動作し，共通言語ランタイムによって作成および所有されているネイティブのオペレーティング・システム・スレッドの数を示します。 |
| **使用方法** | これは，OS プロセスのスレッドのサブセットです。 |
| **パフォーマンス** | なし |
| **操作** | このカウンタには，CLR が内部操作で使用するスレッドは含まれません。 |
| **しきい値** | なし |

### # of Current Recognized Threads

| | |
|---|---|
| **正式名** | .NET CLR LocksAndThreads¥# of Current Recognized Threads |
| **カウンタのタイプ** | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| **説明** | CLR が現在認識しているスレッドの数を表示します。 |
| **使用方法** | 一意のスレッドのみが追跡されます。 |
| **パフォーマンス** | なし |
| **操作** | 同じスレッド ID で CLR に再入力するスレッドや, 終了後に再作成されたスレッドは 2 回カウントされません。 |
| **しきい値** | なし |

### # of Total Recognized Threads

| | |
|---|---|
| **正式名** | .NET CLR LocksAndThreads¥# of Total Recognized Threads |
| **カウンタのタイプ** | 累計値 |
| **説明** | アプリケーションの開始以降，CLR が認識しているスレッドの総数を示します。 |
| **使用方法** | 一意のスレッドのみが追跡されます。 |
| **パフォーマンス** | なし |
| **操作** | 同じスレッド ID で CLR に再入力するスレッドや, 終了後に再作成されたスレッドは 2 回カウントされません。 |
| **しきい値** | なし |

## Contention Rate/Sec

| | |
|---|---|
| **正式名** | .NET CLR LocksAndThreads¥Contention Rate/Sec |
| **カウンタのタイプ** | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| **説明** | ランタイムのスレッドがマネージ・ロックの取得に失敗した回数を示します。 |
| **使用方法** | 競合の回数が増加する場合や，総数が大幅に増える場合は，アプリケーションでスレッドの競合が発生している可能性が高くなります。この問題を解消するためには，共有リソースにアクセスするコードを特定するか，同期機能を使用する必要があります。 |
| **パフォーマンス** | Total # of Contentions と組み合わせて監視することによって，スレッドのボトルネックを特定できます。 |
| **操作** | |
| **しきい値** | なし |

## 読み込みカウンタ

本項では，最も重要な読み込みカウンタについて説明します。

### Current Assemblies

| | |
|---|---|
| **正式名** | .NET CLR Loading/Current Assemblies (_Global_) |
| **カウンタのタイプ** | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| **説明** | プロセスに読み込まれているアセンブリの数を表示および記録します。 |
| **使用方法** | この値は，現在実行中のアプリケーション内のアプリケーション・ドメインすべての累計値です。 |
| **パフォーマンス** | なし |
| **操作** | アセンブリが複数のアプリケーション・ドメインからドメイン中立で読み込まれた場合，このカウンタの値は 1 回のみインクリメントされます。 |
| **しきい値** | なし |

### Rate of Assemblies

| | |
|---|---|
| **正式名** | .NET CLR Loading/Rate of Assemblies |
| **カウンタのタイプ** | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| **説明** | 1 秒間にメモリに読み込まれたアセンブリの数を示します。 |
| **使用方法** | この値は，現在実行中のアプリケーション内のアプリケーション・ドメインすべての累計値です。 |
| **パフォーマンス** | なし |
| **操作** | アセンブリが複数のアプリケーション・ドメインからドメイン中立で読み込まれた場合，このカウンタの値は 1 回のみインクリメントされます。 |
| **しきい値** | なし |

## Bytes in Loader Heap

| | |
|---|---|
| **正式名** | .NET CLR Loading/Bytes in Loader Heap |
| **カウンタのタイプ** | 累計値 |
| **説明** | すべてのアプリケーション・ドメインについて，クラス・ローダによってコミットされたバイト数を示します。 |
| **使用方法** | コミットされたメモリとは，ディスク・ページング・ファイル内で予約された物理メモリ領域を指します。 |
| **パフォーマンス** | このカウンタの値を安定させる必要があります。変動が大きい場合，アプリケーション・ドメインに読み込まれるアセンブリの数が多すぎることを示します。 |
| **操作** | なし |
| **しきい値** | なし |

## セキュリティ・カウンタ

本項では，最も重要なセキュリティ・カウンタについて説明します。

### Total Runtime Checks

| | |
|---|---|
| **正式名** | .NET CLR Security/Total Runtime Checks |
| **カウンタのタイプ** | 累計値 |
| **説明** | アプリケーションの開始以降，実行されたランタイムのコード・アクセス・セキュリティ（CAS）チェックの総数を示します。 |
| **使用方法** | CAS は，さまざまなレベルの信用をコードに割り当て，コード ID に基づいて信用レベルを適用します。ランタイムのコード・アクセス・セキュリティ・チェックは，呼び出し元が特定のアクセス許可を要求すると実行されます。ランタイム・チェックは，呼び出し元が行うすべての呼び出しに対して実行され，呼び出し元の現在のスレッド・スタックをチェックします。 |
| **パフォーマンス** | Stack Walk Depth カウンタと併せて監視することにより，セキュリティ・チェック時に発生するパフォーマンス低下を把握できます。 |
| **操作** | このカウンタは，ランタイムのセキュリティ・チェックの終了時に更新されます。 |
| **しきい値** | なし |

### Stack Walk Depth

| | |
|---|---|
| **正式名** | .NET CLR Security/Stack Walk Depth |
| **カウンタのタイプ** | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| **説明** | 最後に実行したランタイム・コード・アクセス・セキュリティ・チェック時のスタックの深さを示します。 |
| **使用方法** | ランタイムのコード・アクセス・セキュリティ・チェックは，スタックをクロールすることによって実行されます。 |
| **パフォーマンス** | なし |
| **操作** | Total Runtime Checks カウンタと併せて監視することにより，セキュリティ・チェック時に発生するパフォーマンス低下を把握できます。 |
| **しきい値** | なし |

# 第IV部

## Web サーバの監視

# 第8章

# Apache の監視

本章では，Apache の監視に関するベスト・プラクティスを紹介します。

**本章の内容**

# 概要

Apache HTTP サーバは，設定と拡張が可能なオープン・ソースの Web サーバです。1995 年に NCSA httpd（HTTP デーモン）をベースに開発された後，Apache HTTP サーバは商用 Web サイトや Web ベース・アプリケーション用の Web サーバとして非常に広く普及しています。

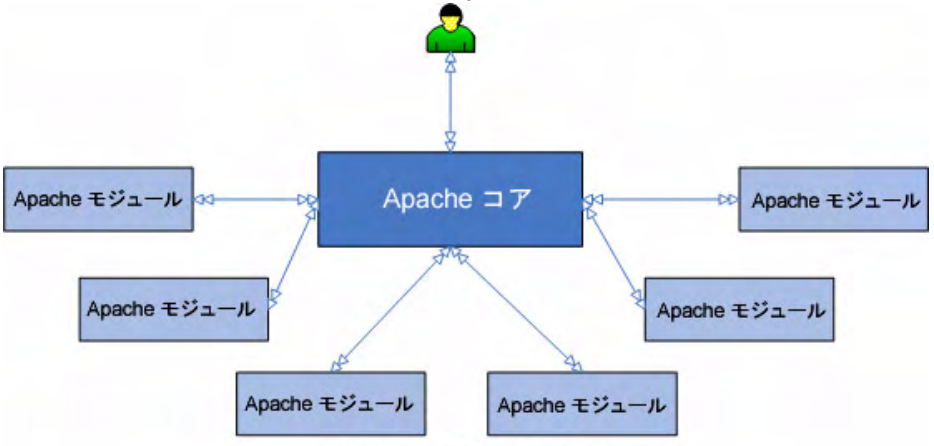
このように Apache は主要な Web サーバとして導入されているため，高レベルなアーキテクチャ，監視およびチューニング・カウンタ，その他パフォーマンスに関連するベスト・プラクティスを理解することが重要です。本章では，上記の点に加えて，LoadRunner および ALM Performance Center で提供されている Apache Web サーバの監視機能についても説明します。

# アーキテクチャ

Web サーバには，HTTP プロトコルを使って要求された処理を実行する機能があります。一般的にサーバは，特定のリソースに対する要求を受信し，クライアントへの応答としてリソースを渡します。Apache では，この要求処理を Apache コアと Apache モジュールに分けて実行します。

- コアは，要求処理の手順を定義し，手順を進めます。
- モジュールは，要求処理の各フェーズを実際に実装します。

このようなアーキテクチャは，サードパーティにとっては機能のオーバーライドや拡張がしやすいというメリットがあり，管理者にとっては不要なモジュールを無効にすることでメモリ管理を最適化できるというチューニング上のメリットがあるため，Apache は理想的なプラットフォームになっています。




高レベルの概念アーキテクチャ

Apache 2.0 と Apache 1.3 はいずれも運用環境で求められる品質を満たしたバージョンですが，2.0 のアーキテクチャと機能の方が優れています。ここでは，Apache 2.0 アーキテクチャのパフォーマンスに関する内容を説明します。

- **マルチ・プロセッシング・モジュール**：Apache 1.3 では起動時に複数の子プロセスが生成されますが，Apache 2.0 ではマルチ・プロセッシング・モジュール（MPM）がサポートされています。MPM の採用により，純粋なプロセス・ベースのサーバ，純粋なスレッド・サーバ，またはその組み合わせとして Apache を設定することが可能になります。プロセス内にある複数のスレッドは同時実行されるので，ほとんど場合，スレッド・サーバの方がプロセス・ベース・サーバよりも高いスケーラビリティを発揮します。

- **モジュールとフィルタ**：これまでに説明したように，Apache ではモジュール・アーキテクチャが採用されています。Apache 2.0 では，拡張機能としてフィルタが追加されています。モジュールは，フィルタを使用することによって他のモジュールが生成した内容を変更することができます。これにより，静的ファイルだけでなく，動的に生成されたファイルの暗号化，ウィルス・スキャン，圧縮も可能になります。

- **Apache Portable Runtime**：Apache 2.0 では，Apache Portable Runtime（APR）ライブラリのサポートにより，Windows と UNIX の両方のプラットフォームでの安定稼働が可能になりました。APR は，ファイルやネットワーク・アクセス API など，オペレーティング・システム間での相違部分を抽象化します。この抽象化層では，プラットフォーム固有のチューニングと最適化が行われます。APR はメモリ・プールという概念に基づき，メモリ管理コードの大幅な簡素化と，メモリ・リークの低減を実現しています。

Apache **mod_status** モジュールは，Apache アーキテクチャに準拠した方法でサーバ・ステータスの把握と監視を行うカウンタを提供します。このモジュールは，サーバの稼働状況とパフォーマンスに関する情報を提供します。サーバ統計は，読みやすい形式の HTML ページ（http://your.server.name/server-status など），または自動監視機能向けのシンプルなマシンによる判読可能なリスト（http://your.server.name/server-status?auto）で提供されます。いずれの形式も，URL クエリ文字列（http://your.server.name/server-status?auto&refresh=30 は，マシン向けのステータス情報を 30 秒ごとに自動更新します）を追加することによって自動更新が可能です。

また，**mod_status** モジュールではさらに詳細なステータス情報を提供する設定もできます。この設定は，標準設定では無効です。

---

**注**：Apache は，統計収集のために Web サーバへの接続を監視し，サンプリングごとに 1 回のヒットを登録します。したがって Apache のグラフでは，Apache サーバにクライアントが接続されていない状態でも，1 秒あたり 1 回以上のヒットが必ず表示されます。

---

# 最も重要な Apache カウンタ

HP LoadRunner/ALM Performance Center Apache モニタは, カウンタをマシンによる読み取り可能なページ（server-status?auto）で表示します。HP Sitescope は両方の読み取りモードをサポートします。最も重要なカウンタは，マシンによる読み取り可能なページで提供されます。

| カウンタ | 説明 |
|---|---|
| **CPULoad** | Apache サーバが現在消費している CPU の割合（パーセンテージ)。 |
| **ReqPerSec** | 1 秒あたりの要求の数（1 秒あたりのヒット数)。 |
| **BytesPerSec** | 1 秒あたりの転送バイト数。 |
| **BytesPerReq** | 要求あたりの転送バイト数。 |
| **BusyWorkers** | 要求を処理しているアクティブ・スレッドの数。 |
| **IdleWorkers** | 非アクティブ/アイドルなスレッドの数。 |

---

**ヒント：**上記のカウンタはすべて server-status?auto ページで参照できます。カウンタ・データを解析する VuGen スクリプトを作成し，lr_user_data_point を使って LoadRunner/ALM Performance Center にオンラインで送信することができます。

---

## 最適化とチューニング

パフォーマンスに問題が発生している場合，問題点を解消するためにはチューニングと最適化が必要です。プロアクティブに監視を行い，問題が実際に発生する前に回避できる対策を講じることをお勧めします。本項では，Apache Web サーバで使用できるチューニングのためのパラメータ，最適化の手法，ベンチマーク方法をいくつか紹介します。ここで紹介する内容を実践する場合は，パラメータの機能とサーバ上の作業負荷をよく理解した上で，ご使用のサーバ構成に適しているかどうかを確認してください。

- HostnameLookups ディレクティブは Off にします（標準設定は Off）。On にすると，DNS ルックアップが大量の時間を消費し，サーバが低速になります。

- KeepAlive ディレクティブは On にします（標準設定は On）。KeepAlive を On にすると HTTP セッションが長くなり，同じ TCP 接続を使って複数の要求を送信可能になります。この設定は，応答時間を高速にする上で大きな効果を発揮します。

- KeepAliveTimeout ディレクティブは，Apache が次の要求を受け取るまでの間，接続を確立した状態で待機する時間を秒数で示します。標準設定値は 15 秒です。値を大きくすると，アイドルなクライアントとの接続で待機状態になるサーバ・スレッドの数が増大します。

- .htaccess ファイルは使用しません。.htaccess ファイルの使用を完全に無効にするには，AllowOverride ディレクティブを **none** に設定します。.htaccess ファイルの使用が有効になるように AllowOverride を設定している場合，Apache は .htaccess ファイルの検索ですべてのディレクトリをチェックします。したがって .htaccess ファイルの使用を有効にすると，実際に使用しない場合でもパフォーマンスが低下してしまいます。また，.htaccess ファイルはドキュメントが要求されるたびに読み込まれます。

- メモリ使用率を最適化するには，未使用のモジュールのロードを解除することをお勧めします。

- MaxKeepAliveRequests ディレクティブの設定。スワップが発生すると，各要求で遅延時間が発生し，ユーザが満足する速度を下回ってしまうため，Web サーバではスワップが発生しないようにします。スワップが発生すると，ユーザは停止と再読込を行うことになり，負荷はさらに増大します。したがってスワップを回避する方法として，MaxClients を適切な値に設定することにより，子プロセスの数が過剰に増えないようにします。MaxKeepAliveRequests ディレクティブは，要求を処理するために生成される子プロセスの最大数を指定し，同時処理する要求の数の上限を設けます。MaxClients の値を超える接続が発生すると，通常は，ListenBacklog ディレクティブの設定値を上限として，キュー内で待機します。この値には，スループットの低下や応答時間の延長が発生しない範囲で処理できるクライアント数の最大値を設定してください。

# 第9章

# IIS の監視

本章では，IIS の監視に関するベスト・プラクティスを紹介します。

**本章の内容**



## 概要

Microsoft Internet Information Services（IIS）は，Apache HTTP サーバに次いで 2 番目に普及している Web サーバです。IIS は，すべての Windows オペレーティング・システム・エディションにおいて各種フレーバで提供されており，進化と拡張を続けています。IIS 6.0 では大幅な改善が加えられ，Web サーバとしてだけでなくアプリケーション・サーバとしても利用可能になっています。さらに最新リリースである IIS 7.0 では，パフォーマンスと信頼性の向上に貢献する重要な機能が追加されました。

IIS では，FTP/FTPS，SMTP，NNTP，HTTP/HTTPS の各サーバがサポートされますが，本章では HTTP/HTTPS サーバについて，IIS アーキテクチャ，パフォーマンスの監視とチューニングに関するガイドラインを紹介します。ここでは，IIS 7.0 に関する説明も行いますが，IIS 6.0 を中心とした内容になります。

# アーキテクチャ

IIS には，アプリケーション分離モードと呼ばれる 2 つの要求処理モデルが用意されており，いずれかのモードでサーバが実行されます。アプリケーション分離とは，プロセス境界によってアプリケーションを分離することを指し，これによってアプリケーションや Web サイトで発生した問題が他に影響を与えるのを防ぐことができるので，アプリケーションに関する問題の修正のためにサービスの再起動に費やされる時間も短縮できます。

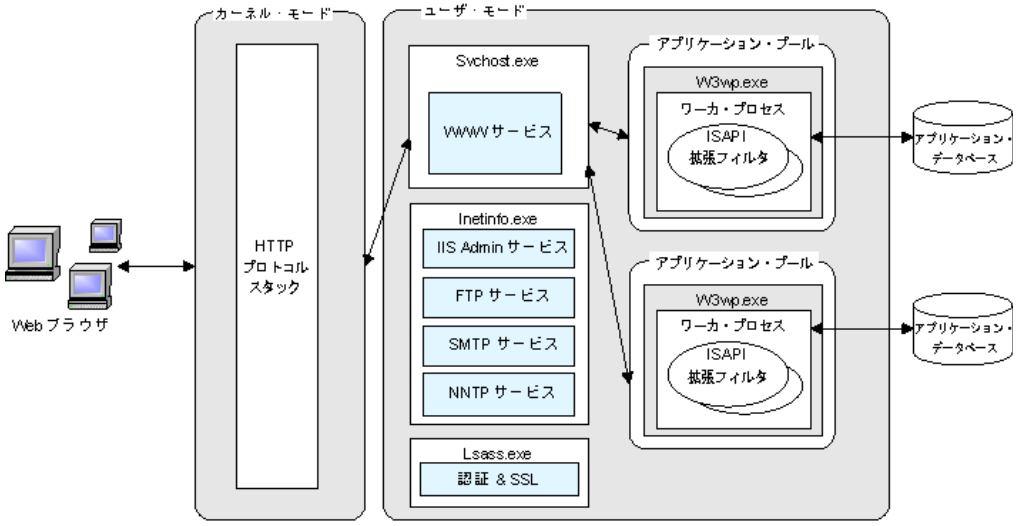
IIS 6.0 では，2 つのアプリケーション分離モードがサポートされ，それぞれアプリケーション分離の構成が異なります。

- **ワーカ・プロセス分離モード**：Web アプリケーションをアプリケーション・プールとしてグループ化します。これにより，アプリケーションは自己完結型のワーカ・プロセス内で稼働します。ワーカ・プロセスは，要求を処理するユーザ・モード・コードであり，静的ページを返す処理や Internet Server API（ISAPI）の拡張またはフィルタの呼び出しを行います。このモードでは，複数のアプリケーション・プール，稼働状態の監視とリサイクル，セキュリティとパフォーマンスの向上，スケーラビリティの向上，プロセッサの関係など，IIS 6.0 アーキテクチャの利点をすべて活用できます。
- **IIS 5.0 分離モード**：IIS の旧バージョン用に設計されているアプリケーションとの互換性を確保します。IIS 6.0 をこのモードで実行すると，IIS 5.0 とほとんど同じ方法で要求が処理されます。アプリケーションがワーカ・プロセス分離モードで機能しない場合を除き，このモードは使用しないでください。

いずれのモードも，HTTP プロトコル・スタック（HTTP.sys）を使って HTTP 要求を受信し，応答を返します。HTTP.sys は HTTP 要求をリッスンし，キューに追加して，要求の処理が完了した時点で応答を返します。

HTTP.sys はカーネル・モードで実行されます。カーネル・モードは，デバイス・ドライバなどのオペレーティング・システム・コードが実行されるモードであり，HTTP 要求は高い優先度でオペレーティング・システムによって処理されます。要求の実際の処理は，ワーカ・プロセスがユーザ・モードで行います。

次の図は，ワーカ・プロセス分離モードを示します。




アプリケーション・プールは，複数のワーカ・プロセスで使用できるため，負荷分散やフェイルオーバー機能にも対応でき，アプリケーションのパフォーマンス，信頼性，スケーラビリティの向上に貢献します。複数のワーカ・プロセスを含むアプリケーション・プールは，Web ガーデンと呼ばれます。

上記で説明したとおり，IIS は 4 つのインターネット・サービスを提供します。World Wide Web Publishing Service（WWW サービス）はインターネットおよびイントラネットのコンテンツのホスティング，File Transfer Protocol（FTP）サービスはユーザがファイルのアップロードとダウンロードを行うサイトのホスティング，Network News Transfer Protocol（NNTP）サービスはディスカッション・グループのホスティング，Simple Mail Transfer Protocol（SMTP）サービスは電子メール・メッセージの送受信をそれぞれ行います。IIS の負荷を軽減するために，使用しないサービスは無効化またはアンインストールすることをお勧めします。

IIS 7.0 では，アーキテクチャに次のような拡張が加えられてます。

- 新しいサービスとして Windows Process Activation Service（WAS）が追加されました。HTTP/HTTPS 以外のプロトコルの使用が可能になります。
- IIS と ASP.NET の要求処理パイプラインを統合します。これは，IIS 7.0 でサポートされるアプリケーション・プール・モードに関連する拡張です。
  - IIS 5.0 分離モードはサポートされなくなります
  - IIS 6.0 のワーカ・プロセス分離モードは引き続きサポート対象です
  - 新しいモードとして統合アプリケーション・プール・モードが追加されました。これにより，IIS と ASP.NET の要求処理を統合できます
- モジュールの追加または削除による Web Server エンジンのカスタマイズが可能です。

# 監視

IIS パフォーマンス・カウンタは，Microsoft Windows の汎用監視プラットフォームであるパフォーマンス・データ・ヘルパ・ライブラリ（pdh.dll）を使って表示できます。つまり，IIS パフォーマンス・カウンタは数値型であり，パスごとに一意に特定され，通常は次の構文で指定します。

```
¥¥Computer_name¥Object(Parent/Instance#Index)¥Counter
```

**Computer_name** のパス指定はオプションです。

SiteScope と LoadRunner はいずれも Windows pdh インタフェースを使用して IIS と ASP/ASP.NET に関連したカウンタの監視を行います。pdh インタフェースをリモート・マシンから呼び出すには，適切な権限を持つユーザの認証が必要になります。

ベスト・プラクティスとして，アプリケーションのアーキテクチャとデプロイメントを詳細に把握しておく必要があります。この情報は，製品ライフサイクル全体を通じて，さまざまなパフォーマンス関連のプラクティスを実行する際に非常に役立ちます。たとえば，IIS FTP サーバや Frontpage サーバ拡張は，使用しないのであれば実行する必要はありません。また，完全に ASP.NET ベースのアプリケーションであれば，Active Server Page を監視する必要はありません。

# 最も重要な IIS カウンタ

本項では，パフォーマンスと作業負荷を監視する重要なカウンタについて説明します。ただし，IIS 6.0 との互換性のないカウンタは取り扱いません。

---

**注：** .NET Web ベース・アプリケーションの監視では，.NET CLR も監視対象として含めることをお勧めします。.NET CLR の重要なカウンタについては，第7章「.Net プラットフォームの監視」を参照してください。

---

## WWW サービス

Web サービス・カウンタでは，World Wide Web Publishing Service（WWW Service）プロセス要求の処理効率を把握できます。WWW サービスは，ユーザ・モードのサービスです。このカウンタには，カーネル・モード・ドライバである **HTTP.sys** で発生する処理も反映されています。

このカウンタは，Web サイトごとに設定できますが，**_Total** インスタンスでサーバ全体をグローバルに監視することも可能です。

| カウンタ | 説明 |
|---|---|
| **Bytes Sent/sec** | WWW サービスが 1 秒間に送信したデータ・バイト数。 |
| **Bytes Received/sec** | WWW サービスが 1 秒間に受信したデータ・バイト数。 |
| **Current Connections** | WWW サービスに対して確立されている接続数。 |
| **Not Found Errors/sec** | 要求されたドキュメントが見つからないためにサーバが完了できなかった要求の 1 秒あたりの数。 |
| **Locked Errors/sec** | 要求されたドキュメントがロックされていたためにサーバが完了できなかった要求の 1 秒あたりの数。 |

| カウンタ | 説明 |
|---|---|
| **Current ISAPI Extension Requests** | WWW サービスが同時処理している ISAPI 拡張要求の数。 |
| **ISAPI Extension Requests/sec** | WWW サービスが 1 秒間に処理した ISAPI 拡張要求の数。 |

### WWW サービス・キャッシュ

WWW サービスと FTP サービスは, 1 つのキャッシュを共有するのではなく, キャッシュは FTP サービス専用と WWW サービス専用のパフォーマンス・オブジェクトに区別されます。WWW サービス・キャッシュ・カウンタは，サーバ・パフォーマンスのみを監視するカウンタなので，個々のサイトを監視する設定はできません。

| カウンタ | 説明 |
|---|---|
| **Current File Cache Memory Usage** | ユーザ・モードのファイル・キャッシュで現在使用されているバイト数。 |
| **Current Files Cached** | ユーザ・モード・キャッシュ内にコンテンツが現在格納されているファイルの数。 |
| **Current URIs Cached** | ユーザ・モード・キャッシュ内に現在格納されている URI 情報ブロックの数。 |
| **Current Metadata Cached** | ユーザ･モード･キャッシュ内に現在格納されているメタデータ情報ブロックの数。 |
| **Kernel: URI Cache Hits/sec** | カーネル URI キャッシュの 1 秒あたりの平均ヒット率。 |

## ASP.NET

ASP.NET は，次の ASP.NET システム・パフォーマンス・カウンタをサポートします。Web サーバ上で稼働するすべての ASP.NET アプリケーションのカウンタ値を集計するカウンタと，同じアプリケーションを実行する ASP.NET サーバに適用されるカウンタがあります。

---

**注：**また，一部の IIS デプロイメントでは使用できないカウンタもあります。

---

| カウンタ | 説明 |
|---|---|
| **Requests Disconnected** | 通信エラーが発生したために切断された要求の数。 |
| **Requests Queued** | キュー内で処理されるのを待機している要求の数。クライアント要求の増大に伴ってこの数値も増加する場合，Web サーバが処理可能な同時要求数の限界に到達したことを示しています。標準設定値は 5,000 であり，Machine.config ファイルで変更できます。 |
| **Requests Rejected** | サーバ・リソースの不足が原因で実行されなかった要求の総数。サーバがビジー状態である 503 HTTP ステータス・コードを返した要求の数を示します。 |
| **Errors Total/sec** | HTTP 要求の実行中に発生したエラーの 1 秒あたりの平均数。この数値には，パーサ，コンパイル，実行時のエラーが含まれます。 |
| **Output Cache Turnover Rate** | 出力キャッシュに対して 1 秒間に行われた追加と削除の平均回数。この値が大きい場合，キャッシュの使用効率が低いことを示します。 |
| **Sessions Active** | アクティブなセッションの数。このカウンタは，メモリ内セッション状態を使用する場合にのみサポートされます。 |

| カウンタ | 説明 |
| --- | --- |
| **Transactions/sec** | 1 秒間で開始されたトランザクションの平均数。 |
| **Transactions Pending** | 処理中のトランザクションの数。 |

## Active Server Page

Active Server Page（ASP）を実行するサーバでは，ASP カウンタを監視することによって，サーバやサイトが ASP 要求を処理する効率を把握できます。ASP カウンタは，サーバ・パフォーマンスの監視用に設計されており，WWW サービス全体でグローバルなデータ収集を行うので，ASP アプリケーションを個々に監視することはできません。

| カウンタ | 説明 |
| --- | --- |
| Errors/sec | 発生したエラーの 1 秒あたりの平均数。 |
| Requests/sec | 実行された要求の 1 秒あたりの平均数。 |
| Requests Executing | 現在実行中の ASP 要求の数（アクティブなワーカ・スレッドなど）。 |
| Requests Queued | キュー内で処理されるのを待機している ASP 要求の数。このカウンタの最大値は，AspRequestQueueMax メタデータ・プロパティで設定できます。 |
| Transactions/sec | 開始されたトランザクションの 1 秒あたりの平均数。 |

## 最適化とチューニング

パフォーマンスに問題が発生している場合，問題点を解消するためにはチューニングと最適化が必要です。アプリケーション・コードの最適化が必要になる場合がほとんどですが，チューニングで環境を効率化することによってパフォーマンスが格段に向上するケースもあります。

本項では，このようなチューニング方法をいくつか紹介します。ここには，IIS Web サーバに関する内容と，Web サーバで一般的な方法として適用できる内容が含まれています。また，ここで紹介する方法以外にも，個々のアプリケーションで高い効果を発揮するチューニング方法が多数存在します。

チューニングには，テストや分析を繰り返し行う必要があり，時間がかかります。また，設定を変更する場合には慎重に検証を行う必要があります。チューニングを行う場合は，パラメータの機能とサーバ上の作業負荷をよく理解した上で，ご使用中のアプリケーションに適しているかどうかを確認してください。

- 接続数の上限を調整します。接続数，CPU 使用率，プロセッサ・キューの長さの値がすべて高い場合，CPU にボトルネックが発生していることを示します。対応策として，接続数に上限を設定するか，CPU の処理能力を増強してください。
- ASP デバッグを無効にします。**AppAllowDebugging** と **AppAllowClientDebug** を両方とも **false** に設定し，サーバ側とクライアント側の両方で無効になっていることを確認します。
- **AspBufferingOn** を **true** に設定します。これにより，ASP 出力バッファがクライアントに送信される前にバッファの収集を行います。
- **AspProcessorThreadMax Metabase** プロパティは，IIS が生成できるワーカ・スレッドの数をプロセッサあたりの最大数で指定します。IIS が 1 つの ASP プロセスで生成できるワーカ・スレッドの最大数を計算するには，このプロパティ値に，サーバ上のプロセッサの数を掛けた積を求めます。このプロパティ値を小さくする場合は，パフォーマンスを監視してパフォーマンス低下がみられないことを確認する必要があります。パフォーマンス低下がみられた場合は，大きな値を設定してください。
- **AspRequestQueueMax Metabase** プロパティは，キュー内で待機できる ASP 要求の最大数を指定します。標準設定値は 3,000 ですが，最適な値はアプリケーションの動作によって異なります。要求の実行時間が非常に短く，キューの長さも短い場合，このプロパティ値に小さい値を設定することができます。

- 各 TCP 接続のキープ・アライブ状態が有効になっていること（**connection ＝ keep-alive**）を確認します。無効になっている場合，ファイルごとに新しい TCP 接続が必要になります。ファイルのサイズが小さい場合に IIS で HTTP Keep- Alives を有効にすると，ラウンド・トリップ時間が 2 倍になります。
- HTTP 圧縮を有効にすると，帯域幅を効率的に利用できます。
- すべてのイメージと HTML で HTTP Expire ヘッダを設定することにより，プロキシ・サーバとブラウザが Web サーバを呼び出す回数が低減します。
- 不要なファイル・コンテンツを削除します。不要な空白ライン，タブ，文字などを削除します。ファイル・サイズが大きいと，ファイルのネットワーク転送に時間がかかります。
- できる限り静的ファイルを使用することで，プロセッサの負荷を低減します。
- 複数のワーカ・プロセスの実行が可能なアプリケーション・プールである Web ガーデンを使用します。

# 第V部

## アプリケーション・サーバの監視

# 第10章

# WebLogic の監視

本章では，WebLogic の監視に関するベスト・プラクティスを紹介します。

**本章の内容**



## 概要

Oracle WebLogic は，業界トップクラスの J2EE アプリケーション・サーバの 1 つです。WebLogic のアーキテクチャとインフラストラクチャは高いパフォーマンスとスケーラビリティを念頭に設計されており，Web ベース・アプリケーションや Web サービスなどさまざまなタイプの分散アプリケーションのデプロイメントに対応します。さらに WebLogic は，Sun Microsystems の Java EE 5.0 仕様を完全に実装しています。これによって提供される標準 API セットを利用することによって，データベース，メッセージング・サービス，外部エンタープライズ・システムとの接続など幅広いサービスにアクセスする分散 Java アプリケーションの作成が可能になります。

したがって，パフォーマンスを的確に把握するためには，WebLogic アプリケーション・サーバ環境をよく理解する必要があります。本章では，WebLogic アプリケーション・サーバのアーキテクチャ，監視用カウンタ，チューニングが必要な主要部分について説明します。

# アーキテクチャ

WebLogic は，次のように複数の製品構成で提供されています。

- **WebLogic Server**：J2EE アプリケーション向けにコア・サービスとインフラストラクチャを提供します。
- **WebLogic Enterprise**：WebLogic Server と BEA Tuxedo ソフトウェアで構成されます。
- **WebLogic Express**：「軽量」バージョンです。J2EE 非準拠の WebLogic Server フレーバです。

本章では，WebLogic Server について説明します。

WebLogic アーキテクチャとデプロイメントを理解するためには，WebLogic Server ドメインに関する知識が必要です。

WebLogic Server ドメインとは，論理的に関連性のある WebLogic Server リソースをグループ化したものです。ドメインには管理サーバと呼ばれる特殊な WebLogic Server インスタンスと，管理対象サーバと呼ばれる WebLogic Server インスタンスが含まれます。WebLogic Server インスタンスは，管理サーバと管理対象サーバのいずれとしてもデプロイできます。

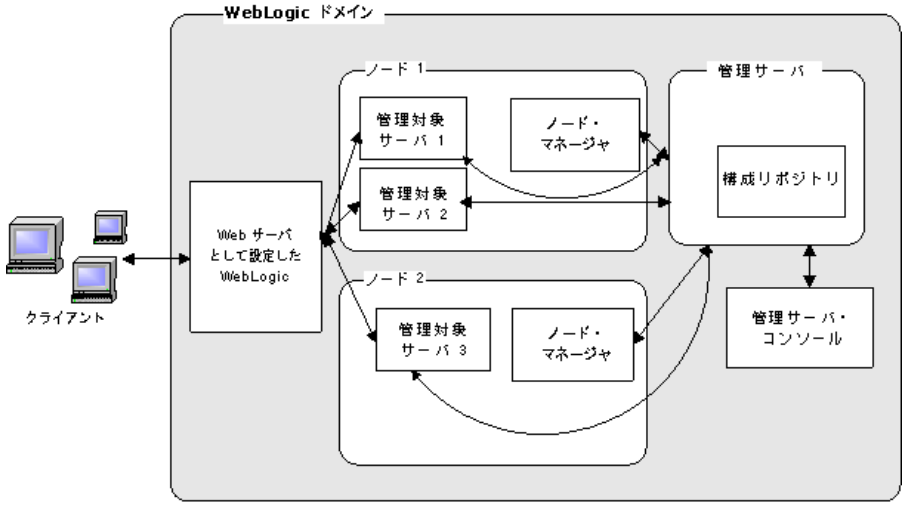
管理サーバがドメイン全体の管理と監視専用であるのに対して，管理対象サーバは，サーバにデプロイされているアプリケーション・プログラムのホスティングと実行を担当します。管理対象サーバは，専用の Java プロセス内で Oracle JRockit JVM を使用して稼働します。これは，管理サーバも同様です。

ドメインには，管理サーバと管理対象サーバに加えて，管理対象サーバとアプリケーションが必要とするリソースとサービスが含まれています。このようなリソースの 1 つにノード・マネージャがあります。ノード・マネージャは，論理エンティティではなくマシンと関連付けられます。ドメインの管理サーバは，ノード・マネージャを使用することによって，マシン上にデプロイされている管理対象サーバを制御します。

1 つの WebLogic Server を使って複数のドメインを作成および実行することも，複数の WebLogic Server を使用して 1 つのドメインを実行することも可能です。WebLogic Server のドメイン設定はパフォーマンスとスケーラビリティに影響を与えることが多いため，ユーザ環境でのドメイン設定を慎重に検討することが重要です。

それぞれの WebLogic Server は，HTTP 1.1 準拠の専用 HTTP リスナを使用する Web サーバとして設定できます。また，Apache，Microsoft IIS，Netscape の各種 Web サーバも使用できます。Web サーバの設定により，サーブレットや JSP が生成する動的コンテンツに加えて，静的 HTML コンテンツの要求も WebLogic Server で処理することができます。

次の図は，WebLogic ドメインを示しています。2 台のマシン/ノード上に管理対象サーバが 3 つデプロイされています。



# 監視

WebLogic Server 管理システムでは，Sun の Java Management Extension（JMX）規格に準拠した Managed Bean（MBean）を使用して，管理，稼働状態，パフォーマンスに関するデータを提供します。MBean のクエリには，JXM または SNMP を使用します。さらに WebLogic Server は，設定の変更やサブシステム障害などの情報をログ・ファイルに記録します。このログ・ファイルは，重大な障害の調査には役立ちますが，負荷を適用している状態での値はあまり参考になりません。

新しい WebLogic バージョンには，WebLogic Diagnostics Framework（WLDF）と呼ばれる診断フレームワークが付属します。WLDF は，上記の MBean を利用し，次のような機能を提供します。

- 障害発生後の分析に使用できる診断スナップショットを取得
- サーバ・インスタンスとアプリケーションからデータ・イベント，ログ・レコード，メトリックスをアーカイブ
- サーバと，そこで実行されているアプリケーションのインストルメンテーションを実行

製品ライフサイクル全体でのパフォーマンスをチェックするには，アプリケーションのアーキテクチャとデプロイメントを完全に理解しておく必要があります。特に，WebLogic サーバ上にデプロイされている J2EE アプリケーションを監視する場合には注意が必要です。たとえば，WebLogic サーバ・デプロイメントをクラスタ・モードで設定しないと，クラスタのモニタ機能は使用できません。

LoadRunner または ALM Performance Center を使用して WebLogic を監視する場合，HP SiteScope WebLogic ソリューション・テンプレートの使用をお勧めします。SiteScope WebLogic ソリューションは，事前設定された監視カウンタ群を含む SiteScope WebLogic モニタと JMX モニタをベースに設計されています。JMX インタフェースを使って WebLogic を監視するので，WebLogic 管理者によるセキュリティ・アクセス設定が必要になります。

別の方法として，SiteScope WebLogic モニタを使って WebLogic 6.x，7.x，8.x を監視し，SiteScope JMX モニタを使って WebLogic 9.x または 10.x を監視することもできます。この場合，カウンタを手作業で設定する必要があります。詳細については，SiteScope のユーザ・ガイドを参照してください。

# 最も重要な WegLogic カウンタ

本項では，パフォーマンスと作業負荷を監視する重要なカウンタについて説明します。WebLogic にはこれ以外にも多数のカウンタが提供されています。監視するカウンタは，MBean から選択します。

カウンタは WebLogic MBean に基づいてエンティティごとに分類されます。

---

**注：**アプリケーション・サーバ上にインストールされている機能によって，使用できるカウンタは異なります。

---

## サーバ

WebLogic Server では，処理は実行キューに追加されます。実行キュー内の処理はスレッドに割り当てられ，スレッドが処理を実行します。

次のカウンタは，サーバが作業負荷を処理する能力を評価し，実行キューまたはスレッド・プールが潜在的なボトルネックになるかどうかを判断する際に役立ちます。

WebLogic Mbean：

| カウンタ | 説明 |
|---|---|
| **MBean: weblogic.management.runtime.ServerRuntimeMBean** | |
| OpenSocketsCurrentCount | このサーバ上でソケット多重化のために予約されてるソケットの現在の数。 |
| **MBean: weblogic.management.runtime.ExecuteQueueRuntimeMBean** | |
| ExecuteThreadCurrentIdleCount | キューに割り当てられているアイドル・スレッドの数。 |
| ExecuteThreadTotalCount | キューに割り当てられている実行スレッドの総数。 |
| PendingRequestCurrentCount | キュー内で待機している要求の数。 |

| カウンタ | 説明 |
|---|---|
| **MBean: weblogic.management.runtime.ThreadPoolRuntimeMBean** | |
| ExecuteThreadIdleCount | プール内にあるアイドル・スレッドの数。この値には，スタンバイ・スレッドとスタック・スレッドは含まれません。この値は，新しい処理が到着した時点ですぐに処理可能なスレッドの数を示します。 |
| ExecuteThreadTotalCount | プール内にあるスレッドの総数。 |
| PendingUserRequestCount | 優先キュー内で待機しているユーザ要求の数。優先キューでは，内部サブシステムとユーザの要求が待機します。この値には，ユーザ要求がすべて含まれます。 |
| QueueLength | 優先キュー内で待機している要求の数。この値は，内部システム要求とユーザ要求の合計です。 |
| Throughput | 完了した要求数の 1 秒あたりの平均値。 |
| StandbyThreadCount | スタンバイ・プール内にあるスレッドの数。スタンバイ・スレッドとは，現在の作業負荷の処理には必要ない余剰スレッドを指し，スタンバイ・プールに追加されます。余剰スレッドは，必要に応じてアクティブ化されます。 |

## EJB

Enterprise JavaBeans は，ビジネス・ロジックをカプセル化するサーバ側のコンポーネントです。EJB はパフォーマンス・ボトルネックになりやすいため，監視が必要です。

EJB には，主にセッション Bean とメッセージ駆動型 Bean の 2 つのタイプがあり，セッション Bean はステートフルまたはステートレスのいずれかになります。

WebLogic MBean：

| カウンタ | 説明 |
|---|---|
| **MBean: weblogic.management.runtime.EJBCacheRuntimeMBean**<br>エンティティ Bean とステートフル Bean のキャッシュ・カウンタを監視します。 | |
| ActivationCount | この EJB Home から，アクティブ化された Bean の総数。 |
| CacheAccessCount | キャッシュから Bean にアクセスを試行した回数の合計。<br>**注：**実行中のサーバでは，CacheHitCount と CacheMissCount の合計が CacheAccessCount と等しくならないことがあります。これは，メトリックスの取得には複数の呼び出しが使用され，呼び出しごとに数が変わることがあるためです。 |
| CachedBeansCurrentCount | この EJB Home からの Bean の中で現在 EJB キャッシュ内にある Bean の総数。 |
| CacheMissCount | キャッシュから Bean へのアクセス試行が失敗した回数の合計。<br>**注：**実行中のサーバでは，CacheHitCount と CacheMissCount の合計が CacheAccessCount と等しくならないことがあります。これは，メトリックスの取得には複数の呼び出しが使用され，呼び出しごとに数が変わることがあるためです。 |

| カウンタ | 説明 |
|---|---|
| PassivationCount | この EJB Home からの Bean の中でパッシブ化された Bean の総数。 |
| **MBean: weblogic.management.runtime.EJBLockingRuntimeMBean** | |
| LockEntriesCurrentCount | 現在ロックされている Bean の数。 |
| LockManagerAccessCount | Bean 上でロックの取得を試行した回数の合計。この値には，クライアントのために，すでにロックされている Bean に対してロックを試行した回数も含まれます。 |
| TimeoutTotalCount | Bean のロックを待機していてタイムアウトが発生したスレッドの現在の数。 |
| WaiterCurrentCount | Bean のロックを待機しているスレッドの現在の数。 |
| **MBean: weblogic.management.runtime.EJBPoolRuntimeMBean**<br>エンティティ Bean，メッセージ駆動型 Bean，ステートレス Bean の EJB インスタンスを監視します | |
| AccessTotalCount | フリー・プールからインスタンスを取得しようとした試行回数の合計。 |
| BeansInUseCurrentCount | 現在使用されているフリー・プールの Bean インスタンスの数。 |
| DestroyedTotalCount | アプリケーション以外の例外がスローされたために，このプールの Bean インスタンスが破棄された回数の合計。 |
| MissTotalCount | フリー・プールからインスタンスを取得しようとして失敗した回数の合計。プール内に使用可能なインスタンスがないと，プールから Bean を取得できません。 |

| カウンタ | 説明 |
|---|---|
| PooledBeansCurrentCount | フリー・プール内で使用可能な Bean インスタンスの現在の数。 |
| TimeoutTotalCount | フリー・プールから使用可能な Bean インスタンスを取得しようと待機していてタイムアウトしたスレッドの数の合計。 |
| WaiterCurrentCount | フリー・プールから使用可能な Bean インスタンスを取得しようと待機している現在のスレッドの数。 |
| **MBean: weblogic.management.runtime.EJBTransactionRuntimeMBean**<br>エンティティ Bean，メッセージ駆動型 Bean，ステートレス Bean，ステートフル Bean のトランザクション・カウンタを監視します | |
| TransactionsCommittedTotalCount | この EJB についてコミットされているトランザクションの総数。 |
| TransactionsRolledBackTotalCount | この EJB についてロールバックされたトランザクションの総数。 |
| TransactionsTimedOutTotalCount | この EJB についてタイムアウトしたトランザクションの総数。 |

## サーブレット

WebLogic MBean：

| カウンタ | 説明 |
|---|---|
| **MBean: weblogic.management.runtime.ServletRuntimeMBean** | |
| ExecutionTimeAverage | サーブレットが作成されてから，そのサーブレットのすべての呼び出し時間の平均値。 |

## JRockit

このカウンタを使用できるのは，サーバで JRockit 仮想マシンを稼働する場合のみです。これは，アプリケーションのパフォーマンスとチューニングの両方で役立つカウンタです。

| カウンタ | 説明 |
|---|---|
| **MBean: weblogic.management.runtime.JRockitRuntimeMBean** | |
| UsedHeap | 仮想マシンが現在使用している Java ヒープ・メモリの容量（バイト単位）。 |
| UsedPhysicalMemory | ホスト・コンピュータが現在使用している物理メモリの容量（バイト単位）。この値は，VM のみではなく，コンピュータ上のすべてのプロセスが使用するメモリ容量を示します。 |
| TotalNurserySize | 保護領域に現在割り当てられているメモリの容量（バイト単位）。<br>保護領域とは，VM がほとんどのオブジェクトに割り当てる Java ヒープの領域を指します。世代別のガーベジ・コレクションは，ヒープ全体でガーベジ・コレクションを行うのではなく，保護領域のみが対象となります。ほとんどのオブジェクトは世代が若い間に終了するため，ヒープ全体ではなく保護領域のみでガーベジ・コレクションを行う方法でも十分に機能します。<br>世代別のガーベジ・コレクションを実行しない場合，保護領域のサイズは 0 になります。 |
| AllProcessorsAvgLoad | ホスト・コンピュータ上のすべてのプロセッサの平均負荷を示すスナップショット。この値は，コンピュータ上にプロセッサが 1 基のみ搭載されている場合，JVM Processor Load と同じ値になります。<br>double 型で返されるので，1.0 は 100% の負荷（アイドル時間なし），0.0 は 0% の負荷（アイドル時間のみ）を示します。 |

| カウンタ | 説明 |
| --- | --- |
| JVMProcessorLoad | VM がホスト・コンピュータ上のすべてのプロセッサに与える負荷を示すスナップショット。ホストにプロセッサが複数搭載されている場合，この値は平均負荷のスナップショットを示します。<br>double 型で返されるので，1.0 は 100% の負荷（アイドル時間なし），0.0 は 0% の負荷（アイドル時間のみ）を示します。 |
| TotalNumberOfThreads | すべてのプロセッサについて，VM で現在実行されている Java スレッド（デーモンと非デーモン）の数。 |
| NumberOfDaemonThreads | すべてのプロセッサについて，VM で現在実行されているデーモン Java スレッドの数。 |

## JDBC 接続プール

Java Database Connectivity（JDBC）は，データベースとのインタフェースと SQL ステートメント実行を行う標準 Java API です。

データベースはパフォーマンス・ボトルネックになることが多いため，あらゆる角度から慎重に監視する必要があります。ここでは，JDBC 接続プールに関するカウンタを紹介します。このカウンタを使用することによって，データベースに負荷がかかった場合の稼働状態を的確に把握することができます。

| カウンタ | 説明 |
| --- | --- |
| **MBean: weblogic.management.runtime.JDBCDataSourceRuntimeMBean** | |
| ActiveConnectionsAverageCount | データソースのこのインスタンス内でアクティブになっている接続の数の平均値。アクティブな接続とは，アプリケーションが使用している接続を指します。 |
| ActiveConnectionsCurrentCount | アプリケーションが現在使用している接続の数。 |

| カウンタ | 説明 |
| --- | --- |
| ConnectionDelayTime | データベースへの物理的な接続の確立にかかる時間の平均値（ミリ秒単位）。この値は，接続にかかる時間の合計を接続数で割って計算します。 |
| CurrCapacity | データソース内の接続プール内にある JDBC 接続の現在の数。 |
| LeakedConnectionCount | リークした接続の数。リークした接続とは，データソースから予約された後，close() 呼び出しによってデータソースに返されなかった接続を指します。 |
| NumAvailable | このデータソースで現在使用可能（使用中でない）データベース接続の数。 |
| NumUnavailable | データソースのこのインスタンスにおいて，現在使用可能でない（システムが使用中またはテスト中）データベース接続の数。 |
| PrepStmtCacheHitCount | キャッシュのステートメントを使用した回数の累積値。 |
| PrepStmtCacheMissCount | キャッシュのステートメントではステートメント要求を満たせなかった回数。 |
| WaitingForConnectionCurrentCount | データベース接続を待機している接続要求の数。 |

## JMS

WebLogic JMS は，エンタープライズ・クラスのメッセージ・システムであり，WebLogic Server プラットフォームと緊密に統合されています。

次のカウンタを使用できるのは，アプリケーションが WebLogic JMS を使用する場合のみです。このカウンタは，JMS サーバがボトルネックになっているかどうかを判断する際に非常に役立ちます。

WebLogic MBean：

| カウンタ | 説明 |
|---|---|
| **MBean: weblogic.management.runtime.JMSRuntimeMBean** | |
| ConnectionsCurrentCount | WebLogic Server に対する現在の接続数。 |
| **MBean: weblogic.management.runtime.JMSServerRuntimeMBean** | |
| BytesCurrentCount | この JMS サーバ上に現在格納されているバイト数。この値には，保留中のバイトは含まれません。 |
| BytesPageableCurrentCount | すべてのメッセージにおいて，現在ページアウト可能だがページアウトされていないバイト数の合計値。JMS サーバは，この値を MessageBufferSize パラメータ未満に抑えるように調整します。 |
| BytesPendingCount | この JMS サーバ上で現在保留中（受信確認またはコミットが完了していない）のバイト数。保留中のバイト数は，現在のバイト数よりも大きくなります。 |
| BytesReceivedCount | 前回のリセット以降，この JMS で受信されたバイト数。 |
| DestinationsCurrentCount | この JMS サーバの宛先の現在の数。 |
| MessagesCurrentCount | この JMS サーバ上に現在格納されているメッセージの数。この値には，保留中のメッセージは含まれません。 |

| カウンタ | 説明 |
| --- | --- |
| MessagesPageableCurrentCount | この JMS サーバで, 現在ページング可能な状態だがページアウトされていないメッセージの数。 |
| MessagesPendingCount | この JMS サーバ上で現在保留中（受信確認またはコミットが完了していない状態）のメッセージの数。保留中のメッセージの数は，現在のメッセージの数よりも大きくなります。 |
| MessagesReceivedCount | 前回のリセット以降，この宛先で受信されたメッセージの数。 |
| SessionPoolsCurrentCount | この JMS サーバ上で現在インスタンス化されているセッション・プールの数。 |

## JTA

トランザクション管理は WebLogic の基本機能の 1 つであり，高い精度と一貫性が確保される方法でデータベースの変更を処理します。

次のカウンタは，サーバとアプリケーションがどの程度の作業負荷に対応可能かを判断する際に役立ちます。

---

**ヒント：**トランザクションのロールバック率を確認してください。想定よりもロールバック率が高い場合，ロールバックの原因を究明し，オペレーティング・システム，アプリケーション・サーバ，データベース・サーバ，LoadRunner のトランザクションで測定された他のカウンタとの関連を調べてください。

---

| カウンタ | 説明 |
|---|---|
| **MBean: weblogic.management.runtime.JTARuntimeMBean** | |
| TransactionTotalCount | 処理されたトランザクションの総数。この値には，コミットされたトランザクション，ロールバックされたトランザクション，ヒューリスティックなトランザクションが含まれます。 |
| TransactionCommittedTotalCount | コミットされたトランザクションの数。 |
| TransactionRolledBackTotalCount | ロールバックされたトランザクションの数。 |
| TransactionRolledBackTimeoutTotalCount | タイムアウトが原因でロールバックされたトランザクションの数。 |
| TransactionRolledBackResourceTotalCount | リソース・エラーが原因でロールバックされたトランザクションの数。 |
| TransactionRolledBackAppTotalCount | アプリケーション・エラーが原因でロールバックされたトランザクションの数。 |

| カウンタ | 説明 |
| --- | --- |
| TransactionRolledBackSystemTotalCount | 内部システム・エラーが原因でロールバックされたトランザクションの数。 |
| TransactionHeuristicsTotalCount | ヒューリスティック状態で完了したトランザクションの数。 |
| TransactionAbandonedTotalCount | 破棄されたトランザクションの数。 |
| AverageCommitTime | サーバがトランザクションをコミットするのにかかった時間の平均値（ミリ秒単位）。 |
| ActiveTransactionsTotalCount | サーバ上のアクティブ・トランザクションの数の合計。 |

## 最適化とチューニング

パフォーマンスの問題を解決する上で，最適化とチューニングは重大な役割を果たします。アプリケーション・コードの最適化が必要になる場合がほとんどですが，チューニングで環境を効率化することによってパフォーマンスが格段に向上するケースもあります。

本項では，このようなチューニング方法をいくつか紹介します。ここには，WebLogic アプリケーション・サーバに関する内容と，一般的なアプリケーション・サーバに適用できる内容が含まれています。また，ここで紹介する方法以外にも，アプリケーションのパフォーマンスで高い効果を発揮するチューニング方法が多数存在します。

チューニングには，テストや分析を繰り返し行う必要があり，時間がかかります。また，設定を変更する場合には慎重に検証を行う必要があります。チューニングを行う場合は，パラメータの機能とサーバ上の作業負荷をよく理解した上で，ご使用中のアプリケーションに適しているかどうかを確認してください。

### プール・サイズのチューニング

EJB，JDBC，スレッドに関連するプールを適切なサイズにチューニングすると，サーバの容量が増大しパフォーマンスも向上します。プールのチューニングを行う場合は，これまでに説明したカウンタを監視し，待機時間や LoadRunner のトランザクション応答時間に注意してください。応答時間を適切に調整することが重要です。

### Prepared Statement Cache の使用

準備されたステートメント・キャッシュは，コンパイル済みの SQL ステートメントをメモリ内に格納します。これにより，同じステートメントを後で使用するときに，データベースへのラウンドトリップを回避できます。

### JVM のチューニング

- 収集アルゴリズムとして, 同時と並行のどちらがアプリケーションに適しているかを検討します。
- 最適なヒープ・サイズを検討します。
  - ピーク時の作業負荷でのアプリケーションの動作を監視します。
  - 収集を行う頻度を分析します。頻度が高すぎると空きメモリ容量が少なくなり，アプリケーション・コードの最適化が必要になることがあります。
  - 完全な GC の実行にかかる時間を分析します。5 秒以上かかる場合は，ヒープ・サイズを小さくします。
  - 平均メモリ使用率を分析します。完全な GC の実行後のヒープの空きが 85% ある場合，ヒープ・サイズを縮小できます。

### 実行キュー

キューの長さが小さく CPU 使用率が低い場合，スレッドの数を増やします。これにより，CPU 使用率が高くなります。

### 一般的な内容

- HTML ページ，画像，CSS ファイル，JavaScript ファイルなど静的コンテンツは必ず Web サーバで処理します。これにより，アプリケーション・サーバ・マシンで消費される CPU 時間が低減し，他のジョブへの割り当てが増大します。
- WebLogic クラスタを使用して，スケーラビリティと可用性を向上します。

# 第11章

# WebSphere の監視

本章では，WebSphere プラットフォームの監視に関するベスト・プラクティスを紹介します。

**本章の内容**



## 概要

IBM WebSphere アプリケーション・サーバは，IBM WebSphere プラットフォームの主力製品であり，業界トップクラスの J2EE アプリケーション・サーバの 1 つです。WebSphere のアーキテクチャとインフラストラクチャは高いパフォーマンスとスケーラビリティを念頭に設計されており，Web ベース・アプリケーションや Web サービスなどさまざまなタイプの分散アプリケーションのデプロイメントに対応します。さらに WebSphere は，Sun Microsystems の Java EE 5.0 仕様を完全に実装しています。これによって提供される標準 API セットを利用することによって，データベース，メッセージング・サービス，外部エンタープライズ・システムとの接続など幅広いサービスにアクセスする分散 Java アプリケーションの作成が可能になります。

したがって，パフォーマンスを的確に把握するためには，WebSphere アプリケーション・サーバ（WAS）をよく理解する必要があります。本章では，WebSphere アプリケーション・サーバのアーキテクチャ，監視用カウンタ，チューニングが必要な主要要素について説明します。

# アーキテクチャ

WebSphere アプリケーション・サーバは，次の 5 つのエディションで提供されています。

- **WebSphere Application Server Network Deployment :** 高度なパフォーマンスおよび管理機能を搭載することによって，ミッション・クリティカルなアプリケーション向けにほぼ連続可用性を実現します。
- **WebSphere Application Server for z/OS :** Network Deployment エディションと類似した機能を搭載した z/OS およびユーザ向けのエディションであり，z/OS Workload Manager を備えています。
- **WebSphere Application Server :** Java EE 5 に準拠し，シングル・サーバ環境において高いスケーラビリティと簡単な管理機能を提供します。
- **WebSphere Application Server Express :** WebSphere Application Server エディションの軽量版です。
- **WebSphere Application Server Community Edition :** オープン・ソースの Apache Geronimo をベースにした軽量版の Java EE 5 アプリケーション・サーバです。

WebSphere Application Server ファミリのすべてのエディションは，同じアーキテクチャ構造をベースに，それぞれが異なる機能，プラットフォームの互換性，ライセンス体系を提供します。

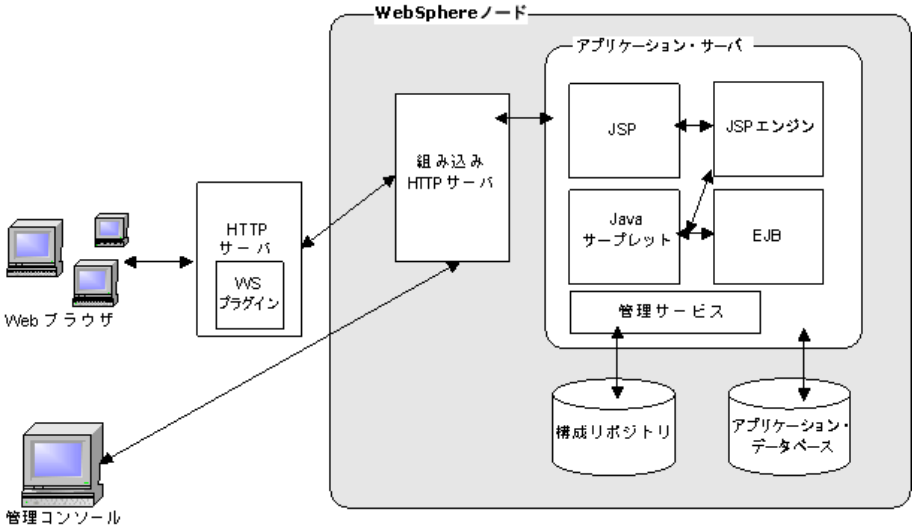
WebSphere Application Server はセル，ノード，サーバという概念に基づいて設計されています。セルとノードは，Network Deployment 環境へのアクセスで重要な役割を果たします。

- **サーバ :** サーバは，実際のコード実行を担当します。構成によって，アプリケーション・サーバと JMS サーバなどいくつかのタイプがあります。サーバはそれぞれが専用の JVM 上で稼働します。
- **ノード :** ノードとは，WebSphere で管理するサーバ・プロセスの論理的なグループであり，グループ内のプロセスは同じ構成と操作コントロールを共有します。一般的に，ノードは WebSphere Application Server の単一の物理インストールと関連付けられています。
- **セル :** セルは，ノードを 1 つの管理ドメインにグループ化したものです。

一般的な WebSphere セルには，1 つのノード上にインストールされたソフトウェア・コンポーネントや，スケーラビリティや信頼性を向上する目的で複数ノードに分散されたソフトウェア・コンポーネントが含まれます。セルには次のようなコンポーネントが含まれます。

- HTTP サービスを提供する Web サーバ
- アプリケーション・データを格納するデータベース・サーバ
- WebSphere Application Server（WAS）

次の図は，単一の WebSphere ノードで構成されるアーキテクチャを示します。



# 監視

WebSphere Application Server は，パフォーマンス監視インフラストラクチャ（PMI）を提供します。PMI は，サーバ・サイドの監視インフラストラクチャであり，クライアント・サイドの API を提供します。PMI を使用することによって，アプリケーション・サーバの全体的な稼働状態とパフォーマンスを監視できます。パフォーマンス・データは，JMX を使って提供されます。

---

**注：**PMI を有効にするには，WebSphere 管理コンソールを使用します。

---

製品ライフサイクル全体でのパフォーマンスをチェックするには，アプリケーションのアーキテクチャとデプロイメントを完全に理解しておく必要があります。特に，WebSphere サーバ上にデプロイされている J2EE アプリケーションを監視する場合には注意が必要です。たとえば，Web サービスのカウンタを監視できるのは，Web サービスを使用するアプリケーションに限定されます。

LoadRunner または ALM Performance Center を使用して WebSphere を監視する場合，HP SiteScope WebSphere ソリューション・テンプレートの使用をお勧めします。ソリューション・テンプレートには，管理用のカウンタ群が事前設定されています。

また，SiteScope WebSphere モニタを使用することも可能です。この機能を使用するには，カウンタを手作業で設定する必要があります。詳細については，SiteScope のユーザ・ガイドを参照してください。

# 最も重要なカウンタ

本項では，パフォーマンスと作業負荷を監視する重要なカウンタについて説明します。WebSphere ではこれ以外にも多数のカウンタが提供されています。カウンタは，SiteScope モニタの設定時に選択できます。

次のカウンタは，IBM WebSphere のカテゴリに従って分類されています。

---

**注：** アプリケーション・サーバ上にインストールされている機能によって，使用できるカウンタは異なります。

---

## Enterprise Java Beans

| カウンタ | キー | 説明 |
|---|---|---|
| ReadyCount | beanModule.readyCount | 同時に Ready 状態になる Bean（エンティティとセッション）の数。バージョン 3.5.5 以降と 4.0 では，このカウンタの名称は **concurrent active** です。 |
| LiveCount | beanModule.concurrentLives | 同時に稼働状態になる Bean の数。 |
| MethodResponseTime | beanModule.avgMethodRt | Bean メソッド（ホーム，リモート，ローカル）の平均応答時間（ミリ秒単位）。 |
| ActiveMethodCount | beanModule.activeMethods | 同時にアクティブになるメソッドの数（同時に呼び出されるメソッドの数）。 |
| MessageCount | beanModule.messageCount | Bean の onMessage メソッド（メッセージ駆動型 Bean）に送信されたメッセージの数。 |
| MessageBackoutCount | beanModule.messageBackoutCount | Bean の onMessage メソッド（メッセージ駆動型 Bean）への送信に失敗したメッセージの数。 |
| PooledCount | beanModule.poolSize | プール内にあるオブジェクト（エンティティおよびステートレス）の数。 |
| WaitTime | beanModule.avgSrvSessionWaitTime | プールから ServerSession を取得するのにかかった平均時間（メッセージ駆動型 Bean）。 |

## JDBC 接続プール

| カウンタ | キー | 説明 |
|---|---|---|
| Concurrent waiters | connectionPoolModule.concurrentWaiters | 接続を現在待機しているスレッドの数。 |
| Faults | connectionPoolModule.faults | 接続プール内で発生したエラー（タイムアウトなど）の総数。 |
| Percent used | connectionPoolModule.percentUsed | プールの平均使用率（パーセンテージ)。 |

## Java 仮想マシン（JVM）

| カウンタ | キー | 説明 |
|---|---|---|
| FreeMemory | jvmRuntimeModule.freeMemory | JVM ランタイム内の空きメモリ。 |
| ProcessCpuUsage | jvmRuntimeModule.cpuUsage | Java 仮想マシンの CPU 使用率（パーセンテージ)。 |
| UsedMemory | jvmRuntimeModule.usedMemory | JVM ランタイム内の使用メモリ。 |

## Servlet Session

| カウンタ | キー | 説明 |
|---|---|---|
| ActiveCount | servletSessionsModule.activeSessions | 同時にアクティブになるセッションの数。WebSphere Application Server が要求を処理している間，セッションはアクティブになります。 |
| LiveCount | servletSessionsModule.liveSessions | メモリ内に現在キャッシュされているローカル・セッションの数。 |

## トランザクション

| カウンタ | キー | 説明 |
|---|---|---|
| ActiveCount | ransactionModule.activeGlobalTrans | 同時にアクティブになるグローバル・トランザクションの数。 |
| LocalActiveCount | transactionModule.activeLocalTrans | 同時にアクティブになるローカル・トランザクションの数。 |
| RolledbackCount | transactionModule.globalTransRolledBack | ロールバックされたグローバル・トランザクションの総数。 |
| LocalRolledbackCount | transactionModule.localTransRolledBack | ロールバックされたローカル・トランザクションの数。 |
| GlobalTimeoutCount | transactionModule.globalTransTimeout | タイムアウトしたグローバル・トランザクションの数。 |
| LocalTimeoutCount | transactionModule.localTransTimeout | タイムアウトしたローカル・トランザクションの数。 |

## スレッド・プール

| カウンタ | キー | 説明 |
|---|---|---|
| ActiveCount | threadPoolModule.activeThreads | 同時にアクティブになるスレッドの数。 |
| PoolSize | threadPoolModule.poolSize | プール内にあるスレッドの数の平均値。 |
| PercentMaxed | threadPoolModule.percentMaxed | すべてのスレッドの使用時間の平均値（パーセンテージ）。 |
| DeclaredthreadHungCount | threadPoolModule.declaredThreadHung | 応答停止を宣言したスレッドの数。 |

## Web アプリケーション

| カウンタ | キー | 説明 |
|---|---|---|
| ConcurrentRequests | webAppModule.servlets.concurrentRequests | 同時処理されている要求の数。 |
| ServiceTime | webAppModule.servlets.responseTime | サーブレット要求の応答時間（ミリ秒）。 |
| ConcurrentRequests | webAppModule.url.concurrentRequests | サーブレットと関連付けられいる URI について，同時処理されている要求の数。 |
| ServiceTime | webAppModule.url.responseTime | サーブレットと関連付けられている URI の平均サービス応答時間（ミリ秒）。 |

## システム

| カウンタ | キー | 説明 |
|---|---|---|
| CPUUsageSinceLast Measurement | systemModule.cpuUtilization | 前回のデータ収集以降の平均システム CPU 使用率。<br>**注：**<br>➤ 最初の呼び出しでは, 初期化を行う必要があるため，0 など無効な値が返されます。その後の呼び出しでは，期待される値が返されます。<br>➤ SMP マシンでは，すべての CPU の平均使用率が返されます。 |
| FreeMemory | systemModule.freeMemory | システム上で使用可能な実メモリの空き容量。<br>**注：**<br>➤ 多くのオペレーティング・システムは, 割り当てられていないメモリの一部を追加の I/O バッファとして使用するので, 割り当てられていない実メモリは使用可能な実メモリ容量の下限値でしかありません。<br>➤ 解放可能なバッファ・メモリの正しい容量は, プラットフォームと実行中のアプリケーションによって変動します。 |

# 最適化とチューニング

パフォーマンスの問題を解決する上で，最適化とチューニングは重大な役割を果たします。アプリケーション・コードの最適化が必要になる場合がほとんどですが，チューニングで環境を効率化することによってパフォーマンスが格段に向上するケースもあります。

本項では，このようなチューニング方法をいくつか紹介します。ここには，WebSphere Application Server に関する内容と，一般的なアプリケーション・サーバに適用できる内容が含まれています。また，ここで紹介する方法以外にも，アプリケーションのパフォーマンスで高い効果を発揮するチューニング方法が多数存在します。

チューニングには，テストや分析を繰り返し行う必要があり，時間がかかります。また，設定を変更する場合には慎重に検証を行う必要があります。チューニングを行う場合は，パラメータの機能とサーバ上の作業負荷をよく理解した上で，ご使用中のアプリケーションに適しているかどうかを確認してください。

### プール・サイズのチューニング

EJB，JDBC，スレッドに関連するプールを適切なサイズにチューニングすると，サーバの容量が増大しパフォーマンスも向上します。プールをチューニングするには，関連のカウンタを監視します（詳細については，203ページ「最も重要なカウンタ」を参照してください）。特に，同時要求の数，待機の数，LoadRunner トランザクションの応答時間に注意してください。誤った設定を行わないようにするには，アプリケーションの設計を考慮する必要があります。

### Prepared Statement Cache の使用

準備されたステートメント・キャッシュは，コンパイル済みの SQL ステートメントをメモリ内に格納します。これにより，同じステートメントを後で使用するときに，データベースへのラウンドトリップを回避できます。準備されたステートメント・キャッシュのサイズは，同時処理される要求の数とアプリケーションの設計に基づいて決定します。

### JVM のチューニング

- 収集アルゴリズムとして，同時と並行のどちらがアプリケーションに適しているかを検討します。
- 最適なヒープ・サイズを検討します。
  - ピーク時の作業負荷でのアプリケーションの動作を監視します。
  - 収集を行う頻度を分析します。頻度が高すぎると空きメモリ容量が少なくなり，アプリケーション・コードの最適化が必要になることがあります。
  - 完全な GC の実行にかかる時間を分析します。5 秒以上かかる場合は，ヒープ・サイズを小さくします。
  - 平均メモリ使用率を分析します。完全な GC の実行後のヒープの空きが 85% ある場合，ヒープ・サイズを縮小できます。

### 一般的な内容

- HTML ページ，画像，CSS ファイル，JavaScript ファイルなど静的コンテンツは必ず Web サーバで処理します。これにより，アプリケーション・サーバ・マシンで消費される CPU 時間が低減し，他のジョブへの割り当てが増大します。
- 不要な機能は無効化します。たとえば，アプリケーションで Web サービスのアドレス指定（WS-Addressing）を使用しない場合，この機能を無効にすることによってパフォーマンスが向上します。
- トランザクション・ログには高速ディスクが割り当てられていることを確認します。

# 第VI部

## データベース・リソースの監視

# 第12章

# データベース・リソースの監視について

最新のアプリケーションでは，ほとんどの場合，複数の階層からなるアーキテクチャで稼働する設計が採用されています。このようなアーキテクチャでは，アプリケーション機能は複数の階層にまたがっており，それぞれが専用のサーバ上で実行されます。このような層には，次のようなコンポーネントなどが含まれます。

- **ユーザ・インタフェース**：ユーザとアプリケーション間の通信を行います。
- **ビジネス層**：アプリケーション実行に必要なあらゆるビジネス規則に関連します。
- **データ層**：ビジネス・トランザクションの管理に必要なデータを処理します。

この構造には，クライアントが比較的軽量になる，サーバ・サイドのみのデプロイメントでよい，機能を分離できる，データベースへの直接アクセスがない，開発と運用を含むアプリケーション総コストを低く抑えることができる，などのメリットがあります。

その一方で，分散型の構造は設計が複雑になり，それぞれの層がパフォーマンスを低下させる要因になる可能性があります。中でも，データベース層からの応答時間の遅延は，パフォーマンスに関するエンド・ユーザからのクレームの原因となりやすい要因です。

データベースは，データ，クエリ，ロジックなど，常に変更されています。したがって，今日のデータ駆動型のアプリケーションでは，データベースの最適化は非常に重要な役割を果たします。

次に示すように，データベースには，アプリケーション全体のパフォーマンスに影響を与える要因が多数あります。

- アプリケーション開発で採用されたデータベース設計が不適切
- テーブル設計で採用された標準が不適切
- データベースのインデックスが非効率
- テーブル全体のパーティション分割が不適切
- クエリで採用されたロジックが非効率
- ストアド・プロシージャが不適切
- ストレージ・ハードウェアの設定が不適切
- データベース・サーバ・マシンが複数のアプリケーションに割り当てられている

# 第13章

# Oracle の監視

本章では，Oracle の監視に関するベスト・プラクティスを紹介します。

**本章の内容**



## 概要

Oracle データベースは，Oracle Corporation が開発したリレーショナル・データベース・マネジメント・システム（RDBMS）です。Oracle データベースには，可用性，スケーラビリティ，パフォーマンス，管理，セキュリティを向上する多彩な機能が備わっています。このような機能を備えた Oracle データベースはエンタープライズ・クラスの RDBMS として採用されており，RDBMS 部門をリードする製品となっています。

Oracle データベースは，さまざまな機能によってアプリケーション開発を包括的にサポートします。また，Java と .NET のいずれのデータにもアクセスできる機能も提供されています。

Oracle データベースでは，システムの規模に対応した複数のエディションが用意されています。

- **Standard Edition（SE）**：基本的なデータベース機能を備えています。一般的に，1～4 基の CPU を搭載したサーバ向けです。搭載 CPU が 4 基を超える場合は，Enterprise ライセンスが適用されます。SE にはメモリの上限はなく，Oracle RAC のクラスタ機能を使用できます。
- **Enterprise Edition（EE）**：Standard Edition の拡張版であり，特にパフォーマンスとセキュリティ機能が強化されています。4 基以上の CPU を搭載するサーバ向けです。EE にはメモリの上限はなく，Oracle RAC ソフトウェアのクラスタ機能を使用できます。
- **Standard Edition One**：Oracle 10g で登場したエディションであり，Standard Edition の一部機能が制限されています。1～2 基の CPU を搭載するシステム向けです。メモリの上限はありません。
- **Express Edition（Oracle Database XE）**：2005 年に登場したエディションであり，Windows と Linux プラットフォームで提供されています。サイズはわずか 150 MB です。シングル CPU での使用，ユーザ・データ領域の最大容量 4 GB という制限があります。インストール先となるサーバのメモリ容量に制限はありませんが，使用できる最大容量は 1 GB です。
- **Oracle Database Lite**：モバイル・デバイス用のエディションです。モバイル・デバイス上にデータベースの一部を読み込んでおき，それをサーバ・ベース環境と同期することができます。

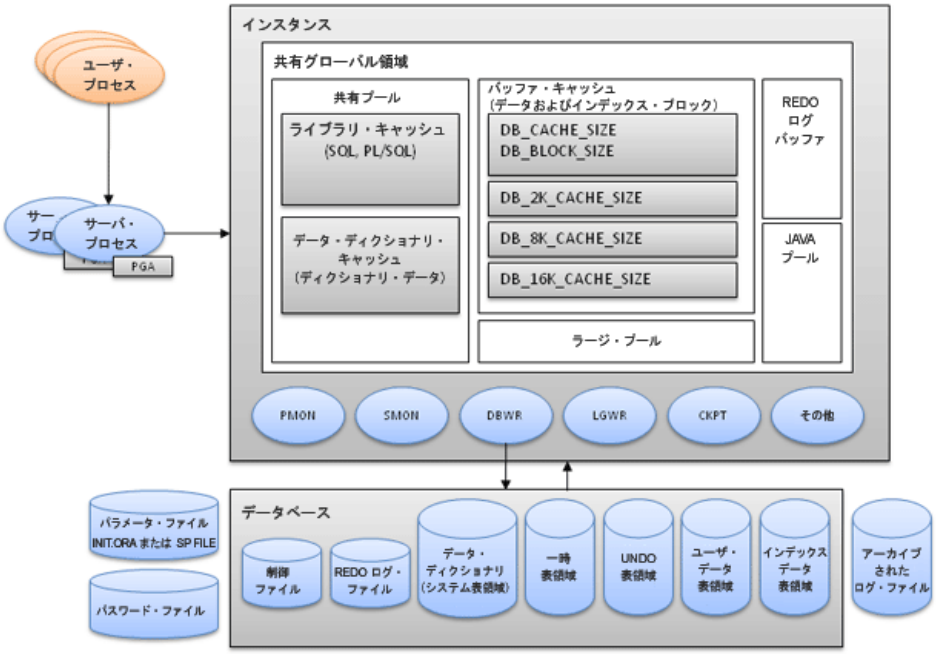
データベース層がアプリケーションのパフォーマンスに大きな影響を与えることはよく知られています。Oracle は世界で最も普及しているデータベースの 1 つであるため，Oracle 環境をパフォーマンスの点から理解することが重要です。本章では，Oracle データベースの高レベルなアーキテクチャと監視機能について説明します。また，重要な監視カウンタとチューニング方法も紹介します。

## アーキテクチャ

Oracle データベースは，インスタンスとデータ・ストレージで構成されます。インスタンスとは，オペレーティング・システム・プロセスとメモリ構造を組み合わせたものであり，ストレージと対話します。このメモリ構造は，システム・グローバル領域（GA）と呼ばれます。ストレージは，論理的に見ると表領域，物理的に見るとデータ・ファイルとして格納されます。表領域にはさまざまなタイプのメモリ・セグメントが含まれ，セグメントには 1 つまたは複数のエクステントが含まれます。エクステントには連続したデータ・ブロックが含まれ，データ・ブロックにはデータ・ストレージの基本単位が含まれます。物理的な構造を見ると，データ・ファイルには 1 つまたは複数のデータ・ブロックが含まれており，データ・ブロックのサイズはファイルによってさまざまです。

Oracle データベースでは，SYSTEM 表領域に格納されている管理情報を元に，コンピュータのデータ・ストレージを追跡します。SYSTEM 表領域にはデータ・ディクショナリが格納され，インデックスとクラスタも格納されます（標準設定）。データ・ディクショナリには，データベース内にあるすべてのユーザ・オブジェクトに関する情報が格納された特殊なテーブル群が含まれています。

次の図は，Oracle データベース・アーキテクチャを示します。図では，インスタンス・レベルでのメモリ構造と，ストレージ・レベルでのデータ・ファイルが示されています。




各 Oracle インスタンスは，共有メモリ領域である**システム・グローバル領域（SGA）**にデータと管理情報を格納します。インスタンスは，起動時に SGA を割り当て，シャットダウン時に割り当てを解除します。SGA には次のコンポーネントが格納されています。それぞれ固定長であり，インスタンスの起動時に設定されます。

- **バッファ・キャッシュ**：最近使用したデータ・ブロックが格納されています。同じデータが要求された場合にはディスクからではなくバッファ・キャッシュから取得できるので，I/O の回数が低減しパフォーマンスが向上します。
- **REDO ログ・バッファ**：データベースへの変更内容のログを示す REDO エントリが格納されています。システム障害時には，これに基づいてインスタンスを復元します。

- **共有プール**：ライブラリ・キャッシュの共有 SQL 領域やデータ・ディクショナリ内の内部情報など，共有メモリ構造が格納されています。共有プールへのメモリ割り当てが不足すると，パフォーマンス低下の原因になります。
  - **ライブラリ・キャッシュ**：共有 SQL 領域であり，各 SQL ステートメントの解析ツリーと実行プランが格納されています。解析と実行プランの作成に必要なメモリ容量を節約し，処理時間を短縮します。
  - **データ・ディクショナリ・キャッシュ**：ユーザ情報，権限，表の名前，データ型などの情報が格納されています。データ・ディクショナリは，SQL ステートメントの解析に使用されます。データ・ディクショナリがパフォーマンス・ボトルネックになると，すべての Oracle ユーザに影響が及びます。

**プログラム・グローバル領域（PGA）**は，サーバ・サイドのプロセスであり，クライアント・マシン上で実行されるユーザ・プロセスを処理します。PGA メモリ領域には，Oracle のサーバ・プロセスが使用するデータと管理情報が格納され，これにはユーザ・セッション，セッション変数，ソート，バインド変数などに関する情報が含まれます。

通常，Oracle はプロセスのグループをバックグラウンドと対話型で同時に実行し，データベースの監視やパフォーマンスの向上を図ります。次に，インスタンス・レベルで実行されるプロセスをいくつか紹介します。

- **データベース・ライタ・プロセス（DBWR）**：データをディスクに書き込みます。
- **ログ・ライタ・プロセス（LGWR）**：データをログに書き込みます。
- **システム監視プロセス（SMON）**：インスタンスの回復，一時セグメントの割り当て解除，空き領域のマージなどを行います。
- **プロセス・モニタ（PMON）**：プロセスの障害発生後，クリーンアップを行います。
- **チェックポイント・プロセス（CKPT）**：チェックポイントに関する信号の送信と，チェックポイントが発生したファイルの更新を行います。

Java プールが使用されるのは，Java コードがインスタンス・レベルで実行され，ラージ・プールがオプションの場合のみです。ラージ・プールを使用する場合，標準設定で共有プールに格納される情報の一部をラージ・プールに格納することによって，共有プールのオーバーヘッドを軽減します。

Oracle アーキテクチャでは，I/O 操作を最小限に抑えることによってパフォーマンスを最適化します。パフォーマンスの監視とチューニングでは，上記のような機能がユーザの環境で適切に設定されているかどうかを確認してください。

## 監視

Oracle では，パフォーマンスの監視とチューニング用のツールとユーティリティがいくつか提供されています。

- **自動データベース監視モニタ（ADDM）**：Oracle データベースを診断し，潜在的な問題があれば解決する方法を見つけます。ADDM は，自動ワークロード・リポジトリ（AWR）が統計データを収集した後に自動的に実行され，パフォーマンスの診断データを提供します。AWR は定期的に実行されるので，データベースのパフォーマンスを定期的に診断し，問題を検出することができます。ADDM では，次のような状態を潜在的な問題とみなします。
  - **CPU ボトルネック**：Oracle などのアプリケーションによってシステム CPU が制約を受けているか。
  - **メモリ構造の不足**：SGA，PGA，バッファ・キャッシュなどの Oracle メモリ構造が十分なサイズか。
  - **I/O 容量の問題**：I/O サブシステムが期待通りのパフォーマンスを発揮しているか。
  - **大きな負荷の原因となる SQL ステートメント**：システム・リソースを大量に消費する SQL ステートメントの有無。
  - **大きな負荷の原因となる PL/SQL 例外とコンパイル**，Java の使用方法。
  - **RAC 固有の問題**：グローバル・キャッシュ・ブロックとオブジェクト，相互に関連して遅延時間が発生していないか。

- **アプケーションが最適な方法で Oracle を使用していない**：非効率な接続管理，過剰な解析，アプリケーション・レベルでのロック競合などの問題が発生しているか。
- **データベース設定の問題**：ログ・ファイルのサイズが不適切，アーカイブの問題，過剰なチェックポイント，パラメータ設定が最適化されていない，などの問題があるか。
- **同時実行の問題**：バッファ・ビジー待機の問題が発生しているか。
- **さまざまな問題領域のホット・オブジェクトとトップ SQL**：

  ADDM と AWR は，パフォーマンスの問題の特定やチューニングのスタート・ポイントとして役立つデータを提示します。

- **Oracle Enterprise Manager**：Oracle 環境の管理に役立つシステム管理ツールです。Oracle 環境の監視とタスクの自動実行を行います。
- **SQL Trace**：各 SQL ステートメントに関するパフォーマンス情報を提供します。各ステートメントについて，次の統計情報を生成します。
  - 解析，実行，取得の回数
  - CPU と経過時間
  - 物理的な読み取りと論理的な読み取り
  - 処理した行数
  - ライブラリ・キャッシュのミス
  - 解析を実行したユーザ名
  - 各コミットとロールバック
- **TKProf**：SQL Trace の出力を判読可能な書式に変換するユーティリティです。SQL ステートメントのチューニングで高い威力を発揮します。また，SQL ステートメントの実行計画の作成や，データベースに統計データを格納する SQL スクリプトの作成にも使用できます。

Oracle は，さまざまな統計情報を表に格納します。この表は，Oracle SQL ステートメント・オプティマイザも使用します。次に例を示します。

- セッション統計，V$SESSTAT
- システム統計，V$SYSSTAT
- V$LATCH，V$BUFFER_POOL_STATISTICS

HP 監視ソリューションは，上記の表に格納されたデータを使用して，パフォーマンス・テストの実行中にデータにアクセスします。HP SiteScope Oracle Database Solution に組み込まれたカウンタを使用することをお勧めします。

## 最も重要な Oracle カウンタ

| カウンタ | 説明 |
|---|---|
| **sorts (disk) (V$SYSSTAT 1/sid) (absolute)** | ディスク書き込みが 1 回以上必要になったソート操作の回数。<br>ディスクへの I/O を必要とするソートは，リソースを大量に消費します。初期化パラメータ SORT_AREA_SIZE の値を大きくするとパフォーマンスが向上する場合があります。 |
| **sorts (memory) (V$SYSSTAT 1/sid) (absolute)** | 完全にメモリ内で実行され，ディスク書き込みが不要だったソート操作の回数。ソートをまったく行わない場合を除き，メモリ上でのソートのパフォーマンスが最も高くなります。通常，ソートは，表結合の SQL 操作で選択条件を指定すると発生します。 |
| **db block gets (V$SYSSTAT 1/sid) (absolute)** | INSERT，UPDATE，DELETE，SELECT FOR UPDATE において，バッファ・キャッシュ内でアクセスしたブロック数。ブロックの論理的読み取り（キャッシュから）を示します。論理読み取りは，必ず物理読み取りを伴います。物理読み取りの回数は少ない方が望ましいといえます。 |

| カウンタ | 説明 |
|---|---|
| **consistent gets (V$SYSSTAT 1/sid) (absolute)** | 通常のクエリ（update 句のない SELECT）において，バッファ・キャッシュ内でアクセスしたブロックの数。ブロックの論理的読み取り（キャッシュから）を示します。論理読み取りは，必ず物理読み取りを伴います。物理読み取りの回数は少ない方が望ましいといえます。 |
| **physical reads (V$SYSSTAT 1/sid) (absolute)** | ディスクから読み取ったデータ・ブロック数の合計。この値は，**physical reads direct** の値に，バッファ・キャッシュへのすべての読み取りを加えた数値です。物理読み取りの回数は少ない方が望ましいといえます。この値は，キャッシュ・ヒット率を計算する際に，論理読み取りの回数と比較する必要があります。論理読み取りは，database block gets の値と consistent gets の値の合計です。 |
| **physical writes (V$SYSSTAT 1/sid) (absolute)** | ディスクに書き込まれたデータ・ブロック数の合計。この値は，**physical writes direct** の値に，バッファ・キャッシュからのすべての書き込みを加えた数値です。 |
| **redo writes (V$SYSSTAT 1/sid) (absolute)** | LGWR が REDO ログ・ファイルに書き込んだ回数の合計。**redo blocks written** をこの値で割ると，書き込み 1 回あたりのブロック数を計算できます。 |
| **redo entries (V$SYSSTAT 1/sid) (absolute)** | REDO エントリには，INSERT，UPDATE，DELETE，CREATE，ALTER，DROP の各操作を使って行ったデータベースの変更内容を復元するために必要な情報が含まれています。REDO エントリは，必要に応じてデータベースの回復に使用されます。<br>REDO エントリは，成功した REDO 書き込みです。**Redo buffer allocation retries / Redo entries** で計算される割合が 1% を超えないようにします。 |
| **redo buffer allocation retries (V$SYSSTAT 1/sid) (absolute)** | REDO バッファ内の領域を割り当てるために必要になる試行回数の合計。試行が必要になるのは，REDO ライタが待機した場合，またはログの切り替えなどのイベントが発生した場合のいずれかです。<br>REDO バッファ割り当てエントリは，失敗した REDO 書き込みです。**Redo buffer allocation retries / Redo entries** で計算される割合が 1% を超えないようにします。 |

| カウンタ | 説明 |
| --- | --- |
| **redo log space requests (V$SYSSTAT 1/sid) (absolute)** | アクティブなログ・ファイルが一杯になったので，REDO ログ・エントリに割り当てるディスク領域を確保できるまで待機した回数。この領域は，ログの切り替えによって発生します。<br>SGA のサイズや作業負荷のコミット率に比べてログ・ファイルのサイズが小さいと，問題が発生する可能性があります。ログ切り替えでは，新しいログ・ファイルに切り替える前に，コミットされたダーティ・バッファをすべてディスクに書き込む必要があります。したがって，サイズの大きな SGA にダーティ・バッファが大量に格納されていて，REDO ログ・ファイルのサイズが小さい場合には, DBWR がダーティ・バッファをディスクに書き込むまで待機する必要があります。<br>また，V$SESSION_WAIT のログ・ファイル領域とログ・ファイル領域の切り替え待機イベントを確認してください。 |
| **parse count (hard) (V$SYSSTAT 1/sid) (absolute)** | 解析コール（実解析）の総数。ハード解析は，作業ヒープなどのメモリ構造を割り当ててから解析ツリーを構築するため，大量のメモリを消費します。<br>この値はできるだけ低く抑える必要があります。全体に占めるハード解析の割合の適正値は 20% 未満です。 |
| **parse count (total) (V$SYSSTAT 1/sid) (absolute)** | 解析コール（ハードおよびソフト）の総数。ソフト解析は，共有プール内にすでに存在するオブジェクトをチェックして，基になるオブジェクトでの許可が変更されていないかどうかを確認します。<br>全体に占めるハード解析の割合の適正値は 20% 未満です。 |
| **parse time cpu (V$SYSSTAT 1/sid) (absolute)** | 解析（ハードおよびソフト）に費やした CPU 時間の合計（10 ミリ秒単位）。 |
| **parse time elapsed (V$SYSSTAT 1/sid) (absolute)** | 解析の経過時間の合計（10 ミリ秒単位）。この値から **parse time cpu** の値を引くと，解析リソースの待機時間の合計を計算できます。 |

| カウンタ | 説明 |
|---|---|
| **CPU used by this session (V$SYSSTAT 1/sid) (absolute)** | ユーザ呼び出しの開始から終了までに，セッションが使用した CPU 時間（10 ミリ秒単位）。ユーザ呼び出しが 10 ミリ秒以内に完了する場合，ユーザ呼び出しの開始時間と終了時間は等しくなるので，0 ミリ秒になります。 |
| **bytes sent via SQL*Net to client (V$SYSSTAT 1/sid) (absolute)** | フォアグラウンド・プロセスからクライアントが送信したバイト数の合計。Oracle Net Service 経由で送信したデータ容量を示します。 |
| **bytes received via SQL*Net from client (V$SYSSTAT 1/sid) (absolute)** | Oracle Net Service 経由でクライアントが受信したバイト数の合計。Oracle Net Service 経由で送信したデータ容量を示します。 |
| **logons current (V$SYSSTAT 1/sid) (absolute)** | 現在のログインの合計数。V$SYSSTAT でのみ有効です。 |

以上のカウンタに加えて，表領域の使用率を監視することをお勧めします。空き領域が 2% 未満になった場合は，表領域のサイズを増やすことをお勧めします。

## 最適化とチューニング

パフォーマンスに問題が発生している場合，問題点を解消するためにはチューニングと最適化が必要です。アプリケーション・コードの最適化が必要になる場合がほとんどですが，チューニングで環境を効率化することによってパフォーマンスが格段に向上するケースもあります。

本項では，このようなチューニング方法をいくつか紹介します。ここでは，Oracle データベースに関する内容に加えて，データベース・サーバやその他のサーバに適用できる一般的な内容を掲載します。また，ここで紹介する方法以外にも，個々のアプリケーションで高い効果を発揮するチューニング方法が多数存在します。

チューニングには，テストや分析を繰り返し行う必要があり，時間がかかります。また，設定を変更する場合には慎重に検証を行う必要があります。チューニングを行う場合は，パラメータの機能とサーバ上の作業負荷をよく理解した上で，ご使用中のアプリケーションに適しているかどうかを確認してください。

- Oracle のコストベース・オプティマイザが実行中であることを確認します。
- オプティマイザで統計データを定期的に収集します。
- SQL ステートメントのチューニングを行います。
  - 問題のある SQL ステートメント（実行時間が長い SQL ステートメント）を特定
  - Oracle オプティマイザの統計をレビュー（コストベース・オプティマイザが実行中で，最新の統計が収集されていることを確認）
  - 実行計画をレビュー
  - SQL ステートメントを再構築（必要に応じて）
  - インデックスを再構築（必要に応じて）
  - 実行計画の経時的なメンテナンス
- SQL ステートメントでバインド変数を使用します。これにより，共有プールに格納されるカーソルの数を減らすことができます。
- インデックスの使用には注意が必要です。すべてのカラムにインデックスを付けるのではなく，クエリでアクセス頻度の高いカラムのみにインデックスを付けてください。
- 必要に応じた最適化を講じることにより，SQL オプティマイザの効果を高めます。この作業は，SQL ステートメントのパフォーマンス分析に基づいて行います。

- メモリ構造のサイズをチューニングします。共有プール，バッファ・キャッシュ，その他メモリ構造のサイズは，データベース・パフォーマンスに大きな影響を与えます。
  - 一般的な作業負荷をアプリケーションに適用します。
  - 待機時間, バッファ・ヒット率, システム・スワップとページングなどを監視します。
  - 次に，特にチューニングが必要となる重要なパラメータを紹介します。

| パラメータ | 説明 |
|---|---|
| **db_cache_size** | SGA 内のバッファ・キャッシュのサイズを指定します。 |
| **db_keep_cache_size** | オブジェクトは, ロードされると必ずこの領域に格納されます。このキャッシュには，アクセス頻度が高く, メモリ内に保持しておく必要があるオブジェクトが格納されます。たとえば，使用頻度が高くサイズの小さいルックアップ・テーブルなどが含まれます。このキャッシュは, DB_CACHE_SIZE パラメータで定義される標準設定キャッシュのサブセットです。データベースでは, DB_CACHE_SIZE の設定が必ず必要になります。 |
| **shared_pool_size** | 共有プールのサイズを指定します。 |
| **pga_aggregate_target** | インスタンスに接続されているすべてのサーバ・プロセスが使用可能なターゲット集計 PGA メモリを指定します。 |
| **log_buffer** | REDO ログ・バッファのサイズを指定します。 |
| **query_rewrite_enabled** | SQL ステートメントを実行前にリライトするかどうかを指定します。 |
| **cursor_sharing** | 同じカーソルを共有できる SQL ステートメントの種類を指定します。 |

| パラメータ | 説明 |
|---|---|
| **db_file_multiblock_read_count** | 表のフルスキャン中に発生する I/O を最小限に抑えるパラメータの 1 つです。シーケンシャル・スキャン中に 1 回の I/O 操作で読み取るブロック数の最大値を指定します。 |
| **hash_multiblock_io_count** | 1 回の I/O 操作で, ハッシュ結合が読み取り/書き込みを行う連続ブロックの数を指定します。 |

- I/O 操作の回数を抑えるには，バッファ・キャッシュ・ヒット率を高める必要があります。OLTP 環境での適正値は 80% 超であり，99% が最適な値です。
- ディクショナリ・キャッシュのヒット率は，90% 前後が適正値です。**MISS RATE %** カラムの **dc_table_grants**，**d_user_grants**，**dc_users** のエントリは 5% 未満が適正値です。
- Monitor Sorts は，メモリ内のソートとディスク内のソートを示します。ディスクとメモリの比率は 10 未満が適正値です。
- データベース競合を最小限に抑えます。ロックとラッチの発生状態を調査し，できるだけ低減します。
- 複雑で大規模なデータ・ウエアハウスで発生する作業負荷については, HP Oracle Database マシンを使用します。

# 第14章

# Microsoft SQL Server の監視

本章では，Microsoft SQL Server の監視に関するベスト・プラクティスを紹介します。

**本章の内容**

# 概要

Microsoft SQL Server は，最も広く普及しているデータベース・システムの 1 つです。小規模な部門タスクから世界的規模のデータベースまでに対応する高いスケーラビリティを誇ります。Microsoft SQL Server は，単なる「データベース」ではなく，完全なデータ・アーキテクチャ・ソリューションとして，データ・ストレージやデータの加工などあらゆる組織のニーズに対応します。このソリューションは，XML，電子メール/カレンダ，ファイル，ドキュメント，などさまざまなタイプのドキュメントを格納および管理する機能だけでなく，検索，クエリ，データ分析，レポート作成，データ統合，堅牢性に優れた同期など，対話型のデータ処理を行う多彩なサービスを提供します。また，サーバの SQL Server にアクセスしてデスクトップやモバイル・デバイスに接続するアプリケーションの記述では，Microsoft ベースやサードパーティ・ベンダが提供するさまざまなテクノロジを活用できます。

SQL Server は，あらゆる規模の組織のニーズに対応できるように，多数のエディションが提供されています。Express および Compact，Workgroup から Standard および Enterprise の各エディションは，ニーズに合わせてそれぞれ異なる機能セットを提供すると同時に，開発者およびエンド・ユーザ向けには同じレベルの機能を提供します。

SQL Server は，出荷時の設定のままでも高い機能性を発揮し，パフォーマンスの問題も発生しないという定評があります。その一方で，安価なハードウェアの登場やデータの爆発的な増大を背景に，出荷時の設定のみで発揮できるパフォーマンスには限度があることもわかってきました。したがって，パフォーマンス・エンジニアは，さまざまな監視機能を駆使してこのような問題を見つけ出す必要があります。ここでは，SQL Server のエディションの中でも，広く採用されている Enterprise と Standard について説明します。

# アーキテクチャ

SQL Server のインストール時に，標準的な Windows オブジェクトとカウンタに SQL Server のカウンタが追加されます。SQL Server のほとんどのコンポーネントのパフォーマンスはこのカウンタで監視できます。ただし，一般的な手順としては，まず最初に CPU 使用率，ディスク操作，メモリ管理，ネットワーク帯域幅などの Windows システム・リソースの監視を行います（第3章，「Windows の監視」を参照してください）。


このようなリソースはサーバの主要なハードウェア・コンポーネントであり，それぞれがユーザ要求の処理で使用されるので，監視が必要になります。また，このようなコンポーネントのパフォーマンスは，アプリケーション全体のパフォーマンスに直接的な影響を与えます。したがって，この 4 つの領域のいずれかで問題が発生すると，ユーザの不満へとつながります。SQL Server のパフォーマンスは，CPU の性能，使用可能なメモリ容量，ディスク・スループットに大きく依存します。これに対してクライアントのパフォーマンスは，ネットワーク・パフォーマンスに大きく依存します。プロセッサのビジー時間が常に 90% を超えると，作業要求はキューで待機することになり，パフォーマンス低下の原因になります。

SQL Server のパフォーマンスは，メモリに大きく影響されます。物理メモリが不足するとページ・ファイルを使用することになり，これがパフォーマンスの大幅低下につながることがあります。ディスクには機械的な機構が含まれているため，最も低速なコンポーネントだといえます。SQL Server はディスクからデータを取得する必要があるので，ディスク I/O の遅延はシステム全体のパフォーマンスに影響を与えます。さらに，データベース・パフォーマンスを最適化しても，ネットワークに遅延が発生している場合やパケット・ロス率が高いために再転送を行っている場合，サーバ自体の処理速度が高速だとしてもそのメリットをエンド・ユーザは体感できません。

## 関連する Windows カウンタ

ここでは，SQL Server がインストールされているマシンのシステム・リソースを監視する際に重要な役割を果たすカウンタと，監視のヒントをいくつか紹介します。

- **CPU :** 物理プロセッサを新しく追加する作業は簡単ではありません。すべての CPU に対して均等に負荷をかけることが必要です。次のカウンタを監視してください。

  **注 :** カウンタの詳細については，48 ページ「Processor - 最も重要なカウンタ」を参照してください。

  - **% Processor Time :** 個々のプロセッサ時間を測定します。これによって，CPU に負荷が分散されていることを確認します。
  - **Processor Queue Length :** このカウンタの値が常に推奨値を超えていても，CPU 使用率がそれほど高くない（通常のレベル）場合は，SQL Server の **maximum worker threads** の値を小さくすることをお勧めします。これにより，スレッド・プールが強制的に有効になり，要求の処理に使用されます。

- **Context Switches/sec**：この値を小さくするには，次の 2 つの方法があります。
  - **関係マスク**：負荷が大きい環境では，特定のプロセッサで実行するスレッドを指定しておくことで，プロセッサ・キャッシュの再ロード回数が低減し，パフォーマンスが向上します。また，SQL Server がアクセスできるプロセッサを制限することによって，オペレーティング・システム要求の処理効率が向上することがあります。
  - **簡易プーリング**：この SQL Server オプションを使用すると，データベース・モデルがスレッド・ベース（標準設定）ではなくファイバ・ベースに変更されます。ファイバのスケジューリングはデータベース・サーバではなくオペレーティング・システムによって実行されるので，CPU の負荷は小さくなります。

- **メモリ**：SQL Server はメモリを動的に管理し，オペレーティング・システムに対してメモリの要求と解放を行います。適切な動的オプションが選択されていることと，データベースが使用できる最大メモリ容量として，物理的な最大容量に近いレベルが割り当てられていることを確認してください。

  次のカウンタを監視してください。

  **注**：カウンタの詳細については，55ページ「Memory - 最も重要なカウンタ」を参照してください。

  - **Available Bytes**
  - **Pages/sec**：SQL Server の割り当て範囲を超えるディスク I/O やメモリ・アクセスの回数を示します。このカウンタの値は，バックアップやリストア時のスパイクを除いて，0 に近い値が理想的です。
  - **Page Faults/sec**

➤ **ディスク：**すべてのアプリケーション層の中でも I/O に最も負荷がかかる操作が実行されるのはデータベースであるため，ディスク操作の監視は重要です。

次のカウンタを監視してください。

**注：**カウンタの詳細については，66ページ「I/O - 最も重要なカウンタ」を参照してください。

➤ **% Disk Time :** 読み取り / 書き込みに費やされた時間の割合（パーセンテージ）: 単一のディスク・ボリュームについては物理ディスク・カウンタ，複数のディスクにまたがるボリュームについては論理ディスク・カウンタを監視します。この値が 55% を超える場合，I/O ボトルネックが発生していることを示します。この場合，**%Disk Read Time** カウンタと **%Disk Write Time** カウンタをチェックすることをお勧めします。上記の 2 つのカウンタの値の合計は，このカウンタの値と等しくなります。推奨されるチューニング作業には，ディスクの増設，高速ディスクの追加，ディスク・コントローラのキャッシュの増設，ディスクの最適化，RAID デバイスの再構成があります。

➤ **Avg. Disk Queue :** 特定のディスクのキューの実際の長さを示します。ただしこのカウンタは，ストレージ・エリア・ネットワーク（SAN）の稼働期間中の任意の時点での値を示すことがあります。単一のディスク・ボリュームについては物理ディスク・カウンタ，複数のディスクにまたがるボリュームについては論理ディスク・カウンタを監視します。**_Total** カウンタは追加しないでください。このカウンタは全体的な結果を示しているので問題点を的確に特定できなくなり，ディスク・パフォーマンスの前提条件を誤って認識してしまう可能性があります。

**ヒント：**SQL Server では，データ・ファイルとログ・ファイルで発生する I/O パターンが異なるので，それぞれを別のディスクに分離する構成は非常に優れた方法です。また，システムとユーザ・データベースのディスクを分離することもお勧めします。

- **ネットワーク**：一部のアプリケーションでは，大量のデータがネットワーク経由で送信されると，大量のインタラクションが発生することがあります。次のカウンタを監視します。

  **注**：詳細については，73ページ「Network - 最も重要なカウンタ」を参照してください。

  - **Bytes Total/sec**：具体的なデータを示す Bytes Received/sec カウンタと Bytes Sent/sec カウンタと併せて使用されるこのカウンタは，ネットワーク・カードの実際のスループットを示します。チューニングの方法としては，高速なネットワーク・カードの追加やカードの全二重オプションの指定などがあります。サーバとクライアントの両方で主に使用される TCP/IP 以外の不要なプロトコルを削除するように，データベースを再設定します。

## 最も重要な SQL Server カウンタ

SQL Server のパフォーマンス・アーキテクチャは，Microsoft が Windows オペレーティング・システムと .NET Framework で実装しているアプローチに基づいています。このアプローチは，オブジェクト，インスタンス，カウンタに基づくものです（Windows アーキテクチャについてはpage 46 を参照してください）。オブジェクトとは任意の SQL Server リソースであり，SQL Server ロックや Windows XP プロセスなどがあります。オブジェクトには 1 つまたは複数のカウンタがあり，監視対象となるオブジェクトのさまざまな側面を示しています。あるタイプのリソースが複数存在する場合は，オブジェクトに複数のインスタンスが含まれます。標準設定のインスタンスのカウンタは，**SQLServer:＜オブジェクト名＞** という形式で表されます。名前付きインスタンスのカウンタは，**MSSQL$＜インスタンス名＞:＜カウンタ名＞**または**SQLAgent$＜インスタンス名＞:＜カウンタ名＞**という形式で表されます。

SQL のパフォーマンス・オブジェクトは，次のように多数提供されています。

- SQLServer エンジン自体のオブジェクトが 20 個
- Service Broker のオブジェクトが 3 個
- SQL エージェントのオブジェクトが 4 個
- SQL レプリケーションのオブジェクトが 5 個

次の表では，データベース・エンジンのカウンタを示します。

| | カウンタ | 説明 |
|---|---|---|
| CPU | SQL Compilations/sec | 1 秒あたりのコンパイルの回数。 |
| | SQL Re-Compilations/sec | 1 秒あたりの再コンパイル回数。 |
| | Batch Requests/Sec | 1 秒あたりに受信した Transact-SQL コマンドのバッチの数。 |
| メモリ | Total Pages | バッファ・プール内のページ数。 |
| | Target Pages | バッファ・プール内の適切なページ数。 |
| | Total Server Memory (KB) | KB SQL Server が現在使用しているメモリ容量。 |
| | Target Server Memory (KB) | KB SQL Server が効率稼働するために必要なメモリ容量。 |
| | Buffer cache hit ratio | メモリ内で見つかったページの割合（パーセンテージ）。 |
| | Page Life Expectancy | ページが参照されない状態でバッファ・プールにとどまる秒数。 |
| | Stolen Pages | プロシージャ・キャッシュなど，さまざまな用途に使用されたページの数。 |
| | Cache hit ratio | キャッシュ・ヒットとルックアップの比率。 |
| | Memory Grants Pending | ワークスペース・メモリ許可を待機しているプロセスの総数。 |
| | Checkpoint pages/sec | チェックポイント，またはダーティ・ページをすべてフラッシュする必要がある操作によって，ディスクにフラッシュされたページの 1 秒あたりの数。 |
| | Lazy writes/sec | バッファ・マネージャのレイジー・ライタによって書き込まれたバッファの 1 秒あたりの数。 |

| | カウンタ | 説明 |
|---|---|---|
| ディスク | **Full Scans/sec** | 制限のないフルスキャンの 1 秒あたりの数。 |
| | **Page Splits/sec** | インデックス・ページがオーバーフローしたために発生したページ・スプリットの 1 秒あたりの数。 |
| | **Temp Tables Creation Rate** | 1 秒あたりに作成された一時テーブル/テーブル変数の数。 |
| ロック | **Average Wait Time (ms)** | 待機状態になったロック要求 1 つあたりの待機時間の平均値（ミリ秒）。 |

## CPU 関連のカウンタ

**sqlserver.exe** がほとんどの CPU 処理能力を使用している場合，SQL Server 内部で問題が発生している可能性があります。問題の切り分けでは，Windows カウンタ（232ページ「関連する Windows カウンタ」を参照してください）に加えて，次に示すカウンタも役に立ちます。

### SQL Compilations/sec

| | |
|---|---|
| **正式名** | SQLServer:SQL Statistics¥SQL Compilations/sec |
| **カウンタのタイプ** | サンプル間隔での差異（1 秒あたりの速度） |
| **説明** | SQL Server のコンパイルの 1 秒あたりの発生回数です。 |
| **使用方法** | CPU 使用率が過剰に高くなる一般的な原因に，スキーマの問題やメモリ不足などによって発生するクエリ実行計画のコンパイルや再コンパイルがあります。コンパイルが発生すると，メモリ不足などによって計画がキャッシュから削除されない限り，計画はメモリ内にとどまります。 |
| **パフォーマンス** | 安定稼働状態では，90% 以上の計画が再利用されます。 |
| **しきい値** | 10% を超える場合は注意が必要です。 |
| **関連するカウンタ** | ➤ SQL Re-Compilations/sec |

### SQL Re-Compilations/sec

| | |
|---|---|
| **正式名** | SQLServer:SQL Statistics¥SQL Re-Compilations/sec |
| **カウンタのタイプ** | サンプル間隔での差異（1 秒あたりの速度） |
| **説明** | SQL Server の再コンパイルの 1 秒あたりの発生回数です。 |
| **使用方法** | アドホック T-SQL のみを使用する場合やクエリが適切にパラメータ化されていない場合, SQL Server は計画を再利用できなくなり, すべてのクエリについてコンパイルが必要になることがあります。 |
| **パフォーマンス** | 再コンパイルは，デッドロックや，いずれのロック・タイプとも互換性のないコンパイル・ロックを引き起こすことがあります。 |
| **しきい値** | SQL Compilations/sec の 10% を超える場合は注意が必要です。 |
| **関連するカウンタ** | ➤ SQL Compilations/sec |

### Batch Requests/sec

| | |
|---|---|
| **正式名** | SQLServer:SQL Statistics¥Batch Requests/sec |
| **カウンタのタイプ** | サンプル間隔での差異（1 秒あたりの速度） |
| **説明** | 1 秒あたりに受信した Transact-SQL コマンドのバッチの数です。 |
| **使用方法** | SQL Server の CPU がどの程度ビジー状態かを示します。 |
| **パフォーマンス** | このカウンタの値がしきい値を超えたとしても，CPU ボトルネックが実際に発生するとは限りませんが，発生が予想されます。もちろん，このカウンタはハードウェア機能によって変動します。 |
| **しきい値** | 1000 を超える場合は注意が必要です。 |
| **関連するカウンタ** | なし |

**注：**パフォーマンスの監視では，SQLServer:Databases¥ Transaction/Sec: _Total カウンタが役立つことがあります。Batch Requests/sec カウンタがすべてのアクティビティを対象に測定を行うのに対して，このカウンタはトランザクション内で発生したアクティビティのみを対象にします。

## メモリ関連のカウンタ

SQL Server のパフォーマンスと安定性は，使用可能なメモリ容量が十分に確保されているかどうかで決まります。メモリ不足が発生すると，Windows はページング・ファイルの仮想アドレス領域を使用しますが，これはパフォーマンスに直ちに大きな影響を与えます。

メモリ関連の問題を監視するには，Windows カウンタ（232ページ「関連する Windows カウンタ」を参照してください）に加えて，次に示すカウンタも役立ちます。

### Total Pages

| | |
|---|---|
| **正式名** | SQLServer:Buffer Manager¥Total Pages |
| **カウンタのタイプ** | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| **説明** | バッファ・プール内のページ数です（データベース，空きページ，失われたページも含まれます）。 |
| **使用方法** | SQL Server が Windows オペレーティング・システムから取得したページ数の合計を示します。 |
| **パフォーマンス** | マシン上で実行中の他のプロセスが, SQL Server から物理メモリを取得していることを示します。 |
| **しきい値** | 240ページのヒントを参照してください。 |
| **関連するカウンタ** | ➤ Target Pages |

### Target Pages

| | |
|---|---|
| **正式名** | SQLServer:Buffer Manager¥Target Pages |
| **カウンタのタイプ** | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| **説明** | バッファ・プール内のページ数の適正値です。 |
| **使用方法** | 要求を処理するために, SQL Server が必要とするページ数の合計を示します。 |
| **パフォーマンス** | なし |
| **しきい値** | 下のヒントを参照してください。 |
| **関連するカウンタ** | ➤ Total Pages |

---

**ヒント：**Target Pages と Total Pages の値が等しい場合，SQL Server には十分なメモリ容量が確保されていることを示します。Target Pages の方が Total Pages よりも大きい場合，他の Windows プロセスが原因で SQL Server が必要量のメモリを取得できない状態を示します。

---

### Total Server Memory (KB)

| | |
|---|---|
| **正式名** | SQLServer:Memory Manager¥Total Server Memory (KB) |
| **カウンタのタイプ** | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| **説明** | SQL Server が現在使用しているメモリ容量（KB 単位）です。 |
| **使用方法** | SQL Server が Windows オペレーティング・システムから取得した物理メモリ容量の合計を示します。 |
| **パフォーマンス** | 適正値は, マシンに搭載されているメモリ容量の合計よりも小さい値です。 |
| **しきい値** | 241ページのヒントを参照してください。 |
| **関連するカウンタ** | ➤ Target Server Memory (KB) |

### Target Server Memory (KB)

| | |
|---|---|
| **正式名** | SQLServer:Memory Manager¥Target Server Memory (KB) |
| **カウンタのタイプ** | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| **説明** | SQL Server が効率的に稼働するために必要となるメモリ容量です。 |
| **使用方法** | 要求を処理するために, SQL Server が必要とするメモリ容量の合計を示します。 |
| **パフォーマンス** | |
| **しきい値** | 下のヒントを参照してください。 |
| **関連するカウンタ** | ➤ Total Server Memory (KB) |

---

**ヒント：** Total Server Memory (KB) の値が Target Server Memory (KB) よりも小さい場合, SQL Server はメモリ不足で効率的に稼働していないことを示します。物理メモリの追加を検討してください。

---

### Buffer cache hit ratio

| | |
|---|---|
| **正式名** | SQLServer:Buffer Manager¥Buffer cache hit ratio |
| **カウンタのタイプ** | サンプル間隔（ビジー状態の %） |
| **説明** | メモリ内で見つかったページの割合（パーセンテージ）です。 |
| **使用方法** | この値は，直近の数千ページ・アクセスについて，キャッシュ・ヒット数の合計をキャッシュ・ルックアップの合計で割って計算されます。 |
| **パフォーマンス** | データ・ページがバッファ内に存在しないと，SQL Server はディスクからバッファにページを読み込まなければなりません。この処理は，ディスク遅延とシーク時間の影響で時間がかかります。<br>したがって，バッファ・プールをこのカウンタの値の 98% 以上になるように設定してもパフォーマンスが改善しない場合には，物理メモリの追加を検討してください。 |
| **しきい値** | この値は，大きいほど理想的です。90% 前後が適正値です。 |
| **関連するカウンタ** | なし |

### Page Life Expectancy

| | |
|---|---|
| **正式名** | SQLServer:Buffer Manager¥Page Life Expectancy |
| **カウンタのタイプ** | 平均値 |
| **説明** | ページが参照されない状態でバッファ・プール内にとどまる秒数です。 |
| **使用方法** | メモリ使用の点から考えると，ページがとどまる期間が長くなるほど，サーバの稼働状態は良好だといえます。 |
| **パフォーマンス** | サーバのメモリ不足を明確に示す指標です。 |
| **しきい値** | 300 秒未満の場合は対処が必要です。 |
| **関連するカウンタ** | なし |

### Stolen Pages

| | |
|---|---|
| **正式名** | SQLServer:Buffer Manager¥Stolen Pages |
| **カウンタのタイプ** | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| **説明** | プロシージャ・キャッシュなど，サーバのさまざまな用途に使用されたページの数です。 |
| **使用方法** | Stolen Page とは，SQL Server マシン上の別のプロセスが取得したことによって失われたメモリ内のページを指します。 |
| **パフォーマンス** | 値が大きい場合，明らかにサーバでメモリ不足が発生していることを示します。 |
| **しきい値** | なし |
| **関連するカウンタ** | ➤ Total Pages |

### Cache Hit Ratio

| | |
|---|---|
| **正式名** | SQLServer:Plan Cache¥Cache Hit Ratio |
| **カウンタのタイプ** | サンプル間隔（ビジー状態の %） |
| **説明** | キャッシュ・ヒットとルックアップの比率です。 |
| **使用方法** | レコードがキャッシュ内で見つかった時間の割合（パーセンテージ）です。<br>**注：**SQL Server 2000 では，このカウンタは Cache Manager オブジェクトに含まれます。 |
| **パフォーマンス** | SQL Server のキャッシング機能を示す優れた指標です。 |
| **しきい値** | 適正値は 99% 前後です。90% の場合は注意が必要です。 |
| **関連するカウンタ** | なし |

### Memory Grants Pending

| | |
|---|---|
| **正式名** | SQLServer:Memory Manager¥Memory Grants Pending |
| **カウンタのタイプ** | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| **説明** | ワークスペース・メモリ許可を取得しようと待機しているプロセスの総数です。 |
| **使用方法** | 実質的に，このカウンタはメモリ許可を取得しようと待機しているプロセスのキューを示します。 |
| **パフォーマンス** | メモリを取得しようと待機しているプロセスが存在する場合，パフォーマンスが低下していると考えれます。理想的なサーバ稼働環境では，処理を待つメモリ許可は存在しません。 |
| **しきい値** | 0 以外の場合は対処が必要です。 |
| **関連するカウンタ** | なし |

### Checkpoint Pages/sec

| | |
|---|---|
| **正式名** | SQLServer:Buffer Manager¥Checkpoint pages/sec |
| **カウンタのタイプ** | サンプル間隔での差異（1 秒あたりの速度） |
| **説明** | チェックポイント，またはダーティ・ページをすべてフラッシュする必要がある操作によって，ディスクにフラッシュされたページの 1 秒あたりの数を示します。 |
| **使用方法** | チェックポイント操作は SQL Server によって実行されます。この操作では，ダーティ・ページをすべてディスクに書き込む必要があります。 |
| **パフォーマンス** | ディスク I/O の点で考えると，チェックポイントは負荷の高いプロセスです。サーバでメモリ不足が発生すると，SQL Server はバッファ・プール内で領域を確保しようとするので，通常よりもチェックポイント・プロセスの発生頻度が高くなります。<br>このカウンタは，サーバのメモリ不足を明確に示す指標です。 |
| **しきい値** | 値が大きい状態が続く場合には，対処が必要です。 |
| **関連するカウンタ** | なし |

### Lazy Writes/sec

| | |
|---|---|
| **正式名** | SQLServer:Buffer Manager¥Lazy Writes/sec |
| **カウンタのタイプ** | サンプル間隔での差異（1 秒あたりの速度） |
| **説明** | バッファ・マネージャのレイジー・ライタによって書き込まれたバッファの 1 秒あたりの数です。 |
| **使用方法** | レイジー・ライタとは，時間が経過したダーティ・バッファのバッチをフラッシュし，ユーザ・プロセスがバッファを使用できる状態にするシステム・プロセスです。このカウンタは，SQL Server がバッファ・プール（メモリ内）からディスクにダーティ・ページを移動する 1 秒あたりの回数を記録します。 |
| **パフォーマンス** | ディスク I/O はリソースを消費する処理なので，バッファ・プール内で十分な容量を確保することにより，レイジー・ライトが限りなく 0 に近付くようにしてください。<br>このカウンタは，サーバのメモリ不足を明確に示す指標です。 |
| **しきい値** | 0 以外の場合は対処が必要です。20 を超える場合は，バッファ・プールのサイズを大きくする必要があります。 |
| **関連するカウンタ** | なし |

## ディスク関連のカウンタ

ディスク・データの読み取りや書き込みは最も時間のかかる処理であり，SQL Server のリソースを大量に消費します。メモリとディスク間でのデータ転送は，若干の遅延が発生するだけで，サーバ・パフォーマンスに影響を与えてしまいます。SQL Server では，ユーザ処理に遅延が発生しないようにする機能として，データを事前にロードしておくバッファ・キャッシュと，最適なデータ取得方法を詳細に記述したプランが格納されたプラン・キャッシュの 2 つが提供されています。ディスク・パフォーマンスに問題がある場合には，ストレージ・サブシステムの設計と実装をレビューすることをお勧めします。

ディスク関連の問題を監視するには，Windows カウンタ（232ページ「関連する Windows カウンタ」を参照してください）に加えて，次に示すカウンタも役立ちます。

### Full Scans/sec

| | |
|---|---|
| **正式名** | SQLServer:Access Methods¥Full Scans/sec |
| **カウンタのタイプ** | サンプル間隔での差異（1 秒あたりの速度） |
| **説明** | 制限のないフルスキャンの 1 秒あたりの数です。 |
| **使用方法** | テーブル・スキャンは必ず発生する処理であり，インデックス・シークよりも高速実行できることもありますが，一般的にはテーブル・スキャンの回数をできるだけ抑える必要があります。このカウンタは，単一のデータベースではなくサーバ全体を示す指標です。 |
| **パフォーマンス** | 定期的にテーブル・スキャンが発生している場合，SQL Server の内部ジョブが原因として考えられます。ただし，ランダムにスパイクが発生する場合には，インデックスが不十分または欠落している状態を示します。 |
| **しきい値** | 1 の場合は注意が必要であり，2 を超えるとエラーを示します。 |
| **関連するカウンタ** | なし |

### Page Splits/sec

| | |
|---|---|
| **正式名** | SQLServer:Access Methods¥Page Splits/sec |
| **カウンタのタイプ** | サンプル間隔での差異（1 秒あたりの速度） |
| **説明** | インデックス・ページがオーバーフローしたために発生した 1 秒あたりのページ・スプリットの数です。 |
| **使用方法** | ページ・スプリットとは，8 KB データ・ページ内に十分な領域がないことが原因で挿入操作または更新操作を完了できないと発生し，I/O に大きな負荷がかかります。この場合，新しいページが追加され，2 つのページでオリジナル・データを共有することによって，挿入操作や更新操作を実行します。 |
| **パフォーマンス** | 正常稼働状態でもページ・スプリットは時々発生しますが，過剰だとディスク I/O が大量に発生し，パフォーマンス低下につながります。この問題は，インデックスを適切な方法で保守し，FILL FACTOR に適切な値を指定することによって回避できます。 |
| **しきい値** | 100 を超える場合は注意が必要です。 |
| **関連するカウンタ** | なし |

---

**ヒント**：SQL Server では標準で自動拡張が設定されるため，必要に応じてデータ・ファイルとログ・ファイルが拡張されます。これは便利な機能ですが，エンタープライズ・システム上の設定を手作業で調整することをお勧めします。

---

### Temp Tables Creation Rate

| | |
|---|---|
| **正式名** | SQLServer:General Statistics¥Temp Tables Creation Rate |
| **カウンタのタイプ** | サンプル間隔での差異（1 秒あたりの速度） |
| **説明** | 1 秒あたりに作成された一時テーブル/テーブル変数の数です。 |
| **使用方法** | SQL Server は，結合，ソート，計算などの操作の途中の格納場所として，またバージョン・ストアに **tempdb** を使用します。**tempdb** が多用される環境では, tempdb の応答速度がユーザ応答時間に直接的な影響を与えることがあります。 |
| **パフォーマンス** | **tempdb** は，共有グローバル・リソースです。したがって，temdb を多用するデータベースやアプリケーションが 1 つでもあると，同じインスタンス内の他のデータベースのパフォーマンスが低下し，パフォーマンスを調整できなくなってしまいます。 |
| **しきい値** | なし |
| **関連するカウンタ** | なし |

**ヒント：**自動拡張が行われないように，**tempdb** を十分大きな値に設定します。

## ロック関連のカウンタ

ロックは競合の処理で必要になる機能です。SQL Server は，ロックを自動的に処理します。ロックは個々のデータベースや SQL Server 全体の内部的な動作であり，オペレーティング・システム・リソースに関連した動作ではありませんが，応答時間には大きな影響を与えます。トランザクションの実行時間が長くなるとエンド・ユーザからよく苦情が報告されますが，その原因の 1 つがロックです。

ほとんどの場合，SQL Server はロックを自動的に解消します。ただし，**ブロッキング・ロック**と**デッドロック** の 2 つには注意が必要であり，頻繁に発生しないか監視する必要があります。

- **ブロッキング・ロック**：プロセスがリソースを使用しようとしたときに，別のプロセスによってすでにロックされているためにブロックされる状態を指します。
- **デッドロック**：2 つのプロセスがロックを保持しているために，相互に処理を継続できない状態を指します。お互いにいつまでも待つことになります。

### Average Wait Time (ms)

| | |
|---|---|
| **正式名** | SQL Server:Locks¥Average Wait Time (ms) |
| **カウンタのタイプ** | 平均値 |
| **説明** | 待機状態になったロック要求 1 つあたりの平均待機時間（ミリ秒）。 |
| **使用方法** | オブジェクト・ロックが原因で応答時間が遅くなっているかどうかを示します。このカウンタでは，データベース，エクステント，キー，ページ，RID，テーブルなど，さまざまなロックの平均待機時間を測定できます。 |
| **パフォーマンス** | 一定期間，各ロック・タイプについて監視します。タイプごとの平均値を取得し，これを基準値として使用します。 |
| **しきい値** | なし |
| **関連するカウンタ** | なし |

**ヒント：**トランザクションの遅延を引き起こしているロックのタイプが 1 つまたは複数見つかったら，さらに詳細な調査を行い，どのようなトランザクションがロックの原因になっているかを確認します。HP Diagnostics ソフトウェアには，問題になっているステートメントを特定する機能があります。

# 第VII部

## 仮想化テクノロジ

# 第15章

# Microsoft 仮想化環境の監視

本章では，Microsoft 仮想化サーバである Hyper-V の監視に関するベスト・プラクティスを紹介します。

**本章の内容**



## 概要

Microsoft は，データセンターからデスクトップまで幅広い環境を対象に包括的な仮想化製品を提供しており，物理的な資産と仮想化された資産の両方を 1 つのプラットフォームから簡単に管理することができます。

このような仮想化に向けた Microsoft のビジョンと戦略の中心的役割を果たすのが Microsoft Hyper-V です。Hyper-V はハードウェア機能を使った新しい仮想化技術であり，Microsoft Windows Server 2008 x64 エディションに搭載されています。

Hyper-V の仮想化プラットフォームでは，ハイパーバイザと呼ばれる，ハードウェアのすぐ上で稼働する薄いソフトウェア層が採用されています。ハイパーバイザによって，複数のオペレーティング・システムをパーティション内で同時実行することが可能になり，メモリやプロセッサなど重要なシステム・リソースにアクセス・ポリシーを適用することでパーティションを完全に隔離します。

Hyper-V が実現するハイパーバイザ・ベースの仮想化プラットフォームでは，コスト削減，ハードウェア効率活用，インフラストラクチャの最適化，サーバ可用性の向上を通して高い柔軟性が実現されます。

Hyper-V 環境の仮想マシンは，ハードウェア・レベルのセキュリティ機能など優れたセキュリティ機能を利用できます。

Hyper-V は堅牢性とスケーラビリティの両方を実現するので，これまでは物理ハードウェアで実行する必要があった作業負荷を仮想作業負荷に変えても，業務で必要とされるレベルのパフォーマンスを発揮することが可能になります。

Hyper-V には，次のような機能があります。

- **ボリューム・シャドウ・コピー・サービスによるライブ・バックアップ**：ボリューム・シャドウ・コピー・サービス（VSS，Windows Server 2003 以降のゲストオペレーティング・システムに対応）を実行する仮想マシンは，ライブ状態でバックアップを実行でき，ダウンタイムを最小限にまで短縮できます。
- **フェイルオーバ・クラスタリングによる高可用性**：Hyper-V は，Windows フェイルオーバ・クラスタリングのサポートを通じて，予定したダウンタイムと予期しないダウンタイムの両方に対応する高可用性戦略を実装します。
- **クイック・マイグレーション**：Hyper-V がサポートするクイック・マイグレーションでは，データ損失がなく，サービスの中断を最小限に抑える方法で仮想マシンをクラスタ・ノード間で移動します。具体的な手順としては，仮想マシンを保存状態にし，アクティブなメモリとプロセッサの状態をディスクに保存してから，ストレージ・リソースの所有権をクラスタ内のノード間で移動します。次に，新しいノードに仮想マシンのアクティブなメモリとプロセッサの状態が再ロードされ，処理が再開されます。
- **統合サービス**：Hyper-V 統合サービス（IS）は，次に示すように，親と子のパーティション間でセキュアなインタフェースを必要とする 5 つのコンポーネントをサポートします。
  - 時刻の同期
  - ハートビート
  - シャットダウン
  - キー値ペア交換
  - ボリューム・シャドウ・コピー・サービス（VSS）

- **仮想マシンのインポートとエクスポート**：Hyper-V でのインポートおよびエクスポートとは Hyper-V サーバ間で仮想マシンを移動およびコピーする操作を指します。
- **仮想ハードディスク管理**：Hyper-V では，仮想ハードディスク（VHD）の管理オプションが複数提供されていて（最適化，変換，拡張，結合，再接続），Hyper-V マネージャ・コンソールからアクセスできます。
- **仮想マシンのスナップショット**：Hyper-V では，任意の時点での仮想マシンの構成と状態を示すスナップショットを取得できます。取得したスナップショットは，わずか数秒で再ロードできます。
- **仮想マシン接続**：仮想マシン接続（VMC）は，Hyper-V のリモート管理ツールです。VMC は Windows リモート・デスクトップ・プロトコルを使って仮想マシン上で稼働するゲスト・オペレーティング・システムにリモート・アクセスします。

## アーキテクチャ

Microsoft Hyper-V は，Windows Server 2008 に付属するバージョンと，Microsoft Hyper-V Server 2008 というスタンドアロン・バージョンのいずれかで提供されます。この両方のバージョンのアーキテクチャには，非常に高い類似性があります。

Hyper-V は，Windows ハイパーバイザ（リング －1），カーネル・モード・コンポーネント（リング 0），ユーザ・モード・コンポーネント（リング 3）と呼ばれるハイパーバイザ・コンポーネントで構成されます。

この 3 つのプロセッサ・リングは命令の特権レベルを定義するものであり，リング 0 が最上位レベル，リング 3 が最下位レベルとなります。

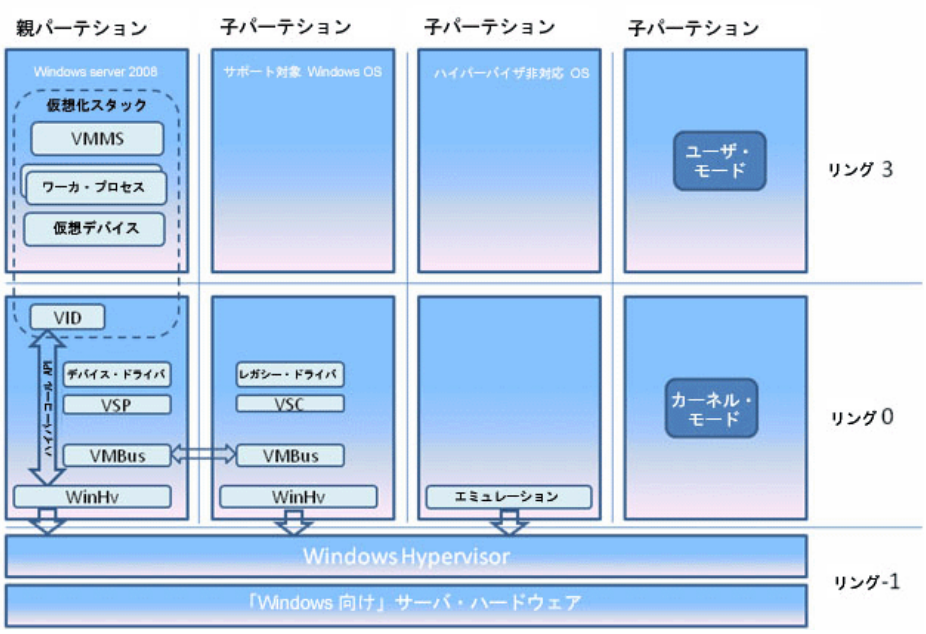
- リング 0：オペレーティング・システムのカーネルを実行するモードです。
- リング 3：ユーザ・アプリケーションを実行するモードです。
- リング －1：ハードウェア仮想化エクステンションを実行するモードです。

  リング －1 では，Windows ハイパーバイザは専用のコンテキスト内で，Windows カーネルよりも高い特権レベルで動作します。これに対して，ゲスト・オペレーティング・システム・カーネルはリング 0，ユーザ・アプリケーションはプロセッサのリング 3 で動作します。

Windows ハイパーバイザ層の上では，1 つの親パーティションと，1 つまたは複数の子パーティションが動作します。

- 親パーティションは，仮想化スタックが動作する制御パーティションです。親パーティションは，ハードウェア・デバイスの所有者でもあり，子パーティションのリソースを管理します。
- 子パーティションは，親パーティションによって作成されたパーティションです。ゲスト・オペレーティング・システムとそのアプリケーションは，子パーティション内で動作します。

パーティションは，ハイパーコールを使ってハイパーバイザ層と通信します。ハイパーコールとはオペレーティング・システムをパーティション化する API であり，ハイパーバイザの最適化機能を活用します。




次に，Hyper-V アーキテクチャのパーティションと層に関連する主要コンポーネントであるハイパーバイザ（リング － 1），ユーザ・モード（リング 3），カーネル・モード（リング 0）について説明します。

## ハイパーバイザ

Windows ハイパーバイザは，物理ハードウェアと 1 つまたは複数のオペレーティング・システムの間にあるソフトウェア・インタフェース層です。Windows ハイパーバイザは，コア・ハードウェアへのアクセスを制御し，パーティションと呼ばれる隔離された実行環境を定義します。

Windows ハイパーバイザの主な機能には，パーティション間の完全な隔離，ハードウェア・アクセスを制限するポリシーの適用，パーティションの監視があります。

## 親パーティション

親パーティションは，ハイパーバイザの起動時にシステム上で最初に作成されるパーティションです。親パーティションは，Windows Server 2008 オペレーティング・システム用に作成され，次の機能を持ちます。

- 親パーティションは，子パーティションの作成と管理を行います。子パーティションには，リモート管理インタフェースを提供する WMI プロバイダも含まれます。
- 親パーティションはハードウェア・デバイスの管理と割り当てを行います。ただし，プロセッサのスケジューリングと物理メモリの割り当てはハイパーバイザが行います。
- 親パーティションのハードウェア・リソースは，子パーティションが共有，または子パーティションに割り当てられます。
- 親パーティションは，電源管理，プラグ・アンド・プレイ操作，ハードウェア障害イベントのログ記録などの処理を行います。

親パーティション内にある仮想化コンポーネントは，まとめて仮想化スタックと呼ばれます。仮想化スタックは親パーティション内で動作し，基盤となるホスト・コンピュータのハードウェアに直接アクセスします。仮想化スタックには，さらに次のようなコンポーネントが存在します。

次に，仮想化スタックのコンポーネントを示します。

- 仮想マシン管理サービス - VMMS
- 仮想マシン・ワーカ・プロセス
- 仮想デバイス

➤ 仮想化インフラストラクチャ・ドライバ（VID）

➤ Windows ハイパーバイザ・インタフェース・ライブラリ

親パーティションには，次のコンポーネントも含まれます。

➤ 仮想化サービス・プロバイダ（VSP）

➤ 仮想マシン・バス（VMBus）

次の項では，親パーティションの主要コンポーネントについて詳しく説明します。

### 仮想マシン管理サービス - VMMS

仮想マシン管理サービス（VMMS）は，仮想マシンの管理を連携して行うコンポーネント群です。

VMMS は，ユーザ・モードとカーネル・モードの両方でシステム・サービス（VMMS.exe）として実装され，子パーティションでの仮想マシンの状態を管理します。

この管理作業には，停止またはオフライン状態の仮想マシンの管理，スナップショットの作成，デバイスの追加と削除の管理が含まれます。子パーティション内で仮想マシンが起動すると，VMMS は新しい仮想マシン・ワーカ・プロセスを生成します。このプロセスは，仮想マシンの管理タスクの実行に使用されます。

VMMS は，仮想マシンで実行する操作を状態ごとに管理します。VMMS が管理する仮想マシンの状態は，起動，アクティブ，非アクティブ，スナップショットの取得/適用/削除，ディスクの結合です。

ただし，一時停止，保存，電源オフなどのオンライン仮想マシンの操作は管理の対象ではありません。このような操作の管理は，管理対象の仮想マシン用に VMMS が生成した仮想マシン・ワーカ・プロセスが行います。

## 仮想マシン・ワーカ・プロセス

仮想マシン・ワーカ・プロセス（vmwp.exe）は，親パーティション内の Windows Server 2008 インスタンスから子パーティション内のゲスト・オペレーティング・システムへ，仮想マシン管理サービスを提供するユーザ・モード・プロセスです。

VMMS は，稼働中の仮想マシンごとに独立した VM ワーカ・プロセスを生成することによって，仮想マシンを分離します。これにより，VM ワーカ・プロセスに障害が発生しても，影響が発生する範囲をその VM ワーカ・プルセスに関連する仮想マシンにとどめることができます。

VM ワーカ・プロセスは，関連する仮想マシンについて次の操作を管理します。

- 仮想マシンの作成，設定，稼働
- 仮想マシンの一時停止と再開
- 仮想マシンの保存と復元
- 仮想マシンのスナップショットの取得

さらに，VM ワーカ・プロセスには仮想マザーボード（VMB）が含まれます。VMB は，ゲスト・メモリ，IRQ 生成，メモリ・マップやポート・マップの対象 I/O を，仮想マシンに別デバイスとして提供します。また，仮想デバイスの管理も行います。

## 仮想デバイス

仮想デバイス（VDevs）は，子パーティションにデバイスの設定や管理機能を提供するソフトウェア・モジュールです。

VDevs には，次の 2 つのタイプがあります。

- コア VDev：

この仮想デバイスは，既存のハードウェア・デバイスをモデル化するものであり，すべての仮想マシンが使用できます。BIOS やデバイス・ドライバなど既存のソフトウェアの互換性を重視し，変更なしでそのまま正常稼働したい場合に使用されます。コア VDevs には，次のデバイスが含まれます。

  - エミュレート・デバイス：この仮想デバイスは，ハードウェア・デバイスのエミュレーションを行います。BIOS，DMA，PCI バス，キーボード/マウス・コントローラなどがあります。

- 合成デバイス：この仮想デバイスは，ハードウェア・デバイスのモデル化を行いません。統合サービスをサポートするゲスト・オペレーティング・システムのみが使用できます。

- プラグイン VDev：

既存のハードウェア・デバイスのモデル化は行わず，ハードウェア管理を担当する親パーティション内で稼働する仮想サービス・プロバイダのインスタンス化，設定，管理を行います。

プラグイン VDev により，VMBus 経由で親パーティションと子パーティション間の直接通信が可能になります。

### 仮想化インフラストラクチャ・ドライバ（VID）

仮想化インフラストラクチャ・ドライバ（Vid.sys）は，仮想化スタックのカーネル・モード・コンポーネントであり，パーティション管理サービス，仮想プロセッサ管理サービス，メモリ管理サービスをすべての子パーティションに提供します。また，仮想化スタックのユーザ・モード・コンポーネントは，Vid.sys を使ってハイパーバイザと通信します。

### Windows ハイパーバイザ・インタフェース・ライブラリ

Windows ハイパーバイザ・インタフェース・ライブラリ（WinHv.sys）は，カーネル・モードのダイナミック・リンク・ライブラリであり，親パーティションで稼働する Windows Server 2008 インスタンス内と，子パーティションにある Hyper-V 対応ゲスト・オペレーティング・システム内でロードされます。

WinHv.sys はハイパーコールの詳細な実装を抽象化します。これにより，オペレーティング・システムのドライバは，Windows の標準的な呼び出し規約に基づいてハイパーバイザを呼び出すことができます。

### 仮想化サービス・プロバイダ（VSP）

仮想サービス・プロバイダ（VSP）は，親パーティション内でホストされます。子パーティションで稼働する仮想サービス・クライアント（VSC）に I/O 関連リソースを提供することによって，デバイス・サービスを子パーティションに発行します。デバイス機能に関するクライアント・サーバ通信では，VSP はサーバ・エンドポイント，VSC はクライアント・エンドポイントになります。VSP と VSC 間の通信はすべて VMBus 経由で実行されます。

### 仮想マシン・バス（VMBus）

仮想マシン・バス（VMBus）は，親パーティションと子パーティション間を接続する論理的なチャネル・ベースのパーティション間通信です。VMBus は高速で最適化された仮想パーティション間通信を可能にします。また，エミュレーションによるオーバーヘッドが小さいため，他の方法よりも高速です。

## 子パーティション

子パーティションは，親パーティションによって作成されたパーティションです。

子パーティションは，物理ハードウェアをソフトウェア・ベースで表現したものであり，仮想マシンと呼ばれます。ゲスト・オペレーティング・システムとそのアプリケーションは，子パーティション内で動作します。

子パーティションがサーバの実物理ハードウェアに直接アクセスすることはなく，仮想ハードウェアと仮想デバイスのみを認識します。

Hyper-V は，次の 3 つのタイプの子パーティションをサポートします。

- Hyper-V 対応 Windows OS をホストする子パーティション。
- Hyper-V 対応のサポート対象/非 Windows OS をホストする子パーティション。
- Hyper-V 非対応の Windows OS をホストする子パーティション。

### Hyper-V 対応 Windows OS をホストする子パーティション

Hyper-V 対応 Windows オペレーティング・システムが動作する子パーティションには，次のカーネル・モード仮想化コンポーネントが含まれます。

**仮想化サービス・クライアント：**
VSC は，子パーティション内にある合成デバイスであり，親パーティション内の VSP が VMBus を介して提供するハードウェア・リソースを使用します。

子パーティションに統合サービスがインストールされると，子パーティションは合成デバイスを使用できるようになるので，VSC は自動的に使用可能な状態になります。

**エンライトメント**

ゲスト OS コードに対する変更であり，これによってハイパーバイザ環境内でゲストとして実行されるゲスト OS コードの動作が効率化されます。

Hyper-V は，ストレージ，ネットワーク，グラフィックス，入力の各サブシステムでエンライトメントをサポートします。

## Hyper-V 対応のサポート対象/非 Windows OS をホストする子パーティション

Windows 以外の Hyper-V 対応オペレーティング・システム環境で動作する子パーティションは，サードパーティ VSC を使って VMBus 経由で親パーティション内の VSP と通信し，ハードウェアにアクセスします。子パーティションについては，統合サービスを子パーティションにインストールすることによって VSC が提供されます。

統合サービスは，仮想マシンの特徴である隔離環境が原因で発生するユーザビリティの問題を解消します。また，子パーティションが他のパーティションやハイパーバイザとの通信に必要なコンポーネントも提供します。

統合サービスは，次のような機能を子パーティションに提供します。

- **ハートビート**：親パーティションからの要求に子パーティションが応答していることを検証します。
- **キー値ペア交換**：子パーティションと親パーティション間で交換されるレジストリ・キーのペアです。
- **時刻の同期**：子パーティションの時刻を親パーティションと同期します。
- **シャットダウン**：親パーティションからのシャットダウン要求に子パーティションが応答します。

Hyper-V は次に示すように，x86 と x64 の Windows OS バージョン向けの統合サービスを提供します。

Win XP（SP3），Win Vista（SP1），Win Server 03（SP2），Win Server 08，Linux Enterprise Server 10。

### Hyper-V 非対応の Windows OS をホストする子パーティション

Hyper-V 非対応オペレーティング・システムで動作する子パーティションには，統合サービスをインストールできません。

したがって，ゲスト・オペレーティング・システムは合成デバイスではなくエミュレート・デバイスを使用しなければならなくなり，その結果パフォーマンスが低下します。

## 監視ツール

次の節では，Hyper-V サーバを使った仮想環境の監視用に Microsoft と HP が提供している監視ツールを紹介します。

### Microsoft の監視ソリューション

Microsoft は，監視ソリューションとして主に次の 2 つを提供しています。

- 信頼性とパフォーマンス・モニタ
- System Center Operations Manager 2007

このツールは，物理ホスト，Hyper-V の親パーティションと子パーティションで発生しそうな潜在的な問題について，調査と通知を行います。

物理ホストについては，温度，電源，稼働時間といった環境的な問題を中心に監視を行います。

Hyper-V の親パーティションについては，論理プロセッサ，システム・メモリ使用率，システム・ストレージのパフォーマンス，システム・ネットワークのパフォーマンス，Windows ハイパーバイザ，親パーティション・サービスを中心に監視を行います。

子パーティションについては，割り当てられた仮想ハードウェア（仮想プロセッサ，メモリ，ストレージ，ネットワーク・アダプタ），子パーティションで動作するサービスおよびアプリケーションを中心に監視を行います。

## 信頼性とパフォーマンス・モニタ

信頼性とパフォーマンス・モニタは，Windows Server 2008 のフル・インストールに標準で付属します。このツールは，ローカル・システムまたはリモート・システムの監視が可能な MMC ベースのアプリケーションです。信頼性とパフォーマンス・モニタは，2 つのツールを 1 つに統合したツールであり，信頼性モニタはシステムの安全性と信頼性に影響を及ぼすイベントに関する情報を提供し，パフォーマンス・モニタは，システム・コンポーネント，サービス，アプリケーションに関して詳細なリアルタイム・パフォーマンス情報を提供します。

信頼性モニタは，ソフトウェアのインストール，アプリケーション障害，ハードウェア障害，オペレーティング・システム障害などさまざまな障害イベントの履歴を追跡します。データは，システム安定性グラフとシステム安定性レポートで提示されます。

システム安定性グラフでは，過去 30 日間のイベントを対象に，1（安定性が最も低い）から 10（安定性が最も高い）のインデックス値が表示されます。安定性インデックスは，履歴期間内で検出された障害の数を元に加重計算されます。

システム安定性レポートでは，実際のイベントや障害，発生したアクティビティ，ステータス，発生したデータなど詳細情報が表示されます。

**パフォーマンス・モニタ**は，リアルタイムでのデータ取得とログへのデータ記録という 2 つのモードで動作します。

リアルタイムでのデータ取得では，選択したパフォーマンス・カウンタを使ってパフォーマンス情報をリアルタイムで表示します。パフォーマンス・カウンタは，オペレーティング・システム，アプリケーション，サービスごとに定義されます。

カウンタはグループ分けされており，システム・パフォーマンスの数値を 1 つ提示するカウンタと，複数のインスタンスが含まれるカウンタがあります。たとえば，LogicalDisk という名前のカウンタ・グループでは 23 のカウンタが定義されています。

データ・ログは，パフォーマンス・カウンタをリアルタイム表示するのではなく，履歴データを表示する機能です。リアルタイム収集モードでは，収集頻度に基づいて，古いデータ・セットが新しいデータで上書きされます。収集データの履歴を保持するには，データ・ログ・モードを使用する必要があります。

## System Center Operations Manager 2007

System Center Operations Manager 2007 は Microsoft のエンタープライズ・ハードウェア，オペレーティング・システム，サービス，アプリケーションの監視ソリューションです。Operations Manager 2007 は，エージェント・ベースのデータ収集機能を使ってリモート・システムの情報を収集し，後で分析できるように SQL データベースに格納します。データ収集設定は，管理パックという概念に基づいて行われます。管理パックには，アプリケーション，オペレーティング・システム，ハードウェアの規則，モニタ，タスクが付属します。

規則とは，Perfmon，EventLog，SNMP，ログ・ファイルなど各種ソースからデータを収集する方法を定義したものです。収集されたデータは Operations Manager データベースに格納され，これを元にレポートが作成されます。モニタとは，マシン監視対象となる稼働状態を定義するステート・マシンです。モニタには 2 つのステート（緑，赤）または 3 つのステート（緑，黄色，赤）があり，受信する監視情報に応じて変化します。モニタでは，収集を規定するデータのしきい値を定義しておくことで，しきい値に違反が発生した時点でアクションを実行することができます。

たとえば，仮想マシンのネットワーク・スループットを監視する場合，特定のスループット値を超えた時点で黄色のステート（警告）をトリガし，アラートを Operations Manager 2007 のオペレーション・コンソールに送信する設定が可能です。タスクとは，Operations Manager エージェント経由でリモート・サーバ上で稼働する Operations Manager 2007 オペレーション・コンソールから，ユーザが起動するアクションです。管理パックではタスクが事前定義されていますが，カスタム・タスクも追加で定義できます。

Hyper-V インフラストラクチャを System Center Operations Manager 2007 SP1 で監視する場合，管理パックをインポートし，これを使用してシステムの安定稼働を維持することになります。最小要件として，Windows Server 2008 のサポートを含む最新の Windows Server Base Operating System 管理パックが必要になります。これにより，オペレーティング・システム，サービス，ストレージ，ネットワーク，プロセッサ，メモリの可用性とパフォーマンスの監視が可能になります。

Hyper-V サーバ，仮想マシン，SCVMM 2008 サーバの監視に必要なツールは，System Center Virtual Machine 2008 管理パックにすべて統合されています。この管理パックは，Virtual Server 2005 R2，Hyper-V，VMware ESX サーバに関する監視とレポート作成機能を提供します。

---

**注：** VMWare ESX サーバを監視するには，SCVMM 2008 管理対象ホストとして設定する必要があります。

---

SCVMM 2008 管理パックは，ストレージ，メモリ，プロセッサ，物理ネットワーク，仮想ネットワーク，仮想マシンの数などについて，Hyper-V サーバのパフォーマンスを監視します。また，仮想プロセッサ，仮想ハードディスク，パススルー・ディスク，仮想マシンのメモリ使用率，仮想ネットワークなど，仮想マシンのパフォーマンスも監視できます。監視は，Hyper-V サーバにロードされている Operations Manager 2007 エージェントが行います。仮想マシンにも Operations Manager 2007 エージェントをインストールしている環境では，アプリケーション用の管理パックを Operations Manager 2007 にインポートすることによって，アプリケーションのパフォーマンス情報も取得できます。

SCVMM 2008 管理パックは，最新の監視情報，規則，レポートを提供します。

- **バーチャル・マシン使用率レポート**：仮想マシンの使用率に関する情報を表示します。選択した間隔について，仮想マシンのプロセッサ，メモリ，ディスク容量の使用率の平均値，合計，最大値を表示します。
- **ホスト使用率レポート**：稼働中の仮想マシンのホストあたりの数を表示します。選択した間隔とホスト・グループについて，ホストのプロセッサ，メモリ，ディスク容量の使用率の平均値，合計，最大値を表示します。
- **仮想化の候補レポート**：仮想マシンの候補となる物理コンピュータの検討に役立ちます。CPU，メモリ，ディスクの使用率，ハードウェア構成（プロセッサ速度，プロセッサの数，RAM の総容量など）に関するパフォーマンス・カウンタ群について，平均値を表示します。
- **ホスト使用率の増加レポート**：選択した間隔について，ホスト・リソースの増加率（パーセンテージ）と仮想マシンの数を表示します。
- **バーチャル・マシン割り当て**：仮想マシンについて，コスト・センターへのチャージバック計算の元となる情報を表示します。

System Center Virtual Machine Manager 2008 と System Center Operations Manager 2007 SP1 を統合することにより，Performance and Resource Optimization（PRO）が使用可能になります。PRO は，Operations Manager 2007 が提供するパフォーマンス情報を活用する Virtual Machine Manager の機能であり，仮想マシン・インフラストラクチャが効率的に動作できる理想的な環境を整えます。PRO は System Center Operations Manager 2007 の監視機能を拡張することによって，仮想化したハードウェア，オペレーティング・システム，アプリケーションの障害が原因でパフォーマンスが低下した状態にも対応できます。

PRO には，次の 2 つの応答オプションがあります。

- 発生した問題を解決する提案を行う際に，アラートを表示します。管理者は，ボタンをクリックするだけで提案された解決方法を実行できます。この解決方法では，組み込みのアクションによって，プロセッサ使用率のしきい値を超えた Hyper-V サーバから別の Hyper-V サーバへと仮想マシンを移動します。カスタム・アクションで PRO を拡張することも可能です。たとえば，Wake-on-LAN を使って事前設定したスタンバイ Hyper-V サーバを起動し，ニーズに応じてプールを動的に拡張できます。
- システムが推奨する解決方法を自動的に実行するオプションであり，管理者の操作は不要です。

## HP Sitescope による Hyper-V の監視

HP SiteScope は，Hyper-V インフラストラクチャを監視する包括的な機能を提供します。

SiteScope version 11.0 以降で提供される Hyper-V パフォーマンス・モニタでは，Hyper-V ベース・サーバのルート・パーティションと子パーティションを監視できます。

最初に作成するモニタでは，接続 URL を設定することによって，ソフトウェアにアクセスし，オブジェクト階層と使用可能なパフォーマンス・カウンタを動的に検出します。このパフォーマンス・カウンタの中から，SiteScope で取得する測定値を選択し，サーバ・ステータスに関するレポートを作成します。

# 関連する Windows カウンタ

ここでは，Hyper-V がインストールされているマシンのシステム・リソースを監視する際に，重要な役割を果たすカウンタを紹介します。

- **CPU :** 物理プロセッサを新しく追加する作業は簡単ではありません。すべての CPU に対して均等に負荷をかけることが必要です。次のカウンタを監視してください。
  - **% Processor Time/_Total**
- **メモリ :** Hyper-V はメモリを動的に管理し，オペレーティング・システムに対してメモリの要求と解放を行います。適切な動的オプションが選択されていることと，データベースが使用できる最大メモリ容量として，物理的な最大容量に近いレベルが割り当てられていることを確認してください。

  次のカウンタを監視してください。
  - **Available Bytes**
  - **Pages/sec**

- **ディスク**：すべてのアプリケーション層の中でも I/O に最も負荷がかかる操作が実行されるのはデータベースであるため，ディスク操作の監視は重要です。

  次のカウンタを監視してください。

  - **Current Disk Queue Length**
  - **Disk Bytes/sec**
  - **Disk Transfers/sec**

- **ネットワーク**：一部のアプリケーションでは，大量のデータがネットワーク経由で送信されると，大量のインタラクションが発生することがあります。次のカウンタを監視してください。

  - **Bytes Total/sec**
  - **Offloaded Connections**
  - **Packets/sec**
  - **Packets Outbound Errors**
  - **Packets Receive Errors**

# 最も重要なカウンタ

次の表では，CPU，メモリ，ネットワーク，ストレージ，一般という 5 つのカテゴリにカウンタを分類しています。

Hyper-V サーバと仮想マシンのパフォーマンスの監視で重要な役割を果たすカウンタをカテゴリ別にまとめます。

| | カウンタ | 説明 |
|---|---|---|
| CPU | %Guest Run Time | ゲスト・コードで経過したプロセッサ時間の割合（パーセンテージ）。 |
| | %Hypervisor Run Time | ハイパーバイザ・コードで経過したプロセッサ時間の割合（パーセンテージ）。 |
| | %Idle Time | アイドル状態で経過したプロセッサ時間の割合（パーセンテージ）。 |
| | %Total Run Time | ゲスト・コードとハイパーバイザ・コードで経過したプロセッサ時間の割合（パーセンテージ）。 |
| | %Guest Run Time (VPGRT) | ゲスト・コードで経過した仮想プロセッサ時間の割合（パーセンテージ）。 |
| | %Hypervisor Run Time | ハイパーバイザ・コードで経過した仮想プロセッサ時間の割合（パーセンテージ）。 |
| | %Total Run Time (VPTR) | ゲスト・コードとハイパーバイザ・コードで経過した仮想プロセッサ時間の割合（パーセンテージ）。 |
| | Total Intercepts/sec | ハイパーバイザのインターセプト・メッセージの発生頻度。 |
| | % Processor time/_Total | アイドルでない状態のスレッドの実行にプロセッサが費やした経過時間の割合（パーセンテージ）。 |

| | カウンタ | 説明 |
|---|---|---|
| メモリ | **1G GPA Pages** | パーティションの GPA 空間に存在する 1 G ページの数。 |
| | **2M GPA Pages** | パーティションの GPA 空間に存在する 2 M ページの数。 |
| | **Deposited Pages** | パーティションに格納されているページ数。 |
| | **Virtual Processors** | パーティションに存在する仮想プロセッサの数。 |
| | **Physical Pages Allocated** | 割り当てられた物理ページの数。 |
| | **Remote Physical Pages** | 優先 NUMA ノードから割り当てられていない物理ページの数。 |
| | **Available Bytes** | プロセスまたはシステム用にすぐに割り当て可能な物理メモリの容量（バイト単位）。 |
| | **Pages/sec** | ハード・ページ・フォルトを解決するために，ディスクに対して読み書きされるページの数。 |

| | カウンタ | 説明 |
|---|---|---|
| I/O | **Current Disk Queue Length** | パフォーマンス・データを収集した時点でディスク上に存在する未処理の要求の数。収集時にサービス内にある要求の数も含まれます。 |
| | **Disk Bytes/sec** | 書き込み/読み取り操作中にディスク間でデータが転送される速度（バイト数）。 |
| | **Disk Transfers/sec** | ディスクでの読み取り/書き込み操作の速度。 |
| | **Read Bytes/sec** | IDE コントローラに接続されているディスクから読み取られる 1 秒あたりのバイト数。 |
| | **Write Bytes/sec** | IDE コントローラに接続されているディスクに書き込まれる 1 秒あたりのバイト数。 |
| | **Read Sectors/sec** | IDE コントローラに接続されているディスクから読み取られる 1 秒あたりのセクタ数。 |
| | **Written Sectors/sec** | IDE コントローラに接続されているディスクに書き込まれる 1 秒あたりのセクタ数。 |
| | **Error Count** | 仮想デバイスで発生したエラー数の合計。 |
| | **Flush Count** | 仮想デバイスで発生したフラッシュ操作の回数の合計。 |
| | **Read Bytes/sec** | 仮想デバイスで 1 秒あたりに読み取ったバイト数の合計。 |
| | **Write Bytes/sec** | 仮想デバイスで 1 秒あたりに書き込んだバイト数の合計。 |
| | **Read Count** | 仮想デバイスで発生した読み取り操作の回数の合計。 |
| | **Write Count** | 仮想デバイスで発生した書き込み操作の回数の合計。 |

| | カウンタ | 説明 |
|---|---|---|
| ネットワーク | **Bytes Total/sec** | 各ネットワーク・アダプタでデータ・バイト（フレーム文字を含む）を送受信する速度。 |
| | **Offloaded Connections** | TCP Chimney オフロード対応ネットワーク・アダプタが現在処理している TCP 接続（IPv4 および IPv6）の数。 |
| | **Packets/sec** | ネットワーク・インタフェースでパケットを送受信する速度。 |
| | **Packets Outbound Errors** | エラーが原因で転送できなかった送信パケットの数。 |
| | **Packets Receive Errors** | エラーが含まれているために上位プロトコルに送信されなかった受信パケットの数。 |
| | **Bytes/sec** | 仮想スイッチを通過した 1 秒あたりのバイト数の合計。 |
| | **Packets/sec** | 仮想スイッチを通過した 1 秒あたりのパケット数の合計。 |
| | **Bytes Dropped** | ネットワーク・アダプタでドロップしたバイト数。 |
| | **Bytes Sent/sec** | ネットワーク・アダプタ経由で送信された 1 秒あたりのバイト数。 |
| | **Bytes Received/sec** | ネットワーク・アダプタ経由で受信された 1 秒あたりのバイト数。 |
| | **Bytes/sec** | ネットワーク・アダプタを通過したバイト数の合計。 |
| | **Packets/sec** | ネットワーク・アダプタ経由で受信された 1 秒あたりのバイト数の合計。 |

| | カウンタ | 説明 |
|---|---|---|
| 一般 | **Health Ok** | 安定状態で動作している仮想マシンの数。 |
| | **Health Critical** | システムの稼働状態が重大である仮想マシンの数。 |
| | **Logical Processors** | システムに存在する論理プロセッサの数。 |
| | **Partitions** | システムに存在するパーティション（仮想マシン）の数。 |
| | **Total Pages** | ハイパーバイザ内のブートストラップおよび格納されたページの数。 |
| | **Virtual Processors** | システムに存在する仮想プロセッサの数。 |
| | **Monitored Notifications** | ハイパーバイザで登録されている監視対象通知の数。 |

## CPU カウンタ

本項では，プロセッサの使用率に関する情報を提供するカウンタについて説明します。

### %Guest Run Time

| | |
|---|---|
| 正式名 | Hyper-V Hypervisor Logical Processor¥%Guest Run Time |
| カウンタのタイプ | サンプル間隔（ビジー状態の %） |
| 説明 | ゲスト・コードで経過したプロセッサ時間の割合（パーセンテージ）です。 |
| 使用方法 | 各 LP で実行されるゲスト・コードの時間の割合（パーセンテージ）を示し，_Total ですべての LP の平均値を示します。たとえば，2 つの LP と CPU テストを実行する VM が 1 つある場合，LP(0) が 95%，LP(1) が 0%，_Total が 47.5% と表示されます。VM は LP(0) で動作します。 |
| パフォーマンス | なし |
| 操作 | なし |
| しきい値 | なし |

### %Hypervisor Run Time

| | |
|---|---|
| 正式名 | Hyper-V Hypervisor Logical Processor¥%Hypervisor Run Time |
| カウンタのタイプ | サンプル間隔（ビジー状態の %） |
| 説明 | ハイパーバイザ・コードで経過したプロセッサ時間の割合（パーセンテージ）です。 |
| 使用方法 | 各 LP で実行されるハイパーバイザ・コードの時間の割合（パーセンテージ）を示し，_Total ですべての LP の平均値を示します。プロセッサ・カウンタ・セットの % Kernel Run Time に類似したカウンタです。 |
| パフォーマンス | なし |
| 操作 | なし |
| しきい値 | なし |

### %Idle Time

| | |
|---|---|
| 正式名 | Hyper-V Hypervisor Logical Processor¥%Idle Time |
| カウンタのタイプ | サンプル間隔（ビジー状態の %） |
| 説明 | アイドル状態で経過したプロセッサ時間の割合（パーセンテージ）です。 |
| 使用方法 | LP の待機時間の割合（パーセンテージ）を示し，_Total ですべての LP の平均値を示します。プロセッサ・カウンタ・セットの % Kernel Run Time に類似したカウンタです。 |
| パフォーマンス | なし |
| 操作 | なし |
| しきい値 | なし |

## %Total Run Time

| | |
|---|---|
| 正式名 | Hyper-V Hypervisor Logical Processor¥%Total Run Time (LPTR) |
| カウンタのタイプ | サンプル間隔（ビジー状態の %） |
| 説明 | ゲスト・コードとハイパーバイザ・コードで経過したプロセッサ時間の割合（パーセンテージ）です。このカウンタは LPTR とも呼ばれます。 |
| 使用方法 | このカウンタの値は，%Guest Run Time と % Hypervisor Runtime の和です。このカウンタは 100% を若干（0.5% 未満）上回ることがありますが，パフォーマンス・カウンタの計算方法が原因で発生します。たとえば，現在時刻の値，値 1，終了時刻の値，値 2 の順番で取得すると，終了時刻の読み取りと値 2 の読み取りの間に値 2 が増加してしまう可能性があります。したがって，開始時刻の値，値 1，値 2，終了時刻の値という順序に変更すると，値は常に 100 を若干下回るようになります。 |
| パフォーマンス | LPTR は，ホスト内の論理プロセッサがどの程度ビジー状態かを示します。 |
| 操作 | なし |
| しきい値 | なし |

## %Guest Run Time (VPGRT)

| | |
|---|---|
| 正式名 | Hyper-V Hypervisor Root Virtual Processor¥%Guest Run Time (VPGRT)<br>**または**<br>Hyper-V Hypervisor Virtual Processor¥%Guest Run Time (VPGRT) |
| カウンタのタイプ | サンプル間隔（ビジー状態の %） |
| 説明 | ゲスト・コードで経過した仮想プロセッサ時間の割合（パーセンテージ）です。 |
| 使用方法 | ゲスト VM については，ゲスト VP が LP 上のハイパーバイザ以外のコードで稼働した時間の割合（パーセンテージ）を示し，_Total ですべてのゲスト VP の合計値を示します。 またルートについては，ルート VP が LP 上のハイパーバイザ以外のコードで稼働した時間の割合（パーセンテージ）を示し，_Total ですべてのルート VP の合計値を示します。ゲスト VP とルート VP の _Total を合計すると，論理プロセッサのカウンタ・セットの % Guest Run Time _Total と等しくなります。 |
| パフォーマンス | VPGRT は，ゲスト内の仮想プロセッサがどの程度ビジーかを示します。<br>このカウンタは，時刻のずれの影響を若干受けます。 |
| 操作 | なし |
| しきい値 | なし |

## %Hypervisor Run Time

| | |
|---|---|
| 正式名 | Hyper-V Hypervisor Root Virtual Processor¥%Hypervisor Run Time<br>**または**<br>Hyper-V Hypervisor Virtual Processor¥%Hypervisor Run Time |
| カウンタのタイプ | サンプル間隔（ビジー状態の %） |
| 説明 | ハイパーバイザ・コードで経過した仮想プロセッサ時間の割合（パーセンテージ）です。 |
| 使用方法 | ゲスト VM については，ゲスト VP が LP 上のハイパーバイザ・コードで稼働した時間の割合（パーセンテージ）を示し，_Total ですべてのゲスト VP の合計値を示します。 またルートについては，ルート VP が LP 上のハイパーバイザ・コードで稼働した時間の割合（パーセンテージ）を示し，_Total ですべてのルート VP の合計値を示します。ゲスト VP とルート VP の _Total を合計すると，論理プロセッサのカウンタ・セットの % Hypervisor Run Time _Total と等しくなります。 |
| パフォーマンス | なし |
| 操作 | なし |
| しきい値 | % Hypervisor Time の適正値は 25% 未満です。これよりも大きくなると，統合サービスがインストールされていない可能性があります。 |

### %Total Run Time (VPTR)

| | |
|---|---|
| 正式名 | Hyper-V Hypervisor Root Virtual Processor¥%Total Run Time (VPTR)<br>**または**<br>Hyper-V Hypervisor Virtual Processor¥%Total Run Time (VPTR) |
| カウンタのタイプ | サンプル間隔（ビジー状態の %） |
| 説明 | ゲスト・コードとハイパーバイザ・コードで経過した仮想プロセッサ時間の割合（パーセンテージ）です。 |
| 使用方法 | このカウンタの値は，%Guest Run Time と % Hypervisor Runtime の和を VP ごとに示しています。Root Virtual Processor カウンタと Virtual Processor カウンタの %Total Run Time を合計すると，すべての論理プロセッサ・カウンタの %Total Run Time の合計と等しくなります。 |
| パフォーマンス | VPTR は, ホスト内の仮想プロセッサがどの程度ビジーかを示します。 |
| 操作 | なし |
| しきい値 | なし |

### Total Intercepts/sec

| | |
|---|---|
| 正式名 | Hyper-V Hypervisor Root Virtual Processor¥%Total Intercepts/sec<br>**または**<br>Hyper-V Hypervisor Virtual Processor¥%Total Intercepts/sec |
| カウンタのタイプ | サンプル間隔での差異（1 秒あたりの速度） |
| 説明 | ハイパーバイザのインターセプト・メッセージの発生頻度です。 |
| 使用方法 | ゲスト VP がハイパーバイザでのサービス実行のために現在のモードを終了する必要がある場合，インターセプトが発生します。一般的にインターセプトが発生する原因には，ゲスト物理アドレス（GPA）からサーバ物理アドレス（SPA）への変換，hlt/cupid/in/out などの特権命令，VP のスケジュールされたタイム・スライスの終了などがあります。 |
| パフォーマンス | なし |
| 操作 | なし |
| しきい値 | なし |

### % Processor time/_Total

---

**注：** このカウンタは Hyper-V 固有のカウンタではなく，標準的な Windows のリソース・カウンタの 1 つです。詳細については，268 ページ「関連する Windows カウンタ」を参照してください。

---

## メモリ・カウンタ

本項では，Hyper-V システムのメモリ管理に関連するカウンタについて説明します。このカウンタは，メモリ消費量やメモリ・プールなどの統計を表示します。

### 1G GPA Pages

| | |
|---|---|
| 正式名 | Hyper-V Hypervisor Partition¥1G GPA Pages<br>**または**<br>Hyper-V Hypervisor Root Partition¥1G GPA Pages |
| カウンタのタイプ | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| 説明 | パーティションの GPA 空間に存在する 1 G ページの数です。 |
| 使用方法 | VM がサイズの大きなページを使用しているかどうかを示します。サイズの大きなページを使用することにより VM パフォーマンス全体が向上します。 |
| パフォーマンス | なし |
| 操作 | なし |
| しきい値 | なし |

## 2M GPA Pages

| | |
|---|---|
| 正式名 | Hyper-V Hypervisor Partition¥2M GPA Pages<br>**または**<br>Hyper-V Hypervisor Root Partition¥2M GPA Pages |
| カウンタのタイプ | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| 説明 | パーティションの GPA 空間に存在する 2 M ページの数です。 |
| 使用方法 | なし |
| パフォーマンス | なし |
| 操作 | なし |
| しきい値 | なし |

## Deposited Pages

| | |
|---|---|
| 正式名 | Hyper-V Hypervisor Partition¥Deposited Pages<br>**または**<br>Hyper-V Hypervisor Root Partition¥Deposited Pages |
| カウンタのタイプ | 期間内の値（一定期間にわたってサンプリング） |
| 説明 | パーティション内に格納されているページ数です。 |
| 使用方法 | VM の管理用にハイパーバイザが使用しているメモリ容量を示します。 |
| パフォーマンス | なし |
| 操作 | なし |
| しきい値 | なし |

### Virtual Processors

| | |
|---|---|
| 正式名 | Hyper-V Hypervisor Partition¥Virtual Processors<br>**または**<br>Hyper-V Hypervisor Root Partition¥Virtual Processors |
| カウンタのタイプ | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| 説明 | パーティションに存在する仮想プロセッサの数です。 |
| 使用方法 | VM で設定されている仮想プロセッサの数を示します。 |
| パフォーマンス | なし |
| 操作 | なし |
| しきい値 | なし |

### Physical Pages Allocated

| | |
|---|---|
| 正式名 | Hyper-V VM Vid Partition¥Physical Pages Allocated |
| カウンタのタイプ | 期間内の値（一定期間にわたってサンプリング） |
| 説明 | 割り当てられた物理ページの数です。 |
| 使用方法 | VM の管理に必要なゲスト・ページと VID ページの数の合計です。 |
| パフォーマンス | なし |
| 操作 | なし |
| しきい値 | なし |

### Remote Physical Pages

| | |
|---|---|
| 正式名 | Hyper-V VM Vid Partition¥Remote Physical Pages |
| カウンタのタイプ | 期間内の値（一定期間にわたってサンプリング） |
| 説明 | 優先 NUMA ノードから割り当てられていない物理ページの数です。 |
| 使用方法 | NUMA ベース・システムにおいて，VM が複数のノードにまたがっているかどうかを確認できます。可能であれば，複数ノードにまたがる構成を避けることをお勧めします。 |
| パフォーマンス | なし |
| 操作 | なし |
| しきい値 | なし |

### Available Bytes

---

**注：** このカウンタは Hyper-V 固有のカウンタではなく，標準的な Windows のリソース・カウンタの 1 つです。詳細については，268 ページ「関連する Windows カウンタ」を参照してください。

---

### Pages/sec

---

**注：** このカウンタは Hyper-V 固有のカウンタではなく，標準的な Windows のリソース・カウンタの 1 つです。詳細については，268 ページ「関連する Windows カウンタ」を参照してください。

---

## I/O カウンタ

本項では，Hyper-V のストレージに関するカウンタについて説明します。

次の I/O カウンタは Hyper-V 固有のカウンタではなく，標準的な Windows のカウンタです。詳細については，268ページ「関連する Windows カウンタ」を参照してください。

- Current Disk Queue Length
- Disk Bytes/sec
- Disk Transfers/sec

### Read Bytes/sec

| | |
|---|---|
| 正式名 | Hyper-V Virtual IDE Controller¥Read Bytes / Sec |
| カウンタのタイプ | サンプル間隔での差異（1 秒あたりの速度） |
| 説明 | IDE コントローラに接続されているディスクから 1 秒あたりに読み取られるバイト数です。 |
| 使用方法 | 仮想 IDE のカウンタは，Hyper-V Virtual IDE Controller カウンタ・セットで表示されます。 |
| パフォーマンス | なし |
| 操作 | なし |
| しきい値 | なし |

### Write Bytes/sec

| | |
|---|---|
| 正式名 | Hyper-V Virtual IDE Controller¥Write Bytes / Sec |
| カウンタのタイプ | サンプル間隔での差異（1 秒あたりの速度） |
| 説明 | IDE コントローラに接続されているディスクに 1 秒あたりに書き込まれるバイト数です。 |
| 使用方法 | なし |
| パフォーマンス | なし |
| 操作 | なし |
| しきい値 | なし |

### Read Sectors/sec

| | |
|---|---|
| 正式名 | Hyper-V Virtual IDE Controller¥Read Sectors / Sec |
| カウンタのタイプ | サンプル間隔での差異（1 秒あたりの速度） |
| 説明 | IDE コントローラに接続されているディスクから 1 秒あたりに読み取られるセクタ数です。 |
| 使用方法 | なし |
| パフォーマンス | なし |
| 操作 | なし |
| しきい値 | なし |

### Written Sectors/sec

| | |
|---|---|
| 正式名 | Hyper-V Virtual IDE Controller¥Written Sectors / Sec |
| カウンタのタイプ | サンプル間隔での差異（1 秒あたりの速度） |
| 説明 | IDE コントローラに接続されているディスクに 1 秒あたりに書き込まれるセクタ数です。 |
| 使用方法 | なし |
| パフォーマンス | なし |
| 操作 | なし |
| しきい値 | なし |

### Error Count

| | |
|---|---|
| 正式名 | Hyper-V Virtual Storage Device¥Error Count |
| カウンタのタイプ | 期間内の値（一定期間にわたってサンプリング） |
| 説明 | この仮想デバイスで発生したエラー数の合計です。 |
| 使用方法 | 統合サービスがロードされていない場合，仮想 IDE と SCSI 両方のアクティビティが Hyper-V Virtual Storage Device カウンタ・セットで表示されます。 |
| パフォーマンス | なし |
| 操作 | なし |
| しきい値 | なし |

### Flush Count

| | |
|---|---|
| 正式名 | Hyper-V Virtual Storage Device¥Flush Count |
| カウンタのタイプ | 期間内の値（一定期間にわたってサンプリング） |
| 説明 | この仮想デバイスで発生したフラッシュ操作の数の合計です。 |
| 使用方法 | なし |
| パフォーマンス | なし |
| 操作 | なし |
| しきい値 | なし |

### Read Bytes/sec

| | |
|---|---|
| 正式名 | Hyper-V Virtual Storage Device¥Read Bytes / Sec |
| カウンタのタイプ | サンプル間隔での差異（1 秒あたりの速度） |
| 説明 | 仮想デバイスで 1 秒あたりに読み取られたバイト数の合計です。 |
| 使用方法 | なし |
| パフォーマンス | なし |
| 操作 | なし |
| しきい値 | なし |

### Write Bytes/sec

| | |
|---|---|
| 正式名 | Hyper-V Virtual Storage Device¥Write Bytes / Sec |
| カウンタのタイプ | サンプル間隔での差異（1 秒あたりの速度） |
| 説明 | 仮想デバイスで 1 秒あたりに書き込まれたバイト数の合計です。 |
| 使用方法 | なし |
| パフォーマンス | なし |
| 操作 | なし |
| しきい値 | なし |

### Read Count

| | |
|---|---|
| 正式名 | Hyper-V Virtual Storage Device¥Read Count |
| カウンタのタイプ | 期間内の値（一定期間にわたってサンプリング） |
| 説明 | この仮想デバイスで発生した読み取り操作の数の合計です。 |
| 使用方法 | なし |
| パフォーマンス | なし |
| 操作 | なし |
| しきい値 | なし |

### Write Count

| | |
|---|---|
| 正式名 | Hyper-V Virtual Storage Device¥Write Count |
| カウンタのタイプ | 期間内の値（一定期間にわたってサンプリング） |
| 説明 | この仮想デバイスで発生した書き込み操作の数の合計です。 |
| 使用方法 | なし |
| パフォーマンス | なし |
| 操作 | なし |
| しきい値 | なし |

## ネットワーク・カウンタ

本項では，Windows リソースのネットワーク・コンポーネントに関連するカウンタについて説明します。

次のネットワーク・カウンタは Hyper-V 固有のカウンタではなく，標準的な Windows のカウンタです。詳細については，268 ページ「関連する Windows カウンタ」を参照してください。

- Bytes Total/sec
- Offloaded Connections
- Packets/sec
- Packets Outbound Errors
- Packets Receive Errors

### Bytes/sec

| | |
|---|---|
| 正式名 | Hyper-V Virtual Switch¥Bytes/Sec |
| カウンタのタイプ | サンプル間隔での差異（1 秒あたりの速度） |
| 説明 | 仮想スイッチを通過した 1 秒あたりのバイト数の合計です。 |
| 使用方法 | なし |
| パフォーマンス | なし |
| 操作 | なし |
| しきい値 | なし |

### Packets/sec

| | |
|---|---|
| 正式名 | Hyper-V Virtual Switch¥Packets/Sec |
| カウンタのタイプ | サンプル間隔での差異（1 秒あたりの速度） |
| 説明 | 仮想スイッチを通過した 1 秒あたりのパケット数の合計です。 |
| 使用方法 | なし |
| パフォーマンス | なし |
| 操作 | なし |
| しきい値 | なし |

### Bytes Dropped

| | |
|---|---|
| 正式名 | Hyper-V Legacy Network Adapter¥Bytes Dropped |
| カウンタのタイプ | 期間内の値（一定期間にわたってサンプリング） |
| 説明 | ネットワーク・アダプタでドロップしたバイト数です。 |
| 使用方法 | 統合サービスをインストールする前に VM を使用するには，レガシー・ネットワーク・アダプタが必要です。VM で統合サービスを使用する環境が整った時点で，ネットワーク・アダプタを使用してください。 |
| パフォーマンス | なし |
| 操作 | なし |
| しきい値 | なし |

### Bytes Sent/sec

| | |
|---|---|
| 正式名 | Hyper-V Legacy Network Adapter¥Bytes Sent / Sec |
| カウンタのタイプ | サンプル間隔での差異（1 秒あたりの速度） |
| 説明 | ネットワーク・アダプタ経由で 1 秒あたりに送信されたバイト数です。 |
| 使用方法 | なし |
| パフォーマンス | なし |
| 操作 | なし |
| しきい値 | なし |

### Bytes Received/sec

| | |
|---|---|
| 正式名 | Hyper-V Legacy Network Adapter¥Bytes Received/ Sec |
| カウンタのタイプ | サンプル間隔での差異（1 秒あたりの速度） |
| 説明 | ネットワーク・アダプタ経由で 1 秒あたりに受信されたバイト数です。 |
| 使用方法 | なし |
| パフォーマンス | なし |
| 操作 | なし |
| しきい値 | なし |

### Bytes/sec

| | |
|---|---|
| 正式名 | Hyper-V Virtual Network Adapter¥Bytes / Sec |
| カウンタのタイプ | サンプル間隔での差異（1 秒あたりの速度） |
| 説明 | ネットワーク・アダプタを通過したバイト数の合計です。 |
| 使用方法 | なし |
| パフォーマンス | なし |
| 操作 | なし |
| しきい値 | なし |

### Packets/sec

| | |
|---|---|
| 正式名 | Hyper-V Virtual Network Adapter¥Packets / Sec |
| カウンタのタイプ | サンプル間隔での差異（1 秒あたりの速度） |
| 説明 | ネットワーク・アダプタ経由で 1 秒あたりに受信されたバイト数の合計です。 |
| 使用方法 | なし |
| パフォーマンス | なし |
| 操作 | なし |
| しきい値 | なし |

## 一般的なカウンタ

本項では，Hyper-V システム全般のパフォーマンスに関するカウンタについて説明します。

### Health Ok

| | |
|---|---|
| 正式名 | Hyper-V Virtual Machine Health Summary¥Health Ok |
| カウンタのタイプ | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| 説明 | 稼働状態が正常な仮想マシンの数です。 |
| 使用方法 | リソース（特にディスク）の稼働状態が「重大」である場合，リソース不足または修復できないエラーが発生しています。 |
| パフォーマンス | なし |
| 操作 | なし |
| しきい値 | なし |

### Health Critical

| | |
|---|---|
| 正式名 | Hyper-V Virtual Machine Health Summary¥Health Critical |
| カウンタのタイプ | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| 説明 | 稼働状態に重大な問題が発生している仮想マシンの数です。 |
| 使用方法 | 稼働状態が「重大」である場合，原因を調査してください。 |
| パフォーマンス | なし |
| 操作 | なし |
| しきい値 | なし |

### Logical Processors

| | |
|---|---|
| 正式名 | Hyper-V Hypervisor¥Logical Processors |
| カウンタのタイプ | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| 説明 | システム内に存在する論理プロセッサの数です。 |
| 使用方法 | ハイパーバイザで管理しているコア /HT の数を示します。デュアル・プロセッサをベースにするクアッド・コアで HT が無効になっている場合は 8 に設定します。HT が有効になっている場合は 16 に設定します。現在，この値は固定値であり，起動後は変更できません。今後のリリースでは，プロセッサのホット・アドやホット・リムーブをサポートし，動的設定が可能になる予定です。 |
| パフォーマンス | なし |
| 操作 | なし |
| しきい値 | なし |

## Partitions

| | |
|---|---|
| 正式名 | Hyper-V Hypervisor¥Partitions |
| カウンタのタイプ | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| 説明 | システムに存在するパーティション（仮想マシン）の数です。 |
| 使用方法 | システム上の仮想マシンは，それぞれがパーティションと呼ばれるコンテナ内で動作します。Hyper-V で「ルート」と呼ばれる「ホスト OS」もパーティションで稼働するため, VM を稼働していない場合でもこの値は 1 になります。したがって，たとえばゲスト VM が 2 つある場合は 3 となり，ゲストの数にルート分の 1 を加えた値になります。 |
| パフォーマンス | なし |
| 操作 | なし |
| しきい値 | なし |

## Total Pages

| | |
|---|---|
| 正式名 | Hyper-V Hypervisor¥Total Pages |
| カウンタのタイプ | 期間内の値（一定期間にわたってサンプリング） |
| 説明 | ハイパーバイザ内のブートストラップおよび格納されたページの数です。 |
| 使用方法 | ハイパーバイザでは，仮想 TLB 内にある仮想プロセッサの数や，ゲスト仮想アドレスからシステム物理アドレスへの変換エントリなどを追跡する際，メモリを必要とします。したがって Total Pages は，ハイパーバイザがパーティション管理に使用するメモリ容量の合計を示します。1 ページのサイズは 4KB ですが，ゲストのサポートに使用する合計容量ではありません。また，ワーカ・プロセス（vmwp.exe）のサイズと vid でのメモリ容量の確認も必要です。RTM 以外のリリースでは数値は提供されないため，オーバーヘッドの値となります（今後対応の予定）。Total Pages の値は，稼働しているゲスト VM によって変動することがあります。 |
| パフォーマンス | なし |
| 操作 | なし |
| しきい値 | なし |

## Virtual Processors

| | |
|---|---|
| 正式名 | Hyper-V Hypervisor¥Virtual Processors |
| カウンタのタイプ | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| 説明 | システムに存在する仮想プロセッサの数です。 |
| 使用方法 | ルート・パーティションと子パーティション（ゲスト VM が動作する場所）内の実行は，すべて仮想プロセッサ（VP）で行われます。各論理プロセッサ（LP）には VP が 1 つ以上あり，これがルート VP です。さらに，ゲストとして VP を設定すると，カウンタの値は VP ごとに 1 ずつ増えます。たとえば，8LP システムで 2 つの VP を稼働するゲストが 1 つある場合，このカウンタの数値は 10 になります。 |
| パフォーマンス | なし |
| 操作 | なし |
| しきい値 | なし |

## Monitored Notifications

| | |
|---|---|
| 正式名 | Hyper-V Hypervisor¥Monitored Notifications |
| カウンタのタイプ | 期間内の値（一定期間にわたってサンプリング） |
| 説明 | ハイパーバイザで登録されている監視対象通知の数です。 |
| 使用方法 | 仮想化によるオーバーヘッド軽減のために，ハイパーバイザで採用されている割り込みの結合手法の一部として提供されます。たとえば，ゲストがネットワーク経由でデータを転送する場合，実際の I/O を実行するルート VP に対してパケットごとに割り込みを送信する方法と，ルート VP に割り込みを 1 回送信し，データ・フローの開始を通知する方法があります。このカウンタは，ルートとゲストに送信する割り込みの「フロー」の数を示します。 |
| パフォーマンス | なし |
| 操作 | なし |
| しきい値 | なし |

# 最適化とチューニング

本項では，Hyper-V ホスト（サーバ）と仮想マシンのパフォーマンス最適化に関するベスト・プラクティスと推奨事項を紹介します。

## サーバの最適化とチューニング

Hyper-V サーバでパフォーマンスを最大限に引き出すには，次の 4 つの領域に注目します。

プロセッサ，メモリ，I/O，ネットワーク

### CPU パフォーマンスのベスト・プラクティス

- Hyper-V でプロセッサのパフォーマンスを最大限に引き出すためには，Windows Server 2008 の Server Core と，Hyper-V の役割またたは Microsoft Hyper-V Server 2008 をインストールすることをお勧めします。Hyper-V の役割のみを親パーティションにロードして Windows Server 2008 を稼働すると，親パーティションが必要とするプロセッサ処理能力を最小限に抑え，子パーティションに割り当てる処理能力が増えます。
- マルチコア・プロセッサ搭載のサーバをお勧めします。最新のマルチコア・プロセッサを採用することで，最高レベルのパフォーマンスを発揮できます。
- 大容量キャッシュ（L2/L3）搭載のプロセッサを選択することで，パフォーマンスを向上できます。

### メモリ・パフォーマンスのベスト・プラクティス

- Hyper-V サーバでメモリ・パフォーマンスを最大限に引き出すためには，最高速のメモリを選択してください。パフォーマンスを引き出すだけでなく，サーバが対応可能な仮想マシンの数も増やすのであれば，メモリを最大容量まで搭載し，高速プロセッサや高速ディスク・サブシステムなどのコンポーネントとのバランスを取ってください。

- メモリ拡張でスロットが無駄にならないように，最高レベルの密度のモジュールサーバを購入してください。たとえば，16 GB の RAM をインストールする場合，1 GB モジュール 16 個，2 GB モジュール 8 個，4 GB モジュール 4 個，8 GB モジュール 2 個という組み合わせがありますが，8 GB モジュールを使用することによって，少ないスロットで多くのメモリ容量をインストールできます。このような方法で拡張すれば，メモリ・モジュールの追加時に低密度モジュールを取り外す必要はなくなります。
- サーバに搭載する RAM を計算する際には，物理サーバの RAM 1 GB 以上を親パーティションに割り当ててください。
- パフォーマンスを最適化するには，NUMA（Non-Uniform Memory Access）システム上の仮想マシン・メモリに割り当てる最大メモリ容量を計算してから，その容量以上のメモリを各プロセッサ用に購入します。ローカル・ノードができるだけ多くのメモリ容量を使用できるようにし，別のノードのメモリを呼び出す回数を減らすためには，すべてのプロセッサに対してメモリを均等に分散する必要があります。Hyper-V の仮想マシンに搭載できる RAM は最大 64 GB なので，メモリの最大容量を割り当てた上で，その最大容量が各プロセッサ・ノードに搭載されるようにするのは不可能な場合があります。
- Windows Server 2008 Server のインストールでは，フルインストールではなく，Core インストール・オプションと Hyper-V の役割を選択します。これにより，子パーティションに割り当てるサーバ RAM を約 80MB 増やすことができます。

### I/O パフォーマンスのベスト・プラクティス

- ウイルス対策アプリケーションでは，ファイル拡張子またはプロセスを除外する設定を行ってください。ファイル拡張子ではなくプロセスを除外する方がセキュリティを強化できるので，こちらをお勧めします。ウイルス対策ソフトウェアで Hyper-V 管理プロセスを除外するには，Hyper-V 仮想マシン管理サービス（VMMS.exe）と仮想マシン・ワーカ・プロセス・サービス（Vmwp.exe）を除外してください。ウイルス対策ソフトウェアでプロセスを除外する設定ができない場合，.vhd，.avhd，.vfd，.vsv，.xml の各ファイル拡張子を除外リストに追加し，スキャン対象から除外します。

- 仮想マシンの読み取り/書き込み回数を最小限に抑えるためには，Hyper-V サーバに搭載するドライブは 10,000 RPM（1 分あたりの回転数）以上のものを選択してください。7200 RPM ではなく 10,000 RPM のドライブを搭載することにより，1 分あたりに実行される読み取り/書き込み操作の数が大幅に増えます。
- Hyper-V サーバで SATA または SAS ドライブを使用すると，1 つのハードディスクを複数の I/O 要求に使用することができ，操作の実行順序を動的に変更することができるので，パフォーマンスが向上します。
- 内蔵ハードドライブを使用して起動する Hyper-V サーバでは，RAID 1（ミラーリング）を使用することで，親パーティションと Hyper-V 設定のフォールト・トレランスを実現できます。
- 仮想マシンのストレージについては，RAID 0（ストライピング）＋ 1（ミラーリング）の冗長性を備えた SAN，iSCSI ターゲット機能，RPM が高く I/O コマンド・キューイング対応のハードドライブを使用します。SATA と SAS の両タイプをサポートするエンクロージャを選択すると，非常に柔軟な構成が可能になります。RAID 0 ＋ 1 ディスク・アレイを構成する場合は，スピンドルをできるだけ多くして I/O 負荷を分散してください。

## ネットワーク・パフォーマンスのベスト・プラクティス

- Hyper-V の管理とバックアップに専用の物理ネットワーク・アダプタを 1 つ以上使用します。Hyper-V 管理専用のネットワーク・アダプタを使用することにより，Hyper-V サーバの管理とバックアップで発生するネットワーク・トラフィックが仮想マシンのトラフィックに影響を与えなくなります。
- iSCSI 通信専用のネットワーク・アダプタを用意し，iSCSI 処理をハードウェアでサポートするアダプタを使用します。
- クラスタ通信専用のネットワーク・アダプタを用意します。
- Hyper-V ホスト・クラスタでは，TCP Chimney オフロードは有効にしないでください。Windows Server 2008 のフェイルオーバ・クラスタリングは，TCP Chimney オフロード機能をサポートしません。
- クラスタ化されていない Hyper-V サーバ上では，TCP Chimney オフロードを有効にします。外部仮想ネットワークにバインドされている物理ネットワーク・アダプタは TCP オフロード・エンジンを使用しませんが，他のアダプタは使用します。

- TCP Chimney を有効にする場合は，その前後でアプリケーション・パフォーマンスをテストします。すべてのアプリケーションが TCP Chimney オフロードを利用できるわけではなく，一部のネットワーク・アダプタは TCP Chimney オフロードによって発生した負荷に対応できるだけの処理能力を備えていません。いずれの場合も，TCP Chimney オフロードを有効にすると一部のアプリケーションでネットワーク・パフォーマンスが低下してしまいます。
- Jumbo Frame を有効にすると，一度に送信できるデータ量が増大して送信パケット数が少なくなるので，処理するフレーム・ヘッダの数も少なくなることから，プロセッサのオーバーヘッド低下とスループット向上につながります。
- Jumbo Frame は，大量のデータをネットワーク経由で送信するアプリケーションやプロトコルのパフォーマンスを大幅に向上する効果があります。
- チーム構成したすべてのネットワーク・アダプタで，受信側のロード・バランシング（異なるクライアントが送信した ARP 要求に対して異なる MAC アドレスで応答）を無効にします。有効になっていると，チーム構成したネットワーク・アタプタにバインドされた外部仮想ネットワークに仮想マシンが接続すると，Hyper-V サーバへの外部通信ができなくなります。

## 仮想マシンの最適化とチューニング

仮想マシンでパフォーマンスを最大限に引き出すには，次の 4 つの領域に注目します。プロセッサ，メモリ，I/O，ネットワーク。

### CPU パフォーマンスのベスト・プラクティス

- Hyper-V の子パーティションでパフォーマンスを最適化するには，Windows Server 2008 または最新のサーバ・オペレーティング・システムを使用します。Windows 2000 Server または Windows Server 2003 仮想マシンを Windows Server 2008 へ移行することにより，仮想マシンのパフォーマンスは向上し，Hyper-V サーバの負荷を軽減できます。
- エミュレート・デバイスのパフォーマンスとスループットを向上するには，サポート対象のゲスト・オペレーティング・システムの最初のアクションの 1 つとして統合サービスをインストールしてください。

- 仮想マシンへの移行では，既存の物理サーバのプロセッサの使用状況を確認してください。物理サーバは，最小仕様または標準仕様の状態で購入されている可能性があります。複数のプロセッサが稼働していないことがわかったら，仮想マシンへの移行後，シングル・プロセッサとして構成し，プロセッサ使用率を監視します。
- 仮想 CD/DVD ドライブが不要な場合は，仮想マシンから削除します。ドライブを使用しない場合でも，メディア挿入時に CD/DVD ドライブが CPU サイクルを使用しているかどうかを定期的にチェックする必要があります。
- PXE ブートまたは統合サービスがないために合成ネットワーク・アダプタをサポートできないオペレーティング・システムでは，レガシー・ネットワーク・アダプタを使用します。レガシー・ネットワーク・アダプタを使用すると，仮想マシンのワーカ・プロセスでパケットを処理する際に，多くのホスト・プロセッサ時間が必要になります。また，仮想化スタックを利用できないため，合成アダプタよりもスループットが低くなります。
- 日常業務でレガシー・ネットワーク・アダプタを使った通信を行う仮想マシンがあれば，別の Hyper-V サーバに分離します。これにより，レガシー・アダプタによるプロセッサ・オーバーヘッドが，合成ネットワーク・アダプタを使用する Hyper-V サーバのパフォーマンスやスケーラビリティに影響しなくなります。
- 信頼されたユーザのみにコンソール･アクセスが限定されているマシンでは, スクリーン・セーバーを無効にすることで，アイドルなプロセッサ・サイクルをなくすことができます。スクリーン・セーバーが必要なマシンについては，不正アクセス防止のためにコンソールをロックすることによって, 画像がなくキー･シーケンスのみをチェックするスクリーン・セーバーを使用できます。
- 仮想マシンのプロセッサ，ネットワーク，ディスク I/O の作業負荷プロファイルを分析し，仮想マシンを追加することによって既存の作業負荷プロファイルにどのような影響が発生するかを把握します。パフォーマンスと I/O について同じ特徴を持つ VM を組み合わせることはお勧めしません。
- 仮想マシンが必要なときに必要な処理能力を必ず確保できるようにするには，容量予約を設定します（論理プロセッサの所定の割合を確保）。

- Hyper-V サーバ環境の仮想マシンで，プロセッサの使用率にスパイクが発生し，ホスト上にある他のマシンに影響を与えている場合，この仮想マシンに容量制限を適用し，パフォーマンスを予測可能な状態になるように管理します。
- VPTR が高く LPTR が低い場合は，処理能力が十分に割り当てられていない仮想マシンが存在することを示しています。各仮想マシンの VPGRT カウンタを監視してプロセッサ使用率が高い仮想マシンを特定し，そのマシンに仮想プロセッサを追加してください。仮想プロセッサの追加がゲスト・オペレーティング・システムでサポートされていない場合は，仮想マシンを追加して作業負荷を分散することによって，アプリケーションのスケールアウトを図ります。
- LPTR が高く VPTR が低い場合，軽い負荷を処理している仮想マシンが多数存在することを示しています。仮想マシン間でコンテキスト切り替えが起こると，ホスト・プロセッサのボトルネックにつながります。ホスト上で動作する仮想マシンのプロセッサ使用率にスパイクが発生する場合，その仮想マシンが他の仮想マシンの処理能力を消費してしまう可能性と，追加で必要になった処理能力を取得できずにパフォーマンスが低下しまう可能性があります。このような状況が頻繁に発生するのは最適な環境とはいえません。Hyper-V サーバを追加し，仮想マシンをそのホストに移動することを検討してください。
- VPTR と LPTR の両方が高い場合，Hyper-V サーバ・プロセッサが過負荷状態であることを示しています。Hyper-V サーバを追加して，既存の仮想マシンをサーバ間でバランスよく分散してください。

### メモリ・パフォーマンスのベスト・プラクティス

- 親パーティションには，1 GB 以上のメモリ容量を確保します。VM に十分な容量のメモリを割り当てることによって，通常稼働時に発生するディスクへのページングを最小限に抑えることができます。ただし，回避することはできません。
- ¥Memory¥Available Mbytes カウンタが常に 10% 未満で，¥Memory¥Pages/sec カウンタが 1000 を超える場合，仮想マシンに割り当てる RAM 容量を増やしてください。¥Memory¥Available Mbytes カウンタが常に 50% を超えていて ¥Memory¥Pages/sec カウンタが常に 250 未満の場合，仮想メモリに割り当てる RAM 容量を減らすことをお勧めします。

## I/O パフォーマンスのベスト・プラクティス

- I/O を分散するには，固定の仮想ハードディスク（単一ファイルとしてカプセル化された仮想ハードディスク）を使用します。仮想ハードディスクは，パフォーマンス，移植性，スナップショットのサポートをすべて満たします。
- パフォーマンスを最大限に引き出すと同時にプロセッサ・オーバーヘッドを最小限に抑えるには，仮想マシンのデータ・ドライブに，SCSI コントローラに接続した仮想ハードディスクを使用します。
- パフォーマンスを高めるためには，仮想ハードディスク・ファイルを別の物理ディスク上に配置してください。

## ネットワーク・パフォーマンスのベスト・プラクティス

- ネットワーク・パフォーマンスを最適化するには，統合サービスをロードし，合成ネットワーク・アダプタを使用します。これによって，合成ネットワーク・アダプタは，子パーティションと親パーティション間の通信を仮想マシン・バス（VMBus）上の専用チャネルで実行します。
- レガシー・ネットワーク・アダプタを使って PXE（Pre-Execution Environment）で仮想マシンをロードし，合成ネットワーク・アダプタに切り替えます（ただし，ゲスト・オペレーティング・システムに統合サービスのサポート対象バージョンがインストールされていることが前提です）。
- Hyper-V サーバ向けの Large Send Offload と IPv4 TCP Checksum Offload 機能に対応した物理ネットワーク・アダプタを購入します。親パーティションのドライバ設定では，適切なオプションを有効化および設定してください。
- Hyper-V サーバ向けの Large Send Offload と IPv4 TCP Checksum Offload 機能を搭載した物理ネットワーク・アダプタを使用することによって，親パーティションでのパケット処理に必要なプロセッサ・サイクルを低減し，仮想マシンのスループットを向上することができます。親パーティションのドライバ設定では，適切なオプションを有効化および設定してください。
- VLAN Tagging のパフォーマンスを最大限に高めるためには，Large Send Offload と TCP Checksum Offload をサポートする物理ネットワーク・アダプタを採用します。
- 仮想マシンのネットワーク・アダプタの出力キューが頻繁に 2 を超える場合，仮想マシンにネットワーク・アダプタを追加してネットワーク負荷を処理する必要があります。追加したネットワーク・アタプタは同じ仮想ネットワークまたは別の仮想ネットワークにバインドできます。

- 既存の仮想ネットワークが追加トラフィックを処理できるかどうかを確認するには，ホストの ¥Network Interface(*)¥Output Queue Length カウンタを監視してキューの長さをチェックします。ホストのネットワーク・アダプタのキューの長さが 2 を超える場合，物理ネットワーク・アダプタを追加し，外部の仮想ネットワークを新規作成してバインドし，これに仮想マシンを再度割り当てることによってネットワーク負荷を分散します。
- Hyper-V サーバ上にあるすべての物理ネットワーク・アダプタと仮想ネットワーク・アダプタで，共通の最大転送単位（MTU）を設定します。

# 第16章

# VMware の監視

本章では，VMware ベースの仮想マシンの監視に関するベスト・プラクティスを紹介します。

**本章の内容**



## 概要

VMware は，仮想化ソリューションのグローバル・リーダーであり，デスクトップからデータセンターまであらゆる規模のビジネスに対応したソリューションを提供します。

VMware は，デスクトップとサーバを仮想化する無償ソフトウェアから，データセンターや IT インフラストラクチャを最適化するエンタープライズ・クラスの包括的なプラットフォームまで，幅広い仮想化製品を展開しています。VMware の製品は，サーバ統合，インフラストラクチャの最適化，高可用性や災害復旧，ダウンタイムの短縮，ラボ理の自動化など，IT 組織が直面しているさまざまな課題に対応します。

ハイパーバイザである **ESX** と **ESXi** は，データセンターを自動的かつ動的に構築することにより，最高レベルの信頼性とパフォーマンスを発揮する製品です。

VMware ESX と VMware ESXi は，「ベアメタル」のハイパーバイザです。このハイパーバイザは物理サーバのすぐ上の層にインストールされ，複数の仮想マシンにパーティション化できます。また，同時に稼働する基盤サーバの物理リソースを共有することも可能です。仮想マシンは，それぞれがプロセッサ，メモリ，ネットワーク・リソース，ストレージ，BIOS を備えた完全なシステムであり，オペレーティング・システムとアプリケーションを変更せずにそのまま実行できます。

VMware ESX と VMware ESXi は同じ機能とパフォーマンスを備えていますが，アーキテクチャと運用管理機能に相違点があります。VMware ESXi は，VMware が提供する最新のハイパーバイザ・アーキテクチャです。ディスク占有量が非常に小さく汎用 OS に依存しないという特徴があり，これまでにない高いレベルのセキュリティと信頼性を発揮します。ディスク占有量が小さくハードウェア並みの高い信頼性を発揮するという点から，VMware ESXi は業界標準 x86 サーバにプレインストールされています。

本章では，VMWare の仮想化プラットフォームとして最も普及している VMWare ESX サーバを中心に説明します。本章を読むことによって，パフォーマンス・エンジニアは ESX サーバのアーキテクチャの理解を深めることができます。また，パフォーマンス・テストと必要に応じたチューニングの参考としても活用してください。

## アーキテクチャ

VMware ESX Server は，物理ハードウェア上で直接実行されるハイパーバイザであり，システム・リソースの論理プールを作成することにより，完全に分離された環境で稼働する多数の仮想マシンが同じ物理リソースを共有することができます。

ESX Server は，システム・ハードウェアと仮想マシンの間に仮想化層を形成します。この層では，システム・ハードウェアは論理的なコンピューティング・リソースのプールとして構成され，ESX Server は別のオペレーティング・システムやアプリケーションに対してリソースを動的に割り当てることが可能になります。また，仮想マシン上で動作するゲスト・オペレーティング・システムは，仮想リソースを物理リソースと同じ方法で使用できます。

## VMware アーキテクチャの層

仮想環境は，次のようなコンポーネントで構成されます。

- ホスト・マシン
- 仮想化ソフトウェア
- 仮想マシン（1 つまたは複数）
- ゲスト・オペレーティング・システム（1 つまたは複数）


次の節では，各コンポーネントについて詳しく説明します。

### ホスト・マシン

仮想環境では，ホスト・マシンが仮想マシンにリソースを提供します。コアとなるリソースには，CPU，メモリ，NIC，ディスクがあります。ホスト・マシンのリソースが多いほど，ホスト可能な仮想マシンの数も増えます。

ホスト・マシンのリソースの一部はホスト・マシン自身が使用しますが，残りは仮想マシンが使用します。

### 仮想化ソフトウェア

仮想化ソフトウェア層は，仮想マシンがホストのリソースにアクセスできるようにする層です。また，複数の仮想マシン間で物理リソースのスケジューリングを行います。仮想化ソフトウェアは，仮想環境全体の基盤となる不可欠な層です。仮想化ソフトウェアには，仮想マシンの作成，仮想マシンに提供されるリソースの管理，リソース競合が発生した場合のスケジューリング，仮想マシンの管理および設定を行うインタフェースの提供などの機能があります。

VMware は，次の 3 つのバージョンのソフトウェアを提供しています。

- **VMware Workstation** は，ホスト・コンピュータのオペレーティング・システムにインストールできる仮想化ソフトウェア・パッケージです。VMware Workstation には，ホスト・ワークステーションにログオンした状態でないと仮想マシンを実行できないという短所があります。ログオフすると，仮想マシンはシャットダウンします。VMware Workstation は主にローカル・ユーザ・ツールとして使用され，リモート管理機能は付属しません。したがって，運用環境には適していません。
- **VMware GSX Server** は，ホスト・コンピュータのオペレーティング・システム（Linux または Windows）にインストール可能だという点で VMware Workstation と類似していますが，VMware Workstation にはない機能が追加されています。たとえば，仮想マシンのリモート管理とリモート・コンソール・アクセスが可能です。また，複数の仮想マシンをサービスとして実行する設定を，コンソールを使用しないで行うことができます。ただし，ホスト・オペレーティング・システムを通じてホスト・ハードウェアのリソースを取得しなければならないという短所があります。仮想マシンはハードウェアに直接アクセスしないため，スケーラビリティとパフォーマンスに制約が発生します。

- **VMware ESX Server** は，完全なオペレーティング・システムです。ESX Server では仮想マシンのパフォーマンスを最適化する設計が採用されており，ホスト・リソースを共有および使用する方法を管理および設定することが可能です。ESX Server は，GSX や Workstation では実現できなかったレベルの VM パフォーマンスを達成します。また，効率的なリソース割り当て，パフォーマンスの微調整，VM とプロセッサの割合の調整，さらに効率的なリソース共有を可能にします。VMware は，ESX Server の Hardware Compatibility List（HCL）を発表しました。HCL に含まれている ESX サポート・ハードウェアは，期待通りのパフォーマンスを発揮することができます。また，ESX ではホスト・オペレーティング・システムが存在しないので，ホスト・オペレーティング・システムの問題を心配する必要がありません。ESX Server はそれ自身がオペレーティング・システムと仮想化ソフトウェアの両方として機能します。

## 仮想マシン

仮想マシンは，ゲスト・オペレーティング・システムが認識する仮想ハードウェア（仮想ハードウェアと仮想 BIOS の組み合わせ）です。物理ハードウェアをソフトウェアによって仮想化したものであり，インストールされているハードウェアが仮想化されていることをゲスト・オペレーティング・システムが認識することはありません。

ゲスト・オペレーティング・システムが認識する情報には，プロセッサのタイプ，ネットワーク・カードのタイプ，メモリ容量，ディスク容量などがあります。

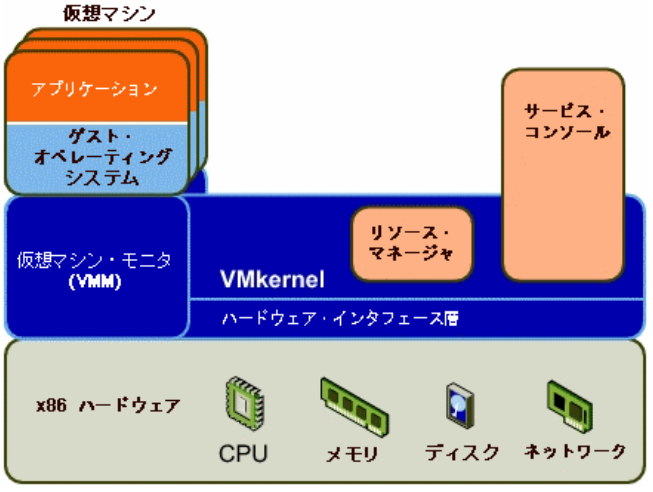
## ゲスト・オペレーティング システム

ゲスト・オペレーティング・システムは，Intel ベースのオペレーティング・システム（Windows，Linux，Novell，DOS）であり，仮想マシン上で動作します。

ゲスト・オペレーティング・システム（「ゲスト・マシン」または単に「ゲスト」）は，VM 上にインストールされているソフトウェアです。オペレーティング・システムのセットアップが完了したら，そのオペレーティング・システム上で動作するアプリケーションをインストールできます。

## 仮想化ソフトウェアの構造

ここでは，ESX Server アーキテクチャ，つまり上記で説明した仮想化ソフトウェア層について説明します。次の図は，ESX の論理的なコンポーネントを示しています。



### VMkernel

VMkernel は VMware が開発した高性能オペレーティング・システムであり，ESX Server ホスト上で直接稼働します。次のように，VMkernel はハードウェア上にあるほとんどの物理リソースを管理します。

- メモリ
- 物理プロセッサ
- ストレージ
- ネットワーク・コントローラ

VMkernel は，仮想化，リソース管理，ハードウェア・インタフェースなど ESX Server の各種コンポーネントを実装します。

### VMkernel リソース・マネージャ

基盤となるサーバの物理リソースをパーティション分割します。電源が入っている仮想マシンに対して，比例配分方式で CPU，メモリ，ディスク・リソースを割り当てます。

ユーザは，仮想マシンごとに，シェア，予約，リミットを指定します。この設定に基づいて，リソース・マネージャは CPU とメモリを各仮想マシンに割り当てます。

### VMkernel ハードウェア・インタフェース層

ハードウェア・インタフェースを使用することにより，ESX Server（および仮想マシン）ユーザはハードウェアの違いを意識することはありません。このインタフェースを通じて，次のようなハードウェア固有のサービスが提供されます。

- デバイス・ドライバ - ハードウェア・デバイスと直接対話します。
- 仮想マシン・ファイル・システム（VMFS）- 分散ファイル・システムです。仮想マシンのディスクやスワップ・ファイルなど，非常に大規模なファイル向けに最適化されます。

### 仮想マシン・モニタ（VMM）

仮想マシン・モニタ（VMM）は，CPU を仮想化します。仮想マシンが起動すると，制御が VMM に渡され，VMM が仮想マシンからの命令を実行します。制御が VMM に移行する時点でシステム・ステートが設定され，VMM はハードウェア上でそのまま稼働可能になります。

### サービス・コンソール

サービス・コンソールは Red Hat Enterprise Linux 3，アップデート 6 をベースとする Linux の限定的なディストリビューションです。

サービス・コンソールは，ESX Server システムの監視および管理用の実行環境を提供します。サービス・コンソールは，物理サーバ・マシンの起動と仮想マシンの管理を行います。サービス・コンソールが起動すると VMkernel がロードされ，

マシンを制御します。サービス・コンソールは，マウス，キーボード，画面，フロッピー・ドライブ，CD-ROM，COM ポート，パラレル・ポートなど，パフォーマンスがあまり重視されないデバイスをサポートします。また，サービス・コンソールは，サポートや管理機能を実装するアプリケーションを実行します。

## 監視ツール

VMware インフラストラクチャでは，パフォーマンス，スケーラビリティ，可用性，信頼性，安定性，管理性を測定するパフォーマンス・カウンタが提供されています。このカウンタを使用して，ESX Server の実行環境で稼働する仮想マシンと物理サーバ・マシンのリソース使用状況を監視できます。

パフォーマンス・カウンタの収集には，さまざまなツールを活用できます。次の項では，ベンダが提供するツールと HP が提供する推奨ツールを簡単に説明します。

### Virtual Center

データセンター管理者向けの監視ツールとして最も広く普及しているツールであり，高い操作性を誇ります。VMware サーバと仮想マシンのグループを 1 つの管理コンソールから監視および制御できます。CPU，メモリ，ネットワーク，ディスクなどのリソースの使用状況に関するデータを集計し，履歴グラフで表示します。

Virtual Center をインストールすると，管理および監視の対象となるホストに接続します。ホストは VC クライアント内に追加されます。このクライアントは ESX サーバ上にエージェントを「インストール」し，VC クライアントからのコマンド実行を可能にするローカル・サービス・アカウントを作成します。

ホストを追加すると，そのホストのパフォーマンス・データをすぐに参照できます。メトリクスは 20 秒ごとに収集され，対象ホストの［パフォーマンス］タブにある VC クライアントで参照できます。表示したい時間枠（日，週，月，年）を選択し，データを分析します。

## コマンドライン・ツール

### esxtop

各仮想マシン，サービス・コンソール，特定の VMkernel システム・サービスの CPU とメモリのデータをリアルタイムで表示します。また，物理プロセッサごとの CPU 使用率，メモリ使用率，仮想マシンで使用できる物理ディスクとネットワーク・デバイスごとのディスクおよびネットワーク帯域幅も表示します。

**esxtop** コマンドは，ホスト・オペレーティング・システムのローカル・コンソール上で実行するか，ホストへの ssh セッションで実行します。**esxtop** コマンドは対話型のツールであり，メトリックスを数秒毎に更新します。

f キーを押すと，メトリックスがカラムに追加され，オンライン監視が始まります。**esxtop** には，インタラクティブ・オプションに加えて 4 つのコマンドライン引数を指定できます。コマンドライン引数には，次のような機能があります。

- 画面の更新を設定（**d**）
- インタラクティブ・コマンドを停止する「セキュア・モード」の設定（**s**）
- 出力データをログに記録する「バッチ・モード」の設定（**b**）
- 更新回数の指定（**n**）

### vmkusage

ESX Server と関連の仮想マシンを実行する単一の物理ホストについて，リソース使用率の履歴データをグラフで表示します。このグラフでは，使用率のトレンドを視覚的に把握できます。

グラフのタイプには直近，日次，週次，月次があり，コンソール・オペレーティング・システムとホスト上で動作する仮想マシンの両方についてトレンド分析を行います。時間間隔ごとにさまざまな分析データが表示されます。

**vmkusage** パッケージは，ESX のインストール・プロセスでインストールされますが，標準では有効になりません。**vmkusage** を使った情報収集とグラフ生成を開始するには，vmkusagectl install コマンドを実行してください。このコマンドは，cron ジョブで **vmkusage** データ収集プロセスを毎分実行する設定を行います。

このツールを実行すると Web ページとしてグラフが生成され，
**http://<ESXservername>.＜会社名＞.com/vmkusage** でアクセスできます。

## 管理ユーザ・インタフェースのステータス・モニタ

物理マシンと仮想マシンのパフォーマンス統計を概要で表示します。履歴グラフは表示しません。

ステータス・モニタは，過去 5 分間の平均値を表示します。過去 5 分間から 20 秒のサンプルを取得し，平均値が計算されます。

ステータス・モニタを表示するには，ブラウザでサーバに接続し，
https://<hostname>:8333 と入力します。ログイン後，ステータス・モニタのページが表示されます。

## HP SiteScope

HP SiteScope は，VMware インフラストラクチャを監視する包括的な機能を提供します。このユーティリティは，VMware の Virtual Center 機能を使用して，ホスト（物理インフラストラクチャ）とゲスト（論理マシン）の情報を表示します。

---

**注：**仮想マシンで動作するゲスト・オペレーティング・システムからパフォーマンス・ツールを実行することはお勧めしません。このような方法で実行するとそのゲスト・オペレーティング・システムしか認識されないので，物理 CPU が共有されているような状況に対応できず，重要なデータが欠落したり無効な結果しか表示されなくなります。

---

# 最も重要な VMware カウンタ

VMware には，プロセッサ，メモリ，ネットワーク，I/O など各デバイスの使用率を測定するさまざまなタイプのカウンタが提供されています。監視を行う場合には，ホスト・マシンのオペレーティング・システム・カウンタとの関連性を考慮してください。これによって，仮想マシンのパフォーマンスを別の角度から把握することができます。

| | カウンタ | 説明 |
|---|---|---|
| CPU | usage | サンプル間隔での CPU 使用率（パーセンテージ）。 |
| | usagemhz | サンプル間隔での CPU 使用率（MHz）。 |
| | ready | 前回の更新間隔中に, CPU が使用可能になるまでに待機した時間。 |
| | reservedCapacity | ホストのルート・リソース・プールの（直接の）子の予約プロパティの合計。 |
| | wait | 前回の更新間隔中に，ハードウェアまたは VMkernel ロック・スレッドのロックを待機した時間。 |
| | swapwait | メモリのスワップインを待機した時間。 |

| | カウンタ | 説明 |
|---|---|---|
| メモリ | **usage** | 使用可能なメモリ容量（パーセンテージ）。 |
| | **vmmemctl** | 他の VM によって要求されるメモリ容量。 |
| | **active** | VM が実際に要求するメモリ容量。 |
| | **granted** | ホストが VM に割り当てたメモリの容量。 |
| | **consumed** | ゲスト・メモリ用に仮想マシンが消費したホスト・メモリの容量。 |
| | **overhead** | VM の保守と実行のために VMkernel が消費したメモリ容量。 |
| | **swapin** | メモリがディスクからスワップインされる容量。 |
| | **swapout** | 開始以降，CLR が認識しているスレッドの総数。 |
| | **shared** | 共有メモリの平均容量。 |
| I/O | **usage** | ホストまたは仮想マシンのすべてのディスク・インスタンスについて，読み取りおよび書き込みを行ったデータ容量の合計。 |
| | **read** | ディスクから読み込まれたデータの容量。 |
| | **write** | ディスクに書き込まれたデータの容量。 |
| | **numberRead** | 前回のサンプル間隔中に実行された I/O 読み取り操作。 |
| | **numberWrite** | 前回のサンプル間隔中に実行された I/O 書き込み操作の回数。 |

| | カウンタ | 説明 |
|---|---|---|
| ネットワーク | **usage** | ホストまたは仮想マシンのすべての NIC インスタンスについて，送受信されたデータ容量の合計。 |
| | **received** | 受信トラフィックのネットワーク・スループットの平均。 |
| | **transmitted** | 送信トラフィックのネットワーク・スループットの平均。 |
| | **droppedRx** | サンプル間隔中にドロップした受信パケットの数。 |
| | **droppedTx** | サンプル期間中にドロップした送信パケットの数。 |

## CPU カウンタ

本項では，プロセッサの使用率に関する情報を提供するカウンタについて説明します。

## usage

| | |
|---|---|
| **正式名** | usage |
| **カウンタのタイプ** | 期間内の値（一定期間にわたってサンプリング） |
| **説明** | サンプル間隔中の CPU 使用率（パーセンテージ）です。ホストと仮想マシンの両方で監視できます。 |
| **使用方法** | 測定単位はパーセンテージです。精度はパーセンテージの 1/100, つまり 1 = 0.01% に設定されています。このカウンタの値は 0～10000 の範囲です。 |
| **パフォーマンス** | このメトリックスを使用すると, 速度の異なる 2 つのホストを比較できます。<br>CPU 使用率は VM レベルのみで測定可能です。<br>すべての VM のスナップショットを作成または削除するような処理をホスト・システム上で実行すると，このカウンタの値は大きく変動します。 |
| **操作** | なし |
| **しきい値** | 100% は，システム上にあるすべてのプロセッサ・コアが完全に使用されている状態を示します。 |

## usagemhz

| | |
|---|---|
| **正式名** | usagemhz |
| **カウンタのタイプ** | 期間内の値（一定期間にわたってサンプリング） |
| **説明** | サンプル間隔中の CPU 使用率（MHz）です。ホストと仮想マシンの両方で監視できます。 |
| **使用方法** | 測定単位は MHz です。 |
| **パフォーマンス** | このメトリックスを使用すると, 速度の異なる 2 つのホストを比較できます。<br>CPU 使用率（MHz）は，VM ごと，または仮想 CPU のインスタンスごとに測定できます。<br>すべての VM のスナップショットを作成または削除するような処理をホスト・システム上で実行すると，このカウンタの値は大きく変動します。 |
| **操作** | ESX Server と仮想マシンによる物理 CPU の使用率は, HostSystem.summary.quickStats.overallCpuUsage と VirtualMachine.summary.quickStats.overallCpuUsage で確認できます。quickStats にはサンプル間隔がなく，VI サービスが値を取得した時点でのパフォーマンス・カウンタのサンプル値を示します。 |
| **しきい値** | なし |

### ready

| | |
|---|---|
| **正式名** | ready |
| **カウンタのタイプ** | 期間内の値（一定期間にわたってサンプリング） |
| **説明** | 前回の更新間隔中に，CPU が使用可能になるまでに待機した時間です。 |
| **使用方法** | 測定単位はミリ秒です。 |
| **パフォーマンス** | このメトリックスを使用すると，速度の異なる 2 つのホストを比較できます。<br>cpu.ready カウンタは，CPU インスタンスごとの値を報告します。 |
| **操作** | %ready は，100 を超えることがあります。これは，VM が CPU を必要としているにも関わらずアクセスできない状態を示しています。短期間だけ 100 を超える場合には問題ありません。ただし %ready が 100 を超える状態が長く続く場合は，VM に十分な CPU が割り当てられていないことを示します。 |
| **しきい値** | なし |

### reservedCapacity

| | |
|---|---|
| **正式名** | reservedCapacity |
| **カウンタのタイプ** | 期間内の値（一定期間にわたってサンプリング） |
| **説明** | ホストのルート・リソース・プールの（直接の）子の予約プロパティの合計です。 |
| **使用方法** | このメトリックスは，ホスト・ステータスを報告します。 |
| **パフォーマンス** | なし |
| **操作** | 子の合計が親を上回るのは，親に reservationExpandable が設定されている場合のみです。 |
| **しきい値** | なし |

### wait

| | |
|---|---|
| **正式名** | wait |
| **カウンタのタイプ** | 期間内の値（一定期間にわたってサンプリング） |
| **説明** | 前回の更新間隔中に，ハードウェアまたは VMkernel ロック・スレッドのロックを待機した時間です。 |
| **使用方法** | 測定単位はミリ秒です。 |
| **パフォーマンス** | |
| **操作** | このメトリックスは，仮想 CPU のインスタンスごとに測定できます。 |
| **しきい値** | なし |

### swapwait

| | |
|---|---|
| **正式名** | swapwait |
| **カウンタのタイプ** | 期間内の値（一定期間にわたってサンプリング） |
| **説明** | スワップ待機時間とは，VM がメモリのスワップインを待機した時間です。VM は，メモリを待機している間，アクティブではなくなります。 |
| **使用方法** | 測定単位はミリ秒です。 |
| **パフォーマンス** | なし |
| **操作** | このメトリックスは，仮想マシンごとに測定できます。 |
| **しきい値** | なし |

## メモリ・カウンタ

本項では，VMware システムのメモリ管理に関連するカウンタについて説明します。このカウンタは，メモリ消費量やメモリ・プールなどの統計を表示します。

### usage

| | |
|---|---|
| **正式名** | usage |
| **カウンタのタイプ** | 期間内の値（一定期間にわたってサンプリング） |
| **説明** | サンプル間隔中のメモリ使用率（パーセンテージ）です。ホストと仮想マシンの両方で監視できます。 |
| **使用方法** | 測定単位はパーセンテージです。精度はパーセンテージの 1/100, つまり 1 ＝ 0.01% に設定されています。このカウンタの値は 0～10000 の範囲です。 |
| **パフォーマンス** | このメトリックスを使用すると, メモリ容量の異なる 2 つのホストを比較できます。 |
| **操作** | なし |
| **しきい値** | 100% は，システム上にあるすべてのメモリが完全に使用されている状態を示します。 |

### vmmemctl

| | |
|---|---|
| **正式名** | vmmemctl |
| **カウンタのタイプ** | 期間内の値（一定期間にわたってサンプリング） |
| **説明** | バルーン・ドライバが現在要求しているメモリ容量です。ホストが, 必要性の低い VM からメモリを解放し始めたことを示します。 |
| **使用方法** | 測定単位は KB です。 |
| **パフォーマンス** | このメトリックスは, バルーニングで再要求されたメモリ容量を示します。バルーン・ページは 1:1 でマッピングされるので，物理ページではなくマシン・ページが使用されます。<br>ただし，ホストがバルーニングを行う場合には，パフォーマンス低下の指標となるスワップ率（スワップインとスワップアウト）をチェックしてください。 |
| **操作** | なし |
| **しきい値** | なし |

### active

| | |
|---|---|
| **正式名** | active |
| **カウンタのタイプ** | 期間内の値（一定期間にわたってサンプリング） |
| **説明** | 短期間の間に VM が使用したメモリ容量です。 |
| **使用方法** | 測定単位は KB です。 |
| **パフォーマンス** | 現在 VM が必要とするメモリ容量を示す「実際の」値です。また, 使用されないメモリはスワップアウトまたはバルーニングされるので, ゲスト・パフォーマンスに影響することはありません。 |
| **操作** | なし |
| **しきい値** | なし |

### granted

| | |
|---|---|
| **正式名** | granted |
| **カウンタのタイプ** | 期間内の値（一定期間にわたってサンプリング） |
| **説明** | ホストが VM に割り当てたメモリの容量です。 |
| **使用方法** | 測定単位は KB です。 |
| **パフォーマンス** | なし |
| **操作** | なし |
| **しきい値** | なし |

### consumed

| | |
|---|---|
| **正式名** | consumed |
| **カウンタのタイプ** | 期間内の値（一定期間にわたってサンプリング） |
| **説明** | ゲスト・メモリ用に仮想マシンが消費したホスト・メモリの容量です。 |
| **使用方法** | 測定単位は KB です。 |
| **パフォーマンス** | 仮想マシン：ゲスト・メモリ用に仮想マシンが消費したホスト・メモリの容量です。<br>ホスト・マシン：次の式で計算されます。<br>**ホストの総メモリ容量ー空きメモリ容量**<br>これには，サービス・コンソール用に予約されたメモリも含まれます。サービス・コンソール用に予約されたメモリは，すべて使用済みとみなされる点に注意してください。 |
| **操作** | このカウンタは，マシン・ページを示します。 |
| **しきい値** | なし |

### overhead

| | |
|---|---|
| **正式名** | overhead |
| **カウンタのタイプ** | 期間内の値（一定期間にわたってサンプリング） |
| **説明** | VM の保守と実行のために VMkernel が消費したメモリ容量です。 |
| **使用方法** | 測定単位は KB です。 |
| **パフォーマンス** | なし |
| **操作** | なし |
| **しきい値** | なし |

### swapin

| | |
|---|---|
| **正式名** | swapin |
| **カウンタのタイプ** | 期間内の値（一定期間にわたってサンプリング） |
| **説明** | ディスクからスワップインされるメモリの容量です。 |
| **使用方法** | 測定単位は KB です。 |
| **パフォーマンス** | このカウンタの値が大きい場合, メモリ不足が原因でパフォーマンスが低下していることを示します。 |
| **操作** | なし |
| **しきい値** | なし |

### swapout

| | |
|---|---|
| **正式名** | swapout |
| **カウンタのタイプ** | 期間内の値（一定期間にわたってサンプリング） |
| **説明** | ディスクにスワップアウトされるメモリの容量です。 |
| **使用方法** | 測定単位は KB です。 |
| **パフォーマンス** | このカウンタの値が大きい場合, メモリ不足が原因でパフォーマンスが低下していることを示します。 |
| **操作** | なし |
| **しきい値** | なし |

### shared

| | |
|---|---|
| **正式名** | shared |
| **カウンタのタイプ** | 期間内の値（一定期間にわたってサンプリング） |
| **説明** | 共有メモリの平均容量です。 |
| **使用方法** | 測定単位は KB です。 |
| **パフォーマンス** | 共有メモリは，共有が可能なメモリ・プール全体を示します。 |
| **操作** | なし |
| **しきい値** | なし |

## I/O カウンタ

本項では，VMware システムの I/O 管理に関連するカウンタについて説明します。

### usage

| | |
|---|---|
| **正式名** | usage |
| **カウンタのタイプ** | 期間内の値（一定期間にわたってサンプリング） |
| **説明** | ホストまたは仮想マシンのすべてのディスク・インスタンスについて，読み取りおよび書き込みを行ったデータ容量の合計を示します。 |
| **使用方法** | 測定単位は KB です。 |
| **パフォーマンス** | ホストについては，このメトリックスは仮想マシンごとの値を積み重ね棒グラフで表示することができます。 |
| **操作** | なし |
| **しきい値** | なし |

### read

| | |
|---|---|
| **正式名** | read |
| **カウンタのタイプ** | 期間内の値（一定期間にわたってサンプリング） |
| **説明** | ディスク読み取り速度です。パフォーマンス測定間隔中に読み取られたデータの容量を示します。 |
| **使用方法** | 測定単位は KB です。 |
| **パフォーマンス** | このメトリックスは，blocksRead と blockSize を掛けた積です。 |
| **操作** | なし |
| **しきい値** | なし |

### write

| | |
|---|---|
| **正式名** | write |
| **カウンタのタイプ** | 期間内の値（一定期間にわたってサンプリング） |
| **説明** | ディスク書き込み速度です。パフォーマンス測定間隔中に書き込まれたデータの容量を示します。 |
| **使用方法** | 測定単位は KB です。 |
| **パフォーマンス** | このメトリックスは, blocksWritten と blockSize を掛けた積です。 |
| **操作** | なし |
| **しきい値** | なし |

### numberRead

| | |
|---|---|
| **正式名** | numberRead |
| **カウンタのタイプ** | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| **説明** | 前回のサンプル間隔中に実行された I/O 読み取り操作の数です。 |
| **使用方法** | 数値 |
| **パフォーマンス** | この操作は最大 64 KB の可変サイズで行われます。 |
| **操作** | なし |
| **しきい値** | なし |

### numberWrite

| | |
|---|---|
| **正式名** | numberWrite |
| **カウンタのタイプ** | 瞬間値（サンプル間隔中に取得された値の 1 つ） |
| **説明** | 前回のサンプル間隔中に実行された I/O 書き込み操作の数です。 |
| **使用方法** | 数値 |
| **パフォーマンス** | この操作は最大 64 KB の可変サイズで行われます。 |
| **操作** | なし |
| **しきい値** | なし |

## ネットワーク・カウンタ

本項では，VMware システムのネットワーク機能に関連するカウンタについて説明します。

### usage

| | |
|---|---|
| **正式名** | usage |
| **カウンタのタイプ** | 期間内の値（一定期間にわたってサンプリング） |
| **説明** | ホストまたは仮想マシンのすべての NIC インスタンスについて，送受信されたデータ容量の合計です。 |
| **使用方法** | 測定単位は KBps です。 |
| **パフォーマンス** | なし |
| **操作** | なし |
| **しきい値** | なし |

### received

| | |
|---|---|
| **正式名** | received |
| **カウンタのタイプ** | 期間内の値（一定期間にわたってサンプリング） |
| **説明** | 受信トラフィックのネットワーク・スループットの平均です。 |
| **使用方法** | 測定単位は KBps です。 |
| **パフォーマンス** | なし |
| **操作** | NIC ごとの値を測定します。 |
| **しきい値** | なし |

### transmitted

| | |
|---|---|
| **正式名** | transmitted |
| **カウンタのタイプ** | 期間内の値（一定期間にわたってサンプリング） |
| **説明** | 送信トラフィックのネットワーク・スループットの平均です。 |
| **使用方法** | 測定単位は KBps です。 |
| **パフォーマンス** | なし |
| **操作** | NIC ごとの値を測定します。 |
| **しきい値** | なし |

### droppedRx

| | |
|---|---|
| **正式名** | droppedRx |
| **カウンタのタイプ** | 期間内の値（一定期間にわたってサンプリング） |
| **説明** | サンプル間隔中にドロップした受信パケットの数です。 |
| **使用方法** | 数値 |
| **パフォーマンス** | なし |
| **操作** | NIC ごとの値を測定します。 |
| **しきい値** | なし |

### droppedTx

| | |
|---|---|
| **正式名** | droppedTx |
| **カウンタのタイプ** | 期間内の値（一定期間にわたってサンプリング） |
| **説明** | サンプル間隔中にドロップした送信パケットの数です。 |
| **使用方法** | 数値 |
| **パフォーマンス** | なし |
| **操作** | NIC ごとの値を測定します。 |
| **しきい値** | なし |

# 最適化とチューニング

ここでは，パフォーマンスを最適化するためのベスト・プラクティスと推奨構成を紹介します。

## CPU パフォーマンスのベスト・プラクティス

CPU を大量に消費するアプリケーションの場合，CPU の仮想化によるオーバーヘッドがそのまま全体的なパフォーマンス低下へとつながります。

ここでは，CPU パフォーマンスを最適化するためのベスト・プラクティスと推奨構成を紹介します。

- 仮想 CPU の数をできるだけ少なくします。仮想 CPU は，使用されていない状態でも ESX Server のリソースを必要とします。
- シングル・プロセッサ仮想マシンでは UP HAL/カーネルを使用します。マルチプロセッサ仮想マシンでは SMP HAL/カーネルを使用します。
- CPU やメモリを過剰に使用するプログラムをサービス・コンソール内で実行しないようにします。これは，仮想マシンと ESX Server のパフォーマンス低下の原因になります。
- ESX 3.0.1 は，64 ビット版のゲスト・オペレーティング・システムを完全にサポートします。64 ビットのゲストとアプリケーションは，32 ビット版よりも高いパフォーマンスを発揮します。
- ゲスト・オペレーティング・システムのタイマ割り込みの間隔がパフォーマンスに影響を与えることがあります。オペレーティング・システムによってタイマ割り込みは異なります。仮想クロック割り込みが大量に発生すると，パフォーマンスに大きな影響を与える可能性があり，ホスト CPU 消費量が増大します。可能であれば，タイマ割り込みの間隔が長いゲストを選択してください。

## メモリ・パフォーマンスのベスト・プラクティス

ESX Server 仮想マシンでは，仮想マシンでのメモリ・アクセス時間が増大する場合と，ESX Server がコード実行やデータ構造用に必要とするメモリ容量が増大する場合に，メモリ・オーバーヘッドが発生します。メモリ・オーバーヘッドには，サービス・コンソールと VMkernel で発生するシステム全体としての固定のオーバーヘッドと，仮想マシンごとのオーバーヘッドがあります。

ESX Server 3.0 では，サービス・コンソールの一般的なメモリ使用量は 272MB であり，VMkernel による使用量はこれよりも小さくなります。オーバーヘッド・メモリには，仮想マシンのフレーム・バッファと各種仮想化データ構造用に予約されている領域が含まれます。オーバーヘッド・メモリは，仮想 CPU の数，ゲスト・オペレーティング・システム用に設定されたメモリ容量，ゲスト・オペレーティング・システムが 32 ビットまたは 64 ビットのいずれかによって異なります。

ここでは，メモリ・パフォーマンスを最適化するためのベスト・プラクティスと推奨構成を紹介します。

- ESX が必要とする総メモリ容量を上回る物理メモリ容量に，仮想マシンがすべて同時に稼働した場合に必要になる容量を加算したメモリをホストに搭載します。
- 仮想マシンに仮想メモリを割り当てる際には，アプリケーションのワーキングセットを格納できるだけのメモリ容量を確保してください。
- 可能であれば，Linux 仮想マシンではゲスト物理メモリを 896MB 未満にします。Linux では，物理メモリが 896MB を超えると，カーネルのメモリ・マッピングに異なる手法が使用されます。この手法では仮想マシンにオーバーヘッドが発生し，パフォーマンスが若干低下する可能性があります。
- ESX Server でメモリのオーバー・コミットを実行する場合，ESX Server 上で十分なスワップ領域を確保してください。ESX Server は，仮想マシンごとにスワップ・ファイルを作成しますが，このファイルのサイズは，仮想マシンの設定メモリ・サイズと予約サイズの差に等しくなります。このスワップ・ファイルは電源投入時に作成されるので，ファイル作成がパフォーマンスに影響を与えることはありません。このスワップ領域には，仮想マシンの設定メモリ・サイズと予約サイズの差，またはそれ以上の容量が必要です。

- スワッピングを回避できない状況でパフォーマンスを向上するには，仮想マシンのスワップ・ファイルを高速/高帯域幅のストレージ・システム上に格納します。標準設定では，仮想マシンのスワップ・ファイルは，仮想マシンと同じ場所に作成されます。この設定を変更するには，sched.swap.dir オプション（VI クライアントで，［設定の編集］→［オプション］→［詳細］→［設定パラメータ］を選択）でパスを指定してください。

## I/O パフォーマンスのベスト・プラクティス

ストレージ・パフォーマンスの問題は，ストレージ・デバイスが適切に設定されていないことが原因で発生する場合がほとんどで，ESX Server 固有の問題であることはまれです。I/O 操作の遅延の影響を受けやすい作業負荷は多数あるので，ストレージ・デバイスを正しく設定することが非常に重要です。

ここでは，ストレージ・パフォーマンスを最適化するためのベスト・プラクティスと推奨構成を紹介します。

- esxtop（QUED カウンタ）でキュー内のコマンド数をチェックし，VMkernel 内で待機している I/O が存在しないことを確認します。
- ストレージ・アレイのパフォーマンスを最適化するには，使用可能なストレージ・パス（複数の HBA と SP）に I/O 負荷を分散します。
- VMFS，分散ファイル・システム，パーティション上でファイルのオープン/クローズを過剰に行う操作は実行しないようにします。可能であれば，ファイルにアクセスした後, 処理とクローズを繰り返すのではなく, 必要な処理をすべて完了してからクローズします。
- ホストとゲストで VMFS パーティションを合わせることをお勧めします。
- 一般的に，ファイバ・チャネル SAN は NAS や iSCSI よりも高いパフォーマンスを発揮します。ファイバ・チャネル SAN では，ファイバ・チャネル・プロトコルの効果で NAS や iSCSI に比べて輻輳や過剰収容は発生しません。
- 使用率の高い仮想マシンが複数ある場合，同じ VMFS ボリュームに同時アクセスしていないことと，複数の VMFS ボリュームに分散されている点を確認します。多数の仮想マシンが同じ VMFS ボリュームに同時アクセスすると SAN I/O が大量に発生し，ディスク・パフォーマンス低下につながります。
- 過剰なファイル・ロックやメタデータ・ロックを必要とする操作を実行しないようにします。このような操作の例には，動的に拡張する vmdk ファイルやファイルへのアクセス権限の操作などがあります。

- HBA カードのキューの深さの最大値を設定します。
- 仮想マシンの未処理のディスク要求の最大数を，必要に応じて増やします。
- iSCSI/NFS　については，リンクのオーバーサブスクリプションが発生しないように，Ethernet の入力リンクが，関連付けられている出力リンクよりも少なくなるようにします。リンクの容量が上限に近い場合，リンク数を減らしてもオーバーサブスクリプションが発生する可能性があります。
- データ取得システムやトランザクション・ロギング・システムなど，大量のデータをストレージに書き込むアプリケーションやシステムでは，ストレージ・デバイスへの Ethernet リンクを共有しないでください。このようなタイプのアプリケーションは，ストレージ・デバイスに対する接続を複数確立することによって，最大限のパフォーマンスを発揮できます。
- ゲスト・ストレージ・ドライバの I/O サイズは，一般的に 64K に標準設定されます。この場合，64K を超える I/O をアプリケーションが発行すると，64K チャンクに分割されます。パフォーマンスを向上するには，これよりも大きなブロック・サイズを発行できるようにレジストリを変更します。

## ネットワーク・パフォーマンスのベスト・プラクティス

ここでは，ネットワーク・パフォーマンスを最適化するためのベスト・プラクティスと推奨構成を紹介します。

- 1 つの vSwitch と物理ネットワークをつなぐ複数のネットワーク・アダプタで, NIC チームを構成します。NIC チームは，物理ネットワーク・アタプタにトラフィックを分散することでパフォーマンスを向上し，ハードウェア障害やネットワーク障害時にパッシブなフェイルオーバを実行します。
- 32 ビット・ゲスト内でエミュレートされる標準の仮想ネットワーク・アダプタは, VMware の vlance ドライバで構成される AMD PCnet32 デバイス（64 ビット・ゲストは e1000）です。ただし，vmxnet の方が vlance より格段に高いパフォーマンスを発揮するので，こちらをお勧めします。
- 別の vSwitches（別の物理ネットワーク・アタプタ）を使用することにより，サービス・コンソール，VMkernel，仮想マシン間の競合や，大きなネットワーク作業負荷を処理する複数の仮想マシン間の競合を回避できます。
- VMkernel ネットワーク・デバイス・ドライバは，ネットワーク・スイッチの速度と全二重設定をすべて同じにしてください。設定が異なると，帯域幅が極端に低下する問題が発生します。

- 同じ ESX ホスト上にある 2 つの仮想マシンを接続するネットワークを確立する場合は，両方の仮想マシンを同じ仮想スイッチに接続します。接続する仮想スイッチが異なると，トラフィック転送により CPU とネットワークで不要なオーバーヘッドが発生します。
- 同じ ESX Server の仮想マシン間のスループットが低下している場合（ゲスト・ドライバのバッファ・オーバーフロー），この問題の解決方法には，受信バッファの数を増やす方法，転送バッファの数を減らす方法，両方を行う方法があります。
- ネットワーク・パフォーマンスを最大限にまで高めるためには，次の機能ハードウェア機能をサポートするネットワーク・アダプタをお勧めします。
  - Checksum Offload
  - TCP Segmentation Offload（TSO）
  - ハイメモリ DMA の処理機能（64 ビット DMA アドレスの処理）
  - Tx フレーム 1 つあたり複数の Scatter/Gather 要素の処理


Java™
COMPATIBLE