

# パフォーマンス改善のための一般的な推奨事項

注： この資料は、Perforce Software 社が作成したドキュメントを東陽テクニカが日本語に翻訳したものです。

オリジナルは、<http://kb.perforce.com/article/931/general-performance-recommendations> をご参照ください。

この資料に関してご不明な点は、テクニカルサポート [ss\\_support@toyo.co.jp](mailto:ss_support@toyo.co.jp) 宛にお問い合わせください。

PERFORCE サーバ (P4D) のパフォーマンスは、多くの要素から影響を受けます。これらの要素は P4D が実行されるマシンのハードウェアおよびオペレーティング・システム、db.\*ファイルが配置されているファイルシステム、そして運用環境における PERFORCE の使用方法に関連します。これらのうち、ある一つの側面の改善が特に有効な場合もありますが、一般的には、総合的にこれらの側面を改善することが、全体的な P4D のパフォーマンス改善につながります。

最良のパフォーマンスを得るために、1 台のマシンを単一の PERFORCE サーバとして専有することをお勧めします。専用のマシンとすることで、そのマシン上で利用可能なすべてのリソースを、単一の PERFORCE サーバが必要な限り使用できます。オペレーティング・システムのファイルシステム・キャッシュに PERFORCE サーバのメタデータをキャッシュする場合、ファイルシステム・キャッシュを他の PERFORCE サーバや他のアプリケーションと共有するよりも単独の PERFORCE サーバで専有する方が効果的です。

## メモリ

P4D 稼動マシンのメモリ、入出力サブシステム、およびプロセッサはすべて、パフォーマンスに影響を与えます。最新のオペレーティング・システムでは、処理空間に不要なメモリのかなりの部分をファイルシステム・キャッシュとして使用するので、メモリ容量をできるだけ大きくすることが推奨されます。ファイルシステム・キャッシュ内に納まる入出力要求は、納まらない入出力要求に比べて早く完了します。

## ディスク・サブシステム

ファイルシステム・キャッシュを超える大きさの入出力要求については、入出力サブシステムでそれを改善できる可能性があります。db.\*ファイルを含むストレージ・サブシステムは、メモリ・キャッシュを備えているべきです。また、ストレージ・サブシステムのメモリ・キャッシュを最大にすることも推奨されます。さらに、最良のパフォーマンスを得るためには、ライトバック・キャッシュを有効にしておくべきです。そのためには、当然ながらストレージ・サブシステムのメモリがバックアップ電源を備えている必要があります。db.\*ファイルが格納されている論理ドライブへの入出力待ち時間は、物理ドライブ自身の回転待ち時間も含め、最小限にとどめるべきです。入出力待ち時間を最小にするには、ホストとストレージ・サブシステムとが直接接続されている必要がある場合があり、通常は最大の回転速度 (例えば、毎分 15,000 回転) を持つ物理ドライブが必要です。

RAID 1+0 (または RAID 10) は、通常はより良好な RAID 構成であり、db.\*ファイルを配置した論理ドライブにも推奨されます。論理ドライブ内の物理ドライブの数も、P4D のパフォーマンスに影響する場合があります。一般的には、論理ドライブ内の物理ドライブの数が増加するのに応じて、パフォーマンスは向上します。必要とされるディスクの空き容量に対して、全体容量の小さな物理ドライブを使用する方が、より良いパフォーマンスを得られることがあります。また、論理ドライブのストライプ・サイズがパフォーマンスに影響する場合があります。最適のストライプ・サイズは論理ドライブ内の物理ドライブの数に依存する場合があります。

ハードウェアベースの RAID 実装（すなわち、ソフトウェアとして実装された RAID ロジックでない）は通常、パフォーマンス特性に優れています。ソフトウェアベースの RAID 実装は CPU サイクルを必要とする場合があります、そうではなくとも、P4D プロセスが CPU サイクルを必要とする場合があるため、ソフトウェアベースの RAID 実装は避けるべきです。

## ジャーナル・ファイルとバージョン化ファイルの場所

トラブルが発生した場合の修復を可能にするために、ライブ・ジャーナル (journal) は、db.\*ファイルを含む物理デバイスとは異なるデバイスに配置する必要があります。また、ライブ・ジャーナルと db.\*ファイルとを分離させることは、パフォーマンス改善のためにも必要な措置です。db.\*ファイルへの書き込み処理の間に、エントリがライブ・ジャーナルに書き込まれ、レコードが db.\*ファイルに書き込まれます。ライブ・ジャーナルと db.\*ファイルが同じ物理デバイス上にある場合、db.\*ファイルへの入出力スループットが低下します。最良のパフォーマンスを得るために、ライブ・ジャーナルは別のホスト・アダプタに接続された別のストレージ・サブシステム上に配置する必要があります。ライブ・ジャーナルは、シーケンシャル書き込み用に最適化された論理ドライブおよびファイルシステム上に置くべきです。

バージョン化ファイルは、db.\*ファイルおよびライブ・ジャーナルがある論理ドライブとは別の論理ドライブに置くべきです。最良のパフォーマンスを得るために、バージョン化ファイルがある論理ドライブは、別のホスト・アダプタに接続された別のストレージ・サブシステム上に配置する必要があります。バージョン化ファイルは通常、かなり多くのディスク空き容量を必要とし、入出力スループットは db.\*ファイルほどは重要ではないため、バージョン化ファイルが格納されている論理ドライブには RAID 5 などのより経済的な構成を使用できます。

## CPU

P4D 稼動マシンのプロセッサおよびメモリが高速であれば、P4D コマンドをより速く実行できます。コマンドによっては、処理の中で部分的に、他のコマンドをブロックするようにリソースを取得します。そういった部分を可能な限り高速に実行することは、最良のパフォーマンスを得る上で重要です。例えば、ほとんどの P4D コマンドには「計算フェーズ」がありますが、その間にいくつかの db.\*ファイルに対して共有ロックが実行されます。db.\*ファイルに対する共有ロックは、同じ db.\*ファイルに書き込もうとしている他の処理をブロックします。コマンドの計算フェーズに必要なデータが、オペレーティング・システムのファイルシステム・キャッシュの中にキャッシュされる場合、プロセッサおよびメモリ速度のみが計算フェーズを抑制します。したがって、P4D 稼動マシンのプロセッサおよびメモリをできるだけ高速にすることが、最良のパフォーマンスを得るためには重要であるということです。

PERFORCE 環境では一般的に、P4D 稼動マシンにおけるプロセッサ数やプロセッサのコア数を増やすよりも、より高速なプロセッサを用いることで、より良いパフォーマンスが得られます。ただし、プロテクション・テーブルやクライアント・ビューの設定が複雑である場合、CPU の必要条件に影響することがあります。CPU 使用率は、"top" (Linux および Unix) や "perfmon" (Windows) などの OS ユーティリティを使用して監視できます。P4D 稼動マシンの CPU 使用率が高い環境で、既に高速なプロセッサを使用している場合は、プロセッサの速度を維持するためにプロセッサ数を増やすか、プロセッサのコア数を増やす必要があるかもしれません。

プロセッサおよびオペレーティング・システムによっては動的周波数制御をサポートしており、動的にプロセッサ電圧とコア周波数を調整することによって、プロセッサが電力消費量を変えられます。プロセッサへの要求が多くなると電圧とコア周波数が増加し、プロセッサが最高速度に達するまで、P4D のパフォーマンスに影響が及ぶ可能性があります。動的周波数制御による省電力化機能は、モバイル・コンピュータでは役立ちますが、P4D 稼動マシンには推奨されません。

動的周波数制御の二つの例を以下に示します。

Intel SpeedStep : 一部の Xeon プロセッサで利用可能であり、モバイル・コンピュータで一般的に利用可能

AMD PowerNow! : サーバ・レベルのプロセッサを含む、AMD プロセッサのアレイで利用可能

これらの機能は、Linux (一部の SuSE ディストリビューションではデフォルトで有効)、Windows および Mac OS X の各プラットフォームでサポートされています。この機能が P4D 稼動マシンで有効にされている場合、それを無効にすることを推奨します。いくつかの Linux ディストリビューション (SuSE 等) では、"powersaved" サービスを "off" に設定することによって、この機能を無効にすることができます。

また、お使いのコンピュータの現在のプロセッサ速度を診断できる場合があります。Linux では、各コアの現在速度が、OS コマンドの "cat/proc/cpuinfo" の出力にある "cpu MHz" の行に示されます。

## オペレーティング・システム

P4D 稼動マシンのオペレーティング・システムの選択も、パフォーマンスに影響することがあります。32 ビットのオペレーティング・システムは、大容量の物理メモリに対処できない可能性があります。それによってファイルシステム・キャッシュの実質的なサイズが制限される場合があります。各種の 64 ビットのオペレーティング・システムには、それぞれ独自のパフォーマンス特性があり、PERFORCE における特定の作業負荷に対して有効に働きます。一般的に、Linux 2.6 後期の 64 ビット・カーネルを使用する Linux ディストリビューションは、ほとんどの PERFORCE の作業負荷に対して良好なパフォーマンス特性を持っています。

## ファイルシステム

ファイルシステムのパフォーマンスは、オペレーティング・システムのパフォーマンスに大きく関係します。通常、種々のオペレーティング・システムは複数のファイルシステムを提供しており、そのそれぞれが、特定の PERFORCE の作業負荷に有効に働く独自のパフォーマンス特性を持ちます。P4D のパフォーマンスを最良にするためには、高いパフォーマンスのファイルシステムに db.\*ファイルを配置するべきです。一般的に、XFS ファイルシステムは、ほとんどの PERFORCE の作業負荷に対して良好なパフォーマンス特性を示します。XFS ファイルシステムは、Linux 2.6 後期の 64 ビット・カーネルを使用する Linux ディストリビューションをはじめ、複数のオペレーティング・システムで利用可能です。

要求されている内容を予測し、ページをキャッシュから読み込むことにより、様々な入出力サブシステムのコンポーネントにおいて入出力を最適化します。この最適化は、一般的に「先読み機能」として知られています。いくつかの実装では、先読みを調整することができ、それによってパフォーマンスが改善される場合があります。しかし、先読みの調整には多少の技術が必要です。例えば、処理の中心がシーケンシャル読み込みである場合は、先読みサイズを大きくすることでパフォーマンスが改善される可能性があります。処理の中心がランダム読み込みである場合、画一的に先読みサイズを大きくすると、キャッシュされた有効なデータが不必要に破棄される可能性があります。

## PERFORCE の使用方法

PERFORCE の使用方法がパフォーマンスに影響することがあり、利用形態によってはパフォーマンスに直接影響する可能性があります。ディポのファイル名は、いくつかの重要な db.\*ファイル (特に db.rev、db.revhex、db.integed) において中核となるため、ディポ・ファイル名のパスが長くなればなるほど、パフォーマンスは低下します。したがって、不必要に長いパス名を使用

しないようにすることが望ましいと言えます。

また、開発手法もパフォーマンスに直接影響することがあります。完全なブランチを頻繁に作成することが要求される開発手法（例えば、バグ修正ごとにブランチを作成）では、メタデータの量が急速に増加し、db.\*ファイルのB-ツリー内により多くの層が発生します。層の数が増加すると、リーフ・ページをたどるためにより多くのキー比較および入出力要求が必要となるため、パフォーマンスに影響を与えます。また、完全なブランチを作成するという自体に対しても、より多くのメタデータの読み取りと書き込みが必要になります。メタデータの読み取りと書き込みが増加すると、ファイルシステム・キャッシュに影響を与え、他のPERFORCEタスクに悪影響を及ぼす可能性があります。完全なブランチを頻繁に作成するのではなく、各バグ修正に必要なファイルだけをブランチするか、または一つのブランチに複数のバグ修正を導入できるような開発技法を検討した方がよいでしょう。

## PERFORCE のアップグレード

PERFORCE では、リリースのたびに P4D の性能改善に努めています。一般的に、最新の P4D リリースを使用することで、最良のパフォーマンスが得られます。なお、新しい P4D リリースをセットアップする前には、その PERFORCE 環境において有効なデータと作業負荷で、一定規模の受け入れテストを実施することをお勧めします。