

# マルチコアによるネットワークアクセラレーション

マルチコアプロセッサに最適化されたソフトウェアでネットワーク性能を飛躍的に向上

ウインドリバー社 プロダクトラインマネージャ Mark Guinther

## 目次

要約.....	1
ネットワークの普及.....	1
性能曲線の鈍化.....	2
マルチコアパラダイム.....	2
対称型マルチプロセッシング (SMP).....	2
非対称型マルチプロセッシング (AMP).....	3
コントロールプレーンとデータプレーンの分離.....	3
マルチコアネットワークアクセラレーションで メリットのあるシステム.....	3
マルチコアプロセッサ.....	4
OS.....	4
ネットワーク技術.....	4
マルチコアネットワークアクセラレーションによるアプローチ.....	5
ファストパス機能.....	5
データプレーン機能.....	5
フローの固定.....	5
フローの共有.....	6
ベンチマーク.....	6
プロセッサのサポート.....	7
結論.....	7
注記.....	8

## 要約

ネットワーク全般の製品や要素の設計において、マルチコア技術の導入が急速に進んでいます。2～32個のコアを持つプロセッサの使用で得られる多数の利点が、アクセス網、エッジ網、コア網向けの機器に生かされています。製品設計に採用されるマルチコアの一般的な用途は、統合によるコスト削減と、ネットワークアクセラレーション技術による性能向上です。ネットワーク分野では、ワイヤレスを搭載したインテリジェントなデバイスから生じる、急増するデータトラフィックを処理するアプリケーション性能の向上に注目が集まっています。対称型マルチプロセッシング (SMP) だけでは、マルチコアネットワークのフルパワーは発揮されません。OS、ネットワークプロトコル、マルチコアプロセッサの専門技術を組み合わせ、さらに高度なアプローチが必要です。

今日のネットワークインフラストラクチャでは、アプリケーションの性能と大量のデータパケットを処理する能力が、エンドユーザの全体的な体験に直接影響を及ぼす可能性があります。機器メーカーにとって、アプリケーションの性能最適化の適切な組み合わせを、いかに迅速に発見できるかが成否を分けかねません。

本資料では、ネットワークアプリケーションの持つネットワーク性能の飛躍的な向上を図るうえで、マルチコアソフトウェアが果たす役割について解説します。現在入手可能な多種多様なマルチコアプロセッサには、技術的に高度な性能機能が多数搭載されていますが、そのメリットを最大限に利用するには、インテリジェントなランタイムソフトウェアによるチューニングが必要です。

## ネットワークの普及

IPネットワークは、インターネット上の大型ルータから携帯機器まで、プロセッサを搭載したほとんどすべてのシステムの標準になりました。現在こういったネットワークでは、電子メール、メッセージ、Webページコンテンツから、ビデオ、音声、位置情報まで、あらゆるものが運ばれています。消費者は多様な情報源にアクセスするメリットを認識しており、それがサービスプロバイダや機器メーカーの収益につながっています。マルチメディアコンテンツへの需要は、昔からずっとあり、ネットワーク技術が追いつくのを待っていたにすぎないという見方もあるでしょう。音声、ビデオ、データサービスの融合には、低遅延を要件とするより高性能のインターネットワーキングデバイスが求められます。現在のホームゲートウェイは、混在するインターネットアクセス、VoIP、ストリーミングビデオを処理することが必要です。同様に、アップルのiPhoneやグーグルのAndroidといった携帯機器は、さらに小さな筐体の中に音声、データ、音楽、インターネットアクセス、ビデオを融合させています。

サービスプロバイダには、マルチメディアコンテンツを供給できる帯域幅へのアクセスを提供することに加えて、収益性の確保に必要となる加入者数に合わせた十分な容量を用意するという課題が上乗せされています。その結果求められるのは、高帯域幅の転送ネットワークと、ネットワークエッジでの高性能パケット処理能力です。しかし、この課題に対処するには、大容量回線だけでは不十分です。サービスプロバイダはコストを抑えながら、処理能力を拡大する、独創的かつ効率的な方法を見つけなければなりません。世界経済の現状では、利益幅は圧縮され、開発サイクルは短くなる一方です。前世代のシステムに単純にブレードを増設するというやり方では、このような時間、コスト、機能面の制約に対処するのは困難です。問題の中心にあるのは、パケット処理装置です。パケット処理の効率化は、ネットワークサービスの向上、顧客体験の向上、他社への乗り換えの減少、利益の増大を図るための前提条件です。高性能かつ高密度で、スケーラブルなネットワーク機器を生み出す、新たなアプローチが求められます。

まさにその時、マルチコアプロセッサの出現により、ネットワークパケット処理に新たなパラダイムが登場しました。多くの半導体メーカーから、2、4、8、16個のコアを内蔵するプロセッサが発表されています。これらのチップは、研究用のコンピュータサーバやエンジニアリングワークステーションでの使用にとどまるものではなくりました。これらのマルチコアチップは、次世代のネットワーク機器や公衆網および専用線の進化に重要な役割を果たしていきます。マルチコア技術によるネットワークの効率化を理解す

るには、まず、マルチコアプロセッサの一般的な価値提案を知る必要があります。

### 性能曲線の鈍化

過去数十年間、プロセッサの性能は2年ごとに倍増することが実証されてきました。ムーアの法則として知られる概念です。しかし近年では、熱や電力面の制約から使用可能なトランジスタの増加分をプロセッサが利用しきれなくなってきたため、プロセッサ速度の曲線は横ばいに近づきつつあります。

しかし、マルチコアプロセッシングが新たなパラダイムを提案します。複数のコアを並列に使用することで、電力、熱、コストの特性を同等性能のシングルチッププロセッサより低く保ちながら、高性能への要求に応える処理能力の向上を実現できます。

大手半導体メーカーの大半がマルチコアプロセッサを提供しています。マルチコアプロセッサには、複数のプロセッシングコアが1つのチップ上に組み込まれています。仮想コア(またはスレッド)により、タスク間の非常に高速なコンテキストスイッチを容易化することで、コアリソースをさらに細分化できます。

マルチプロセッサチップの性能は、クロック速度とコアの数によってさまざまです。ノートパソコンのようなデバイスで優れた性能を発揮しているデュアルコアチップもあれば、16~32個のコアを内蔵するチップもすでに登場しています。これらのチップの多くには、従来はネットワークプロトコルソフトウェアによって導入されてきた、遅延を軽減するためのネットワーク処理機能が統合されています。

ネットワークのパケット処理を専用コアにオフロードすることで、ギガビットイーサネットのワイヤスピードのスループットを、複数のポートで実現できます。このことは、ネットワークのエッジ、アグリゲーション、3G/4G機器のコストと性能面に重大な影響を及ぼします。

マルチコアプロセッサをネットワーキングに利用するメリットは、パケットのスループットだけではなく、一部のコアをパケット処理専用にすることで、このようなタスクをメインOSからオフロードでき、貴重なシステムサイクルを他のシステム機能で使用可能になります。マルチコアパケット処理が最初に実装されたのは、マルチギガビットのネットワークインフラストラクチャ装置ですが、低コストのマルチコアプロセッサの採用により、エッジ、アクセス、さらには宅内機器でのネットワークのオフロード化が進展しようとしています。

### マルチコアパラダイム

図1に、従来のシングルCPUシステムを示します。このシステムでは、CPUはすべてのタスクとイベントを処理しなければなりません。システムの稼働に必要な隠れた活動が数多く含まれていることもあります。各種のシステムタスクやイベント間で処理サイクルが分割されるほか、さまざまな活動のコンテキストの切り替えに時間がかかるため、全体的なシステム性能が低下するおそれがあります。マルチプロセッサシステムなら、1つのコアで処理を続行しながら、別のコアではI/O待ち、割り込み処理、システムリソース管理をブロックするようにできるため、このようなオーバーヘッドを削減することが可能です。

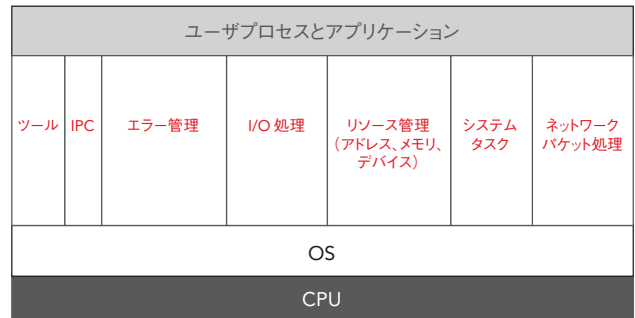


図1: 従来のシングルプロセッサシステム

### 対称型マルチプロセッシング (SMP)

マルチコアシステムは、さまざまな構成で提供されます。コアが2または4個のシステムの一般的な構成は、対称型マルチプロセッシング(SMP)<sup>\*1</sup>です。SMPシステムでは、複数のコアは同じタスクを処理できる同等のリソースとして扱われます<sup>\*2</sup>。図2に、2コアのSMPシステムを示します。すべてのシステムタスクは、どのCPUでも実行できます。処理リソースの割り当てとスケジューリングは、OSが管理します。

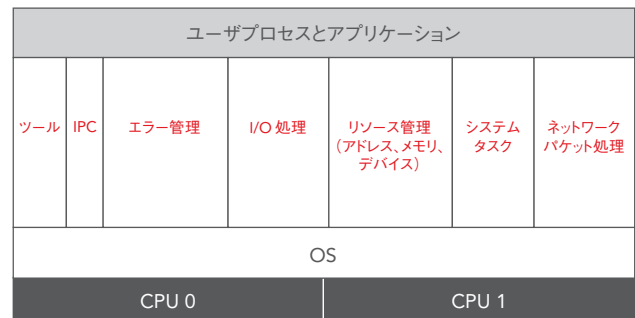


図2: プロセッサ2個の対称型マルチプロセッシング (SMP)

SMPのメリットは直感的にわかるかもしれませんが。作業を複数の作業員で分ければ、短時間で仕事を完了できます。「人手が多ければ仕事は楽になる」ということわざで表現できます。しかし、複数の作業員のメリットは、かならずしも人数に正比例するわけではありません。たとえば、2コアシステムの処理効率はシングルコアシステムの2倍ではなく、4コアシステムは4倍性能アップするわけではありません<sup>\*3</sup>。その理由は、タスクによっては、並列ではなく、順を追って実行される必要があるからです。フレデリック・P・ブルックス Jr.は、著書『人月の神話—狼人間を撃つ銀の弾はない』(ピアソンエデュケーション、2002年)(原書『The Mythical Man-Month』(Addison-Wesley Professional, 1995)の中で、「9人の女性が1か月で赤ん坊を産むことはできない」と表現しています。この説明に該当するソ

ソフトウェアタスクは、常にある割合で存在し、マルチコアプロセッシングの性能が正比例で向上しない原因になっています。

シングルプロセッサ環境向けに書かれたコード（現在の世界中のソフトウェアの大半）は通常、マルチプロセッサ環境で動作するように修正する必要があります。大部分のコードは、データにアクセスするのは1つのプロセッサだけという想定で書かれています。そのため、競合状態や複数のプロセッサが「衝突する」のを防ぐコーディング規則や対処策が実装されていません。コードのクリティカルセクションを特定して、ロックすることは可能ですが、それではコードをマルチプロセッサ対応化できても、マルチプロセッサに最適化されているとはかぎりません。マルチプロセッサ対応ソフトウェアの大半は、ロック処理や同期処理が余分に必要になることが原因で、実際にはマルチコアプロセッサでの動作のほうが遅くなります。並列実行専用設計されたコードだけが、マルチコアプロセッシングによる有意義な性能向上を実現できます。システムのソフトウェアが「並列化」されている程度によって、SMPでどれだけの効率化が可能になるかが決まります。

### 非対称型マルチプロセッシング (AMP)

もうひとつのマルチプロセッシングモデルは、非対称型マルチプロセッシング (AMP) と呼ばれます。AMPでは、プロセッシングコアを同等なリソースとして扱いません。処理を行う特定のコアが決められているタスクがあります。そうすることで、割り込み、I/O、メモリアクセスを分離したり、反復処理のコンテキストスイッチを最小化するというメリットが得られます。AMPでも、SMP同様に並列化は重要です。ただしAMPシステムでは、一部のコアをOSの管理タスクのオーバーヘッドから解放することで、メリットを得ることができます。

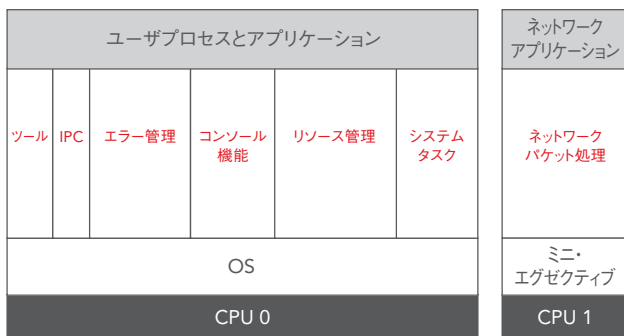


図3: ネットワークパケット処理を行うAMP

独立して実行できるタスク（または大部分が独立しているタスク）は、専用プロセッシングコアに適しています。ネットワークパケット処理はこの条件にぴったりです。図3に、どのようにネットワークパケット処理を他のシステムプロセスから分離し、その結果、専用に処理できるかを示します。オフロードされたソフトウェア（この場合はネットワークパケット処理）が十分に特化または高度化されている場合は、フル機能のOSは必要ありません。図3の小さなランタイムエグゼクティブには、最小の機能セット（メモリ管理、I/O、ハードウェア管理、タイマ）だけが実装されています。システムタスクやコンテキストスイッチに費やす時間を減らすことで、分離されたこれらのコアはより多くの時間をパケット処理に使えます。

### コントロールプレーンとデータプレーンの分離

多くの場合、システムはトップダウンで設計されます。最初にコントロールプレーンとデータプレーンを分離して、次にこれらの機能領域に適した処理を割り当てていきます。たとえば図4のシステムは、CPU 0～CPU 3をSMPモードで構成し、LinuxやVxWorksのようなリアルタイムOS (RTOS) が動作するコントロールプレーンに割り当てています。残りのコア (4～n) プロセッサはAMPモードで構成され、データプレーン機能を処理するミニ・エグゼクティブが動作します。

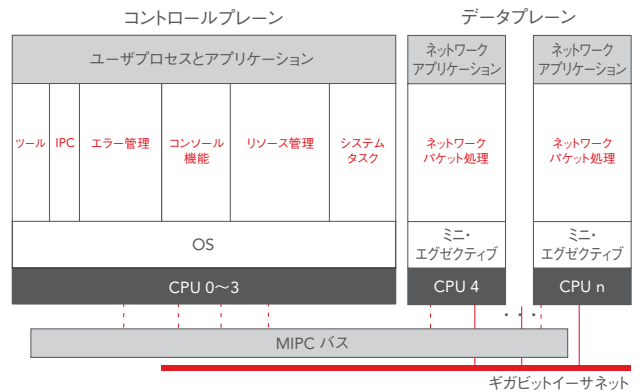


図4: コントロールプレーン/データプレーンAMP構成

効率的な処理には、マルチコア環境に特化したネットワークパケット処理コードを書く必要があります。従来のTCP/IPスタックは、マルチプロセッシングの効率化に必要な並列構造を本質的に欠くため、適切に機能しません。

### マルチコアネットワークアクセラレーションでメリットのあるシステム

マルチコアによるネットワークアクセラレーションは、パケット処理の新しいパラダイムです。従来のTCP/IPスタックは、複数のCPUで処理されるように設計されていませんが、SMPシステムでの処理効率を高めるように一部修正されてきました。マルチコアネットワークアクセラレーションを使用するシステムのスループットは、シングルプロセッサやSMPシステムを採用するシステムと比べて、何倍も向上できます。しかし、このデザインからメリットを得られるのは、どのようなシステムでしょうか。

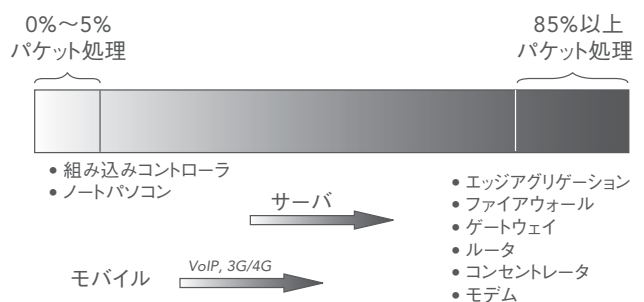


図5: システムCPUサイクルの割り当て

システム内のCPUによって実行されるすべての命令を考えてみましょう。そのうち、パケット処理に費やされるサイクルの比率はどれくらいでしょう。また、一方の端はパケット処理がゼロで、反対側の端はパケット処理が100%という連続体を想像してみてください。

パケット処理が第1の目標であるシステムでは、マルチコアネットワークアクセラレーションのメリットがすぐに得られます。たとえ

ば、ワイヤレス、キャリアイーサネット、光ネットワーク用のネットワークインフラストラクチャ機器などです。サーバの場合も、加入者の高帯域幅への要求を処理する必要があるため、メリットがあります。いずれは携帯機器やモバイル機器でも、3G/4Gアプリケーションの帯域幅の消費が増大し、音声やデータ通信のパケット化が開始されるにつれて、高速パケット処理への依存度が高まる見込みです。

現在マルチコアネットワークアクセラレーションを先行採用しているのは、ネットワーク機器ベンダや通信機器メーカーです。しかし、このようなマルチコアパラダイムは外に向かって拡大し、サーバへと広がり、最終的には、大量のパケット処理が必要なすべての機器に及ぶことが予想されます。

図6に示すように、効率的なAMPシステムの設計には、3つの重要分野での専門技術が求められます。

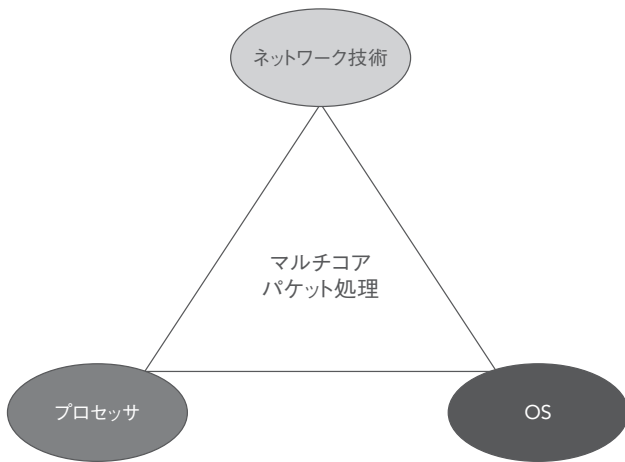


図6: マルチコアパケット処理の要素

### マルチコアプロセッサ

現在のマルチプロセッサチップは、複数のプロセッサが1つのダイの上にあるというレベルをはるかに超えています。大手プロセッサメーカーは、マルチコアチップに数多くの便利な機能を搭載しており、適切に使用すればSMP/AMPシステムを効率よく運用できます。キャッシング、キャッシング、プロセッサ間通信、割り込み管理、メモリ管理などのプロセッサ機能は、ソフトウェアによる仕組みよりも格段に効率よく実行できます。ただし、システムソフトウェアによって適切に利用されることが不可欠です。

#### プロセッサスレッドとは

複数のプロセッシングコアに加えて、シングルコアに複数のスレッドを実装する半導体メーカーもあります。スレッド化されたプロセッサでは、半導体の中に同時に保持される複数のコンテキストが格納されます。そのため、処理中の「スレッド」がブロックした時に非常に高速なコンテキストスイッチが可能になります。これにより、複数のプロセスを実行しているコア上でシステム性能を向上させることができます。プロセス間のコンテキストスイッチは重要です。また、OSに複数スレディングのサポートが含まれています。

マルチスレディングにより、コントロールプレーンの性能を向上できます。コントロールプレーンでは、複数のプロセスがCPUサイクルを取り合っているからです。データプレーンの設計では、Run-to-Completion (完了するまで実

行)モデルが、パケット処理にとって最も効率のよいデザインであるケースが多々あります。各コアの処理の程度によって、マルチスレディングでどれぐらい性能が向上するかが決まります。

スレディングの効率化には、同じ条件でブロックすることの少ない、同一コアからスレッドを割り当てるように配慮する必要があります。1つのスレッドがブロックすると、もう一方のスレッドが実行できる状況になければなりません。1965年にオランダのコンピュータ科学者エドガー・ダイクストラ (Edsger Dijkstra) は、5台のコンピュータが共有する5台のテーブ装置へのアクセスを競合するという問題を提示しました。このシナリオはさまざまに形を変えて語られてきましたが、おそらく最も有名なのは「食事する哲学者の問題」でしょう。5人の哲学者が丸テーブルに着席しており、各人の間に1本のフォークが置かれています。しかし、テーブルの上のスパゲッティを食べるには、左右のフォークが必要になります。この古典的な例え話は、マルチスレッドプログラミングでデッドロックを回避する重要性をわかりやすく説明する際によく使用されますが、複数のプロセスが同じリソースにアクセスするときに、同期(ロック)処理が必要になる仕組みも示しています。このような同期は、マルチスレッド化されたパケット処理を効率化するには、最小に抑える必要があります。

もうひとつの重要な注意点はキャッシングです。スレッドを別個のCPUとして扱うことには、共有されるCPUキャッシュへの影響が考慮されていません。ネットワーク処理コアが複数のメモリ領域にアクセスする場合、コンテキストスイッチのたびにCPUキャッシュをリフレッシュしなければならず、処理性能が制限されるおそれがあります。

### OS

OSは、マルチコアプロセッシング環境で重要な役割を果たします。複数コアのブート、プロセッサ間通信、システム認識、ヒットレスアップグレード、電力管理、コンテキストスイッチを行うための効率的な構造を提供する必要があります。AMPシステムでは、複数のOSの協調を図るために、効率のよいメッセージパッシングの仕組みが不可欠です。

見落としとしてはならないのは、OSのデバッグツールです。性能のチューニングには、複数コアの同時監視、メッセージのアグリゲートやフィルタ、コア間でのブレークポイントの割り当てを行える解析ツールが必要です。マルチコアシステムのデバッグは、複雑な作業になりかねません。適切なツールを使わずに行うのは、無謀な試みに終わるおそれがあります。

### ネットワーク技術

パケット処理システムの効率化にネットワーク分野の専門技術が必要なのは、言うまでもないでしょう。しかし、マルチコアによるネットワークには、それほど明白ではありませんが、別次元の複雑さが関わってきます。ユニプロセッサ用のネットワークスタックをマルチプロセッサ対応にするだけで、うまくいくケースもあるでしょうが、効率化は図れません。マルチコア環境に特化したネットワークスタックの設計により、効率のよい共通データ構造の管理、コネクション管理、コントロールプレーンとデータプレーンの動作の同期が保証されます。第2層、3層、4層と上位層のプロトコルの深い理解は、標準に準拠した、マルチコア環境向けネットワークスタックの構築に不可欠です。

## マルチコアネットワークアクセラレーションによるアプローチ

ウインドリバーは、長年にわたって高性能組み込みソフトウェア分野のリーダーであるとともに、マルチコア最適化ソフトウェアの先陣を切っています。ネットワークパケット処理は、ウインドリバーのマルチコアネットワークアクセラレーション・ソフトウェアによるアプローチを採用することで、大幅に加速化できます。しかし、シングルプロセッサシステムからマルチコアシステムへの転換は、簡単な作業ではありません。

ここでは、マルチコアシステムの設計で考慮すべき新たな可能性、課題、トレードオフをいくつか紹介します。

### ファストパス機能

データプレーンは、マルチコアネットワークアクセラレーションが行われるところです。フローは最初に確立されるときに、コントロールプレーンを経由して、これ以降の通信で使用される転送情報とその他の接続特性をセットアップします。目標は、コントロールプレーンからの介入を必要とせずに、データプレーンでヘッダの検査や修正、暗号化/復号化、アドレス変換、タグ操作などの反復タスクの処理を可能なかぎり行うことです。図7に、ウインドリバーが提供している16コア構成のデータプレーン/コントロールプレーンAMPモデルの例を示します。ネットワークアクセラレーション・エンジンには、小さくて効率的なエグゼクティブ上で動作するネットワークプロトコルソフトウェアが含まれています。

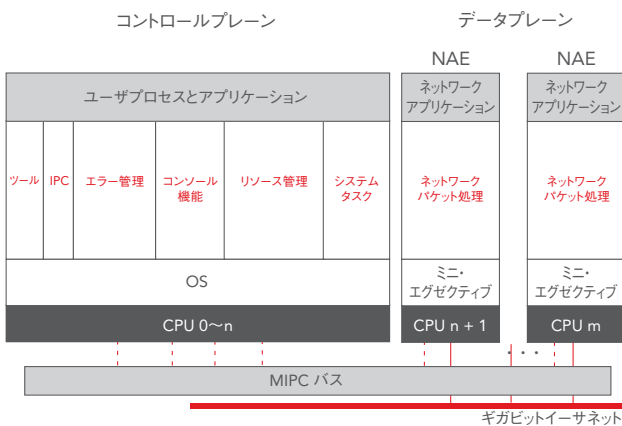


図7: マルチコアネットワークアクセラレーションのフロー

### データプレーン機能

マルチコアプロセッサのメリットをフルに引き出すには、タスクを並列に実行できる必要があります。異なるストリームのパケットをコアが同時に処理できるということです。IPのようなコネクションレス型プロトコルの場合、FIB (forwarding information base) の共有により、これを実現できます。TCPやSCTPなどのコネクション指向のプロトコルの場合は、状態情報<sup>4</sup>がコネクションごとに維持される必要があります。特定のフローのすべてのパケットが同一コアで処理される場合、そのコアはパケットを順番に処理すればよいだけでなく、特定コネクションの状態情報の更新時に他のコアとの競合状態にさらされることもなくなります。このことは、パケット処理リソースの分散に関する設計上の重要な注意点になります。あるフローのパケットがすべて同一コアで処理される場合、このフローはそのコアに「固定」されます。

## フローの固定

一般的に、複数のプロセッサによって更新可能なデータは、競合状態を防ぐためにアクセス中はロックされる必要があります。ロックされている間は同時処理のメリットが実質的に消えるため、ファストパスでのロックは20%以上性能が低下するおそれがあります。

この問題に対処するために、マルチコアネットワークアクセラレーション設計によっては、フロー固定の一種が実装されています。特定の「フロー」(宛先アドレス、5タプル、または他のあらかじめ設定された基準によって識別)が、常に同じコアに送られて処理されることを、ハードウェアによって保証するという仕組みです。これにより、各コアは自分のコネクションについてのみ知っていればよいので、コア間で転送情報とコネクション情報を共有する必要がなくなります。

フローの固定は、確かにコードでのロックの必要性を軽減できますが、新たにロードバランシングの問題を招く可能性もあります。フローのトラフィック負荷が均一でなかったり、フローの存在する時間が均一でないと、システムトラフィック全体からすると不均衡量が徐々に1つのコアに集中し、他のコアはアイドル状態になることがあります。最悪の場合、それによって全体的なシステムスループットが、1個のネットワークアクセラレーションコアの性能にまで低下しかねません。マルチコアネットワークアクセラレーションの性能面のメリットが無効になってしまいます。

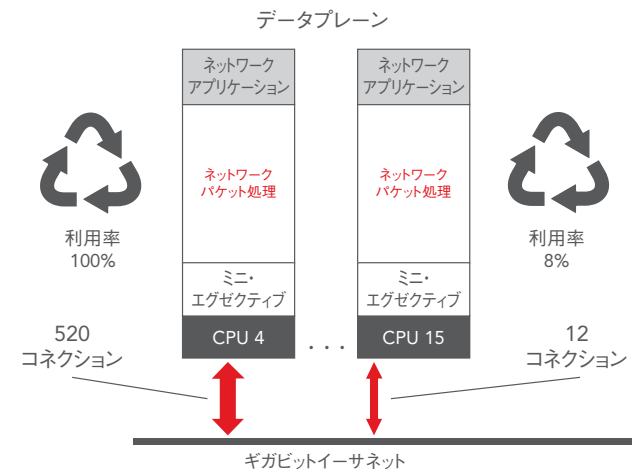


図8: フロー固定による不均衡な作業分散

不均衡な作業負荷を防ぐ、または最小に抑えるために、何らかの措置を講じることが可能です。たとえば、「ディスパッチャ」を作成して、新しいフローを割り当てるコアを決める前に、各コアの相対的な作業負荷を測定できます。ただし、コネクションのコンテンツや寿命は、常に均一ではありません。そのため、将来システムにかかる負荷を予測するための決定的な指標を見つけるのは困難です。システムのバランスが極度に悪化したときは、システムリセットが唯一の対処策という場合もあります。

## フローの共有

フロー固定に代わるアプローチが、フロー共有です。フロー共有モデルでは、どのコアでもいずれのフローの packets も処理することができます。packet は特定のコアを待たずに、空いている次のコアに処理されるため、負荷分担が不均衡になるシナリオを回避できます。しかしフロー共有では、特定の接続またはフローにアクセスする可能性のあるすべてのプロセッサ間で、情報を共有しなければなりません。IP や UDP のようなコネクションレス型プロトコルでは、より簡単に行えます。FIB や SA (セキュリティアソシエーション) の更新頻度が低いため、これらのデータをアトミック操作で保護しても、性能への影響は無視できるほど小さいためです。

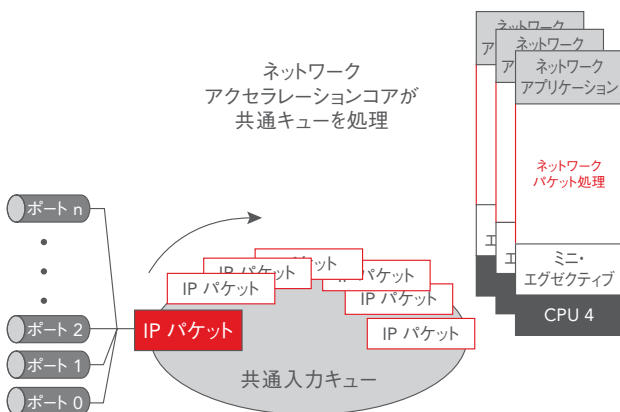


図9: 共通入力キューを使ったフロー共有

TCP や SCTP などのコネクション指向プロトコルでは、状態情報を packet ごとに更新する必要があります。この場合、マルチプロセッシングが場合によっては1秒当たり数100万回中断されるため、粗粒度ロックが非現実的なのは明らかです。

マルチコアプロセッサによっては、1つのフローの packet が1度に1つのコアでしか処理できないことを、ハードウェアによるメカニズムで保証します。この方法は、ソフトウェアロックを回避しながら、アトミック操作をサポートする効率的な手法になり得ます。

すべての入力ポートを、すべてのコアによって処理可能な共通キューに送ることができるプロセッサもあります。空港の舗道で次の乗客を待つタクシーのように、コアは1列に並んで次の packet を待つだけなので、フロー共有には有効な仕組みです。

しかし、すべてのマルチコアチップがこの概念をサポートしているわけではありません。プロセッサによっては、各ポートは packet を一意のキューに受け取り、ネットワークアクセラレーションエンジンがポーリングすることが必要です。ポートの数が多い場合、すべてのネットワークアクセラレーションエンジンが、すべてのキューをポーリングして packet を探すのは実用的ではないかもしれません。このような場合は、1つのコアに負荷がかかりすぎないように、コアのグループを対応するポートのグループに割り当てるのが可能です。負荷が不均等になる可能性は残りますが、新しい packet の処理に複数のコアが使用可能なため、不均衡は軽減されます。

高速化が可能なのは、ファストパスのネットワーク機能だけです。「なぜすべてのネットワーク機能をファストパスに置かないのか」という疑問が生じます。答えは単純で、ファストパスが高速ではなくなってしまうからです。ファストパスを使用する機能を増やせば、キャッシング、メモリ管理、バスの競合、プロセッサ間通信がより大きな要因になります。ここは、システム設計の意思決定で差別化を図れるポイントです。機能のチェックリストは必要ですが、特定の packet 処理設計が要件を満たすかどうかを判断するには不

十分です。

さまざまな機能をファストパスで実行できます。そういった機能がすべて必要なシステムはわずかですが、多くのシステムには以下の1つまたはそれ以上の機能が必要です。

- IPv4/IPv6転送
- IPsec暗号化/復号化
- NAT (Network address translation)
- VLANタグ付け/スイッチング
- ACL (アクセス制御リスト)
- packet の分類/コンテンツフィルタリング
- MPLSエンド/トランジット機能
- トンネリング
- レイヤ4処理 (UDP, TCP, SCTP)

高度に冗長化されたタイムクリティカルな機能は、ファストパスで実行すべきなのは明白です。しかし、要求頻度が低い機能があり、高速化されていないバスを使用しても許容される場合もあることに留意してください。特に、ファストパスに追加することは、他のファストパス機能の性能に影響する点に注意が必要です。

## ベンチマーク

ベンチマークは、自動車の燃費評価のようなものです。リッター当たりの走行距離はさまざまです。ベンチマークは、多数の要因 (CPUクロック速度、packetサイズ、コネクション数、テスト対象のプロトコル、使用されるホストOS、コア数、キャッシュサイズなど) に左右される可能性がある反面、設計の動作について重要な情報を明らかにできます。

まず、わかりきったことですが、かならず自分のシステムと関係のあるプロセッサとプロトコルを使用して、ベンチマークを行うようにします。IPv6 と IPsec AES 256 暗号化方式が重要な場合は、IPv4 の転送性能が実際に得られる性能の代用になるとは考えないでください。キャッシュがより小さければ、キャッシュラインのフェッチが増えるため、アーキテクチャが同じでもキャッシュがより大きい場合よりもスループットが低下することがあります。プロセッサのクロック速度も、ポート数やメモリ帯域幅などの他の制約をシステムが受けていないかぎり、性能に影響する可能性があります。

重要なのに見落とされがちな要因が、ベンチマークに使用されるコントロールプレーンのOSです。マルチコアチップツールキットを packet でフラディングさせることで、プロセッサの性能を適切に示すことができますが、LinuxのようなOSがキャッシュとやり取りする際の性能が反映されないケースもあります。マルチOSシステムでボトルネックを回避するには、OS内部 (スケジューリング、プロセッサ間通信、メモリ管理など) の専門技術が必要になります。

クリアテキストの packet 処理の場合、一般的なベンチマークは、正当な最小 packet / フレームサイズが最もオーバーヘッドを生じるため、これを使用して行われるのが通常です。イーサネットの場合は64バイトです。20バイトのプリアンブルと packet 間ギャップを、64バイトのフレームに追加する (仕様の要件に準拠) 場合、ワイヤ上の最大可能ビット数は1ギガビット/秒です。これは1秒当たり144万 packet に相当します。packetサイズを大きくすると (128、256、512バイト)、同数のビットを転送するのに必要な packet 数が減少します。

イーサネット ペイロード (バイト)	1フレーム当たり総 バイト数 (フレーム サイズ +20バイト)	1秒当たりフレーム数 = 10億 ビット/秒 (1フレーム当たり総 バイト数 × 8ビット/バイト)
64	84	1,488,095
128	148	844,595
256	276	452,899
512	532	234,962
1,024	1,044	119,732
1,280	1,300	96,154
1,518	1,538	81,274

図10: ギガビットイーサネットの理論上の最大スループット

IPsecで使用されるパケットのような暗号化されたパケットの場合、暗号化/復号化方式が原因で、より大きなパケットサイズのスループットが小さいパケットより低下するケースがあります。注目すべきポイントは、「標準的なベンチマーク」ではシステムの正確な動作が示されにくいことです。そのため、予想されるトラフィックのパターンと要件について慎重に考慮することが必要です。最高の指標は常に、具体的な環境で行うベンチマークです。

もうひとつ重要な注意点は、ソリューションのスケラビリティです。ベンチマークはコア数に比例しているか、それともあるコア数を超えると横ばいになるかです。後者の場合は、プロセッサ以外の何らかの限界に設計が達しているということです。将来の世代のCPUで状況が改善される見込みは低く、システムがその第1世代の上限にあると考えられます。スケラブルな設計は、マルチコアプロセッサの改良と歩調を合わせて、性能向上を実現できます。

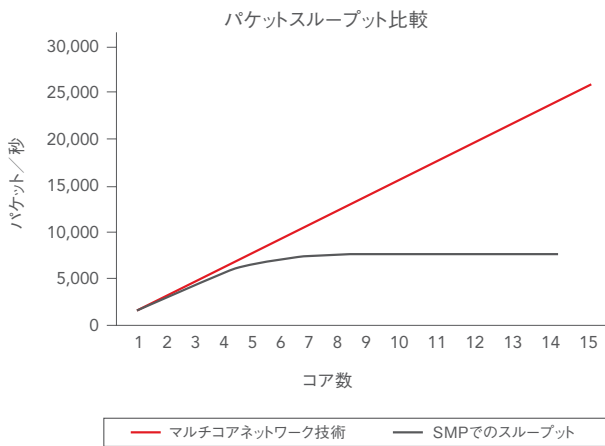


図11: スケラビリティのテスト

## プロセッサのサポート

マルチコアチップはさまざまなメーカーから提供されており、近い将来ますます増えることが予想されます。システム設計者は適切な評価基準を使って、プロジェクトの目標に最も適したチップを選択できます。

あるプロジェクトに最適なチップが、すべてのプロジェクトに適しているわけではありません。チップの選択が製品の世代間で異なるケースもあります。コスト、機能、保守性、市場投入までの時間、パートナーの好み、現在の設計と次世代の設計までの数か月の間に変わる可能性があります。最適化されたシステム設計の一部にプロセッサに依存したコードが含まれる場合でも、複数のプロセッサプラットフォームでサポートされているマルチコアネットワークアクセラレーション・ソフトウェアなら、ハードウェアの変更があっても無駄になりません。

マルチコアによるパケット処理は、システムのソフトウェアの重要部位になる可能性があります。投資を長期間有効に生かすには、複数のプロセッサメーカーとの協力でソフトウェア専門技術で長年の実績を積んだパートナーが提供するソリューションを選択することが大切です。

## 結論

マルチコアプロセッシングは破壊的革新をもたらす技術です。ネットワークパケットは、マルチコアシステムで格段に効率よく処理できますが、ソフトウェアが入念に最適化されている場合に限られます。既存のネットワークスタックでは、マルチコア環境で十分な性能が出ません。マルチコアネットワークの潜在力を発揮させるには、新たなパラダイムが必要です。プロセッサデバイスのプログラミング、OS内部、ネットワーク内部について専門技術が求められます。この3つのうち、2つだけでは不十分なことを忘れないでください。

現在マルチコアによるパケット処理は、マルチギガビットの性能を必要とするエッジ、アグリゲーション、コアのデバイスで利用されています。しかしマルチコアパケット処理は、コアからエンドユーザまでネットワークデバイスの全領域で性能向上を実現できます。

ウインドリバーは、マルチコアパラダイムシフトの最先端に立っています。ウインドリバーが提供するマルチコアソフトウェアとツール分野の広く深い専門技術は、この競争の激しい新たな環境での成功を支援します。ウインドリバーのマルチコアソフトウェアには、SMPおよびAMP対応のOS、仮想化用ハイパーバイザ、データプレーンコア用ベアメタル・エグゼクティブ、ワイヤスピードのパケット転送や他の機能を実現するネットワークアクセラレーション技術、マルチコアプロジェクトのライフサイクル開発全体をカバーする包括的なツール環境などが用意されています。ウインドリバーのマルチコアソフトウェアは主要なマルチコアプロセッサに組み込まれ、最適化されてきました。ウインドリバーのマルチコアソリューションは、設計の柔軟性を最大限に高め、現在そして将来のプロジェクト要件に合わせて拡張することが可能です。

## 注記

1. ここでは対称型および非対称型マルチプロセッサ構成は、コア自体の機能ではなく、OSの観点から定義しています。対称型コアは1つの共通OSが動作します。複数のOSが動作するコアは非対称型です。あるコアが、一部のコアに対しては対称型（共通OSが動作）であると同時に、別のコアに対しては非対称型（異なるOSが動作）であるケースも可能です。
2. これは「普通の」SMPです。別のSMP構成では、プロセッサ予約またはアフィニティを利用して、特定のタスクを特定のコアに専用に割り当てながら、その他のコアを共通プールで扱います。このような動作は、AMP構成に動作の点では似ていますが、すべてのコアが1つの共通OSを共有しているため、SMPとみなされます。
3. システム全体で行われるアグリゲート作業の場合は、理論的に真実です。しかし、パケットスループットのような個別の性能特性を測定する際は、比例的な伸びを達成できるのは、有限の範囲内です。
4. 特に、TCBとソケットは共有される必要があります。効率的なOSスピンロックやアトミックなハードウェア機能により、最小のオーバーヘッドで共有された構造を保護できます。

ウインドリバーは組み込みソフトウェアとモバイルソフトウェアのリーディングカンパニーです。  
企業がデバイスソフトウェアを、より早く高品質かつ低コスト、かつ高信頼性で開発、運用、管理することを可能にします。

**WIND RIVER** ウインドリバー株式会社

<http://www.windriver.co.jp>

東京本社 〒150-0012 東京都渋谷区広尾1-1-39 恵比寿プライムスクエアタワー TEL.03-5778-6001 (代表)

大阪営業所 〒532-0011 大阪市淀川区西中島7-5-25 新大阪ダイビル TEL.06-6100-5760 (代表)

© 2010 Wind River Systems, Inc. Wind River ロゴは Wind River Systems, Inc. の商標です。Wind River、VxWorks は、Wind River Systems, Inc. の登録商標です。  
記載されているすべての名称は、各社の登録商標、商標またはサービスマークです。Rev. 04/2010