

マルチコア技術とVxWorks SMPの概要

目次

要約	1
マルチプロセッシングの概要	1
マルチコア・プロセッサのトレンド	2
マルチコア技術とデバイス・ソフトウェア市場	2
マルチコア・ビジネスの問題点	2
マルチコア技術と従来技術の違い	3
アプリケーションの適合性	3
アムダールの法則：現実的な性能予想	3
マルチプロセッシングのためのOSコンフィギュレーション	5
SMPとAMPの選択	6
VxWorks SMP	6
おわりに	10

要約

マルチプロセッシングの伸びは、非常に重要なトレンドが始まったことを表しています。デバイス性能を新たなレベルまで高める必要があるため、ますます多くのデバイスがマルチコア・プロセッサを使用してデザインされるようになってきました。本資料では、マルチコア技術について詳しく調べ、この技術が次世代のマルチコア・デバイスを開発する組織と開発者に与える選択肢と課題について探ります。また、ウインドリバーのVxWorks SMP製品についても述べ、この製品がマルチコア技術の提供する好機にどのように対応し、ウインドリバーVxWorksプラットフォームのユーザがこのエキサイティングな組込みコンピューティングの新分野に参入することを可能にするかについても言及します。

マルチプロセッシングの概要

マルチプロセッシングとはシステム内で2個以上のプロセッサを使用するもので、これらのプロセッサは協調動作し、全体的な処理負荷を各プロセッサ間で分担します。マルチプロセッシング自体は新しいものではありません。サーバ、企業用システム、通信会社用装置のプロバイダは、これまでも多年にわたってマルチプロセッシングを利用してきました。一部のハイエンド組込みデバイスはバックプレーン・バ

ス上で多数の個別ボードを使用しており、さらには1枚のボード上で複数の個別プロセッサを使用している例もあります。

今日の新技術は単なるマルチプロセッシングではなく、マルチコア・プロセッサを使用したマルチプロセッシングです。マルチコア・プロセッサでは、同一チップ上に複数のプロセッサ・エンジン（コア）が搭載されています。それぞれのコアには、専用のプログラム・カウンタ（すなわち固有の命令ストリーム）があります。チップ・アーキテクチャにより、ある種の相互接続媒体を介してコア同士が「対話」できるようにになっています。通常はローカル・メモリが媒体に使われず。

すべてのマルチコア・プロセッサは、複数のソフトウェアを同時に実行できるようになっています。マルチプロセッシングは真の同時実行を実現します。つまり、まったく同じ瞬間に1つまたは複数のプロセッサ上で複数のソフトウェアを実行しますが、実行時の動きは基本的に異なります。シングルプロセッサ・システムにおけるマルチタスキングやマルチスレッディングでは、通常、複数スレッドの実行を異なる時点でインターリーブしていました。

ハイパースレッディング、同時マルチスレッディング、チップ・マルチスレッディングなどの用語は、すべてシングルプロセッサに適用されるアーキテクチャ手法です。命令は実行待ちのパイプラインを通じてプロセッサに入ります。プロセッサは、メモリから命令がフェッチされるのを待つ間にしばしばブロックされます。複数の命令を実行中でも、やはりプロセッサに入る命令と出てくる結果のシーケンシャル・ストリームがそれぞれ存在します。チップ・マルチスレッディングでは、異なるスレッドの命令間よりも同じスレッド内の命令間のほうが依存する可能性が高いため、この概念は少し拡大されます。

チップ・マルチスレッディングはハードウェアレベルで複数のスレッド間の切り替えを行います。これはソフトウェアにおけるOSの役割によく似ています。しかし、これもやはりマルチコアとは異なります。異なるハードウェア・スレッドが1つのコアに多重送信されるからです。ハイパースレッディング、同時マルチスレッディング、チップ・マルチスレッディングの概念は、相互に排他的なものではありません。チップ・マルチスレッド化されたマルチコア・プロセッサ上でマルチタスキングOSを実行することは可能です。

これらの技術的発展は、複数の個別プロセッサや個別ボードを使って可能になったよりもはるかに多くのデバイス・アプリケーションにおいて、マルチプロセッシングのコスト、電力、スペースを所定の制限内に収める役割を果たしました。マルチコア・プロセッサを使用することによって、デバイス性能が向上し、コストおよびスペースが削減され、多くの機能が実現するという見込みが、より多くのデバイス企業にとって、マルチプロセッシングの世界に参入する動機付けとなるので、このことは非常に重要です。

マルチコア・プロセッサのトレンド

半導体メーカーは、マルチコア・プロセッサへのトレンドを牽引しています。これまで多年にわたり、チップ上により多くのトランジスタをパッケージングしクロック周波数を上げることによって、高性能化は実現されてきました。これらの進歩は、顧客が受け入れることができる電力とコストの予算内で実現することができました。しかし、今や半導体業界は、これまでの戦略を支えてきた物理的過程ではもはやそのような進歩を実現できないところまで来ています。半導体メーカーは、これまでのように周波数を上げることも、デバイスの電力消費を妥当なレベルに抑えることもできなくなっています。図1に、周波数を下げることが電力消費にどのような影響を及ぼすかを示します。

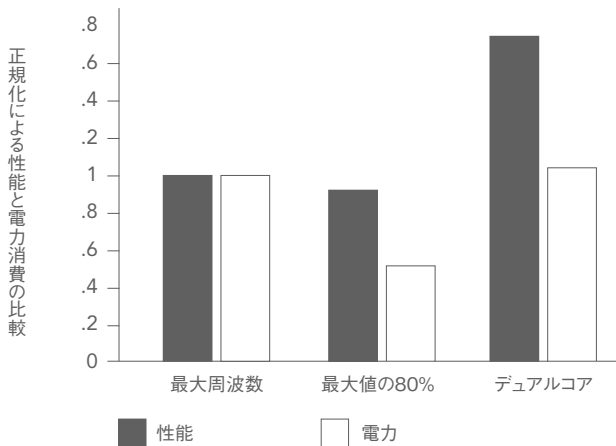


図1: 正規化による性能と電力消費の比較—左側が基準プロセッサ

プロセッサにおける電力消費量は電圧の3乗に比例し、電圧と周波数は比例関係にあります。したがって、周波数が低ければプロセッサの駆動電圧も低く、消費される電力も大幅に減ります。図1の中央に示すように、クロック周波数を10%下げれば性能も10%下がりますが、電力は大幅に削減できます。性能を少し犠牲にすれば、電力消費量は大幅に減ります。したがって、クロック周波数をわずかに下げてこれらのプロセッサを同一チップ上に2つ置くことができれば、理論的には図1の右側に示すような結果が得られるはずです。つまり、電力消費量はほぼ同じで、はるかに高い理論的処理能力を持つプロセッサが得られます。

微細化して搭載できるトランジスタ数が増加したことによって、チップ上に複数のCPUコアを配置して幾分低い周波数で動作させることが、経済的に実現できるようになりました。マルチコア・プロセッサを生み出したのは、このようなトレンドです。しかし、ソフトウェア・エンジニアにとって、このようにして提供される処理能力の向上を利用することと、クロック周波数を上げることによって向上した処理能力を利用することは、基本的に異なります。

マルチコア技術とデバイス・ソフトウェア市場

技術的な市場調査と戦略コンサルティングを行う独立系企業 Venture Development Corporation (VDC) の調査によれば、デバイス・ソフトウェア市場はマルチコア技術に強い関心を示し、また、その利用例も増えています。すでに数多くのデバイス・メーカーが、高性能システムにマルチコア・プロセッサとマルチプロセッシングを採用。エンタープライズ分野においてはマルチコア・プロセッサがすでに主流を占めており、サーバ用に出荷されるIntelプロセッサは10個のうち9個までがマルチコアです。このようにエンタープライズ市場は、すでにマルチコア・プロセッサへの移行を完了しています。

デバイス・ソフトウェア市場は、まだそこまでには至っていません。しかし傾向としては、マルチコア化の流れが変わることはないでしょう。半導体メーカーは、より小さなパッケージと低いコストで計算能力を提供することに凌ぎを削っているため、すべて自社固有のマルチコア・プロセッサを製造しています。Intelなど一部のメーカーは、すべての製品ラインをマルチコア化する方向に舵を切りましたが、Freemoveなどの他メーカーは、それぞれの市場に応じてマルチコアとシングルコアのバランスの取れたポートフォリオを維持しています。

より多くのデバイスがより高性能のプロセッサを必要とします。また、設計者や開発者が電力、重量、スペース、コストの制約の中でより高性能のデバイスを実現するには、マルチコア・プロセッサを採用するより他に選択肢がなくなっています。COTS (commercial off-the-shelf) ボードのメーカーも、このトレンドを支えています。今日では、Curtiss-Wright、Kontron、Radstone、Mercury Computerといった企業から、マルチコア・プロセッサをベースとした数多くのCOTSボードが市販されています。

マルチコア・ビジネスの問題点

マルチコア・プロセッサの利点は、電力とコストのレベルが同じ他のあらゆる高性能プロセッサの利点と同じです。マルチコア・プロセッサはより多くのソフトウェアを同時に実行することができ、その結果、デバイスの高速化、より多くの機能の追加、性能の最適化を実現し、最終的に顧客にとっての価値を向上させます。

しかしマルチコアを採用した場合、すべてのデバイス・メーカーではないにしろ、ほとんどのメーカーにとって、現実にコストが発生します。大多数の開発者は、まったく初めての経験として、本当のソフトウェア同時実行を扱う必要があります。マルチプロセッシング環境用に新しいソフトウェアを記述したり既存のソフトウェア投資を再利用したりすることは、より複雑な仕事である上に時間もかかり、プロジェクトにより多くのリスクをもたらすことになります。すでにマルチプロセッシング・システムの構築を完了したデバイス・メーカーにとっては、マルチコア技術への移行は比較的簡単です。しかし、これまでシングルプロセッサ・システム用のソフトウェアしか作成したことのない多くのデバイス・メーカーは、基本的に異なる環境に足を踏み入れることになります。

マルチコア技術と従来技術の違い

マルチコア環境用のアプリケーション開発は、次の3つの側面において、シングルコア環境用のアプリケーション開発とは基本的に異なります。

1. マルチプロセッシングによってアプリケーションの性能が保証されるわけではありません。また、マルチプロセッシングによってすべての問題を解決できるわけではありません。作業を行うために、基本的に異なるアルゴリズムを見つけることが必要な場合もあります。既存のアルゴリズムはユニプロセッサ環境では機能しても、複数プロセッサに利用を拡大することはできないこともあります。
2. システム・デザインはシングルプロセッサ・システムよりも複雑で、同時に実行できるアルゴリズムを持っていることが前提となります。マルチプロセッサを利用するために、ハードウェアによるマルチプロセッシングのサポート方法、OSとシステム・ソフトウェアによるマルチプロセッシングのサポート方法、アプリケーションの分割方法の間に密接な関連があります。シングルプロセッサ・システムよりも高い性能を備えたシステムを実現するには、ユーザが、ハードウェア、OS、アプリケーション・デザインの適切な組み合わせを選択する必要があります。最大限の性能を実現する最適な組み合わせを得るには、多くの場合、トライ・アンド・エラーを何度も繰り返すことになります。
3. ソフトウェア開発環境は非常に複雑なものとなります。プログラマは多年にわたってシーケンシャル・プログラミングの概念を学び、また、実行してきました。たとえば、アプリケーションの複数のスレッドが同時に動作している場合、隠れていたプログラミングの欠陥が露呈します。これは、そのソフトウェアが記述された当初の基本的な仮定に反したからです。つまり、同時に実行されるコード部分が、同時に実行されることを想定していなかったということです。開発者は、同期に関わる問題（つまり、以前は存在しなかった共有データ関連の新たな同期）や、これらの仮定によって現れてくる競合状態に対処する必要があります。これら異なるスレッドすべての相互動作、バックグラウンドにおける割り込み動作、OSによるスレッドのスケジューリング方法などにより、ソフトウェアの動きは決定性に乏しくなります。

アプリケーションの適合性

1つの仕事があって、その実行に2人の人間があたれば、1人の場合よりも早く仕事を終わらせることができます。マルチコア・プロセッサのコアは、1つの仕事に振り向けることのできる作業者のようなものです。どの作業者の手も空かないようにするには、十分な量の仕事が必要なのは明らかです。この場合、すべての作業者に何らかの作業を割り当てるには、その仕事を十分な数に分割する必要があります。しかし「船頭多くして船山に上る」のたとえのように、作業者が多くいさえすればそれだけ早く問題を解決できるわけではありません。作業者同士がうまく協力しなければ、効率よい作業は望めません。

工場の組立ラインはマルチプロセッシングの一例です。複数の作業をできるだけ並行して進めることができるよう注意深く組み合わせることによって、数多くの仕事を完了させることができます。しかし、作業を進めるための実際のステージ間には依存性もあります。この依存性が適切でなかったり作業が正しく完了しなかったりすると、その後のステップが行き詰ってしまい、生産を完了できなくなります。ほとんど依存性のないアプリケーションは、梱包を仕分けして適切な配送トラックに積み込む配送倉庫に例えることができます。それぞれの梱包は、他の梱包には関係なく個別に処理されます。このシステムは非常に高い並列性を有しています。課題は共有リソ

スの並列性です。配送倉庫の例では、梱包を移動するためのカートの数が少ない場合、すべての作業者がカートの空きを待つこととなります。この状態をリソース競合と呼びます。

タスクによっては、より多くの人員やリソースを投入しても早く終わることができないこともあります。読書などはその例です。これは厳密なシリアル処理です。スーパーコンピュータや大規模並列コンピュータは必ずしもあらゆる計算問題に使われるわけではなく、気象予報や金融モデリングといった特殊なアプリケーションにしか使われません。

アムダールの法則：現実的な性能予想

アムダールの法則は、理解しておくべきもう1つの制限事項です。あらゆる問題には、並列で処理できる作業と、順番に（つまりシリアルに）処理しなければならない作業が一定の比率で存在します。料理を例に考えてみましょう。ある種の作業は特定の順番で進めなければなりません。並列（パラレル）で進めることができる作業もあります。どれだけ多くの作業（あるいはプロセッサ）をその問題に投入できるかに関わらず、どれだけ早く作業を完了させることができるかという問題には、どれだけ作業を並列で行うことができるかに応じて限界があります。アムダールの法則は次のように表わされます。

$$\text{並列処理による速度向上} = 1 / (\text{シリアル\%} + (1 - \text{シリアル\%}) / \text{プロセッサ数})$$

ここで、シリアル%はシリアル処理しなければならない作業のパーセンテージで、(1-シリアル%)はパラレル処理できる作業のパーセンテージです。図2は、アプリケーションのシリアル処理の度合と使用可能なプロセッサ数にアムダールの法則を適用した時の結果を示したものです。

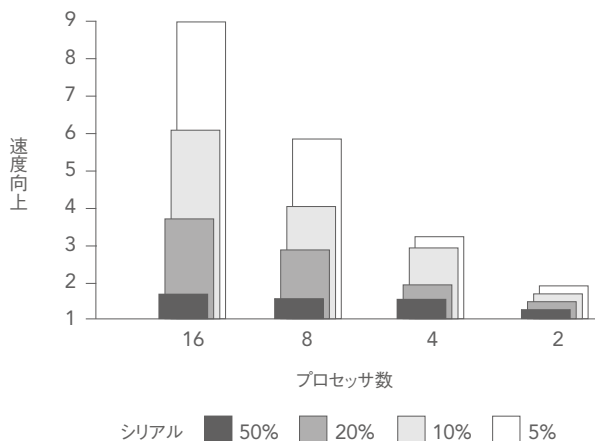


図2：アプリケーションのシリアル処理の度合と使用可能なプロセッサ数にアムダールの法則を適用した結果

シリアル処理の比率が5%の場合は、プロセッサ数の増加とともに比較的リニアに処理速度が向上していきます。しかし16プロセッサの場合を見ると、最大の速度向上は9倍です。シリアル処理比率が5%でも、16プロセッサ・システムで実現可能な最大速度向上率は16倍を大きく下回ります。シリアル処理の比率が増えるにつれて、速度および性能の最大向上率は大きく減少します。シリアル処理比率が50%では、プロセッサ数が4個以上に増えても性能向上率はほとんど変わりません。シリアル処理比率が10%のデバイスを作成する開発者は、8コアシステムによって望ましい速度向上が得られるとは思っていません。この場合の速度向上率は、シングルコアの4倍に過ぎません。

シリアル処理しなければならない作業の量がわずかに増えただけでも、性能向上幅が大きく制限される可能性があります。したがって、多くのアプリケーションでは、CPUを追加することによって性能を向上させようとしても限界があります。ソフトウェアのシリアル処理の量を減らすためのツールと手法が、マルチプロセッシング用にデバイスを最適化する鍵となります。

並列化をより高い度合いで（あるいは完全に）実現できる問題は、マルチプロセッシングによってスループットや性能を大幅に向上させることができます。一例がデータプレーン上のネットワークングです。この場合、システムは、受信パケットの複数の独立したストリームを取り扱います。それぞれのストリームは他のストリームとは独立した存在なので、データ・ストリームの数だけプロセッサを割り当てることができます。チップ上に多くのプロセッサがあることにより、シングルプロセッサ・システムと比較してより小型で消費電力の少ないシステムを実現でき、かつ多くの作業ができるので、マルチコア・システムはこのようなシステムに有効です。

マルチプロセッシング、特にマルチコアは、ハードウェアのアーキテクチャに大きな影響を及ぼします。さらにこれによってシステム・ソフトウェアのアーキテクチャが決定され、最終的にアプリケーションのアーキテクチャにも影響を与えます。

所定のアプリケーションにおける作業を並列化するための分割方法は数多くあります。アプリケーション分割は、デバイス・メーカーがマルチプロセッシング・システムをデザインする際に考慮しなければならない最も重要な側面の1つです。

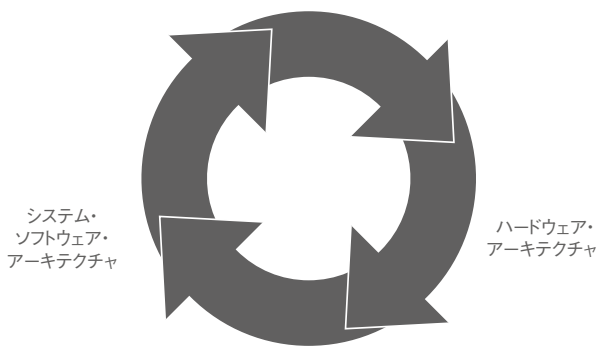


図3: アプリケーション・アーキテクチャ

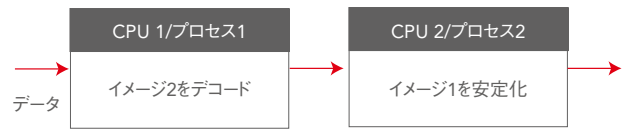


図4: 粗粒度シリアル実行

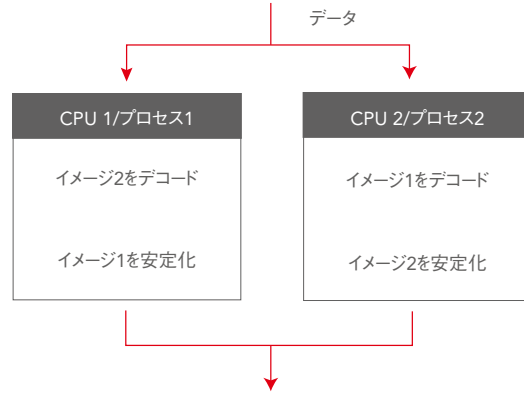


図5: 粗粒度パラレル実行

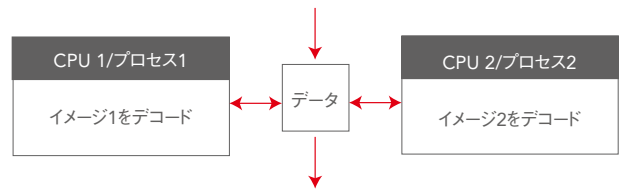


図6: 細粒度同時データアクセス

図に示したすべてのアプリケーション分割の有効性と妥当性は、問題によって異なります。特定の問題やハードウェア・デザインに対してどのアプリケーション分割デザインが適しているかは、システム・デザイナーが判断する必要があります。最上段はパイプライン状のデザインで、組立ラインによく似ています。最下段では、個々の作業が複数の作業者に分散されています（配送倉庫の例と同様）。

最も複雑な分割デザインが図6です。このデザインは細粒度分割を扱うものです。ここでは1セットのデータがあり、必要な性能を得るために、そのデータに対して複数の処理を完了させる必要があります。これらの処理を完了させるにあたって、厳密な順番は必要ありません。最大限の性能を得るには、複数の作業者が同時に同じ問題に取り組む必要があります。これは特定の作業に多数の作業者を割り当てることに似ていますが、これらの作業者がすべて効率的に同じ作業を行い、しかも、より短い時間で完了させることができるような方法を用いる必要があります。各作業者が大きな作業の一部を受け持ち、なおかつ他の作業者とは独立した形で作業を行います。作業は、比較的大きな作業ブロックを各作業者に割り当てる場合よりもさらに細かく分割されます。

マルチプロセッシングのためのOSコンフィギュレーション

マルチプロセッシング・システムに適したOSには、2つのコンフィギュレーションがあります。対称型マルチプロセッシング (Symmetric Multiprocessing: SMP) と非対称型マルチプロセッシング (Asymmetric Multiprocessing: AMP) です。

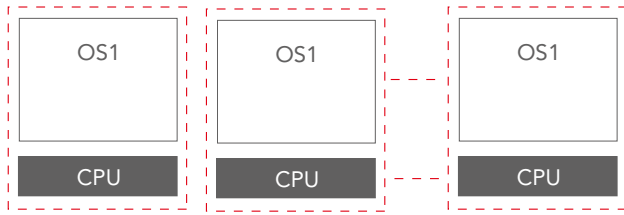


図7: 非対称型マルチプロセッシングOSコンフィギュレーション

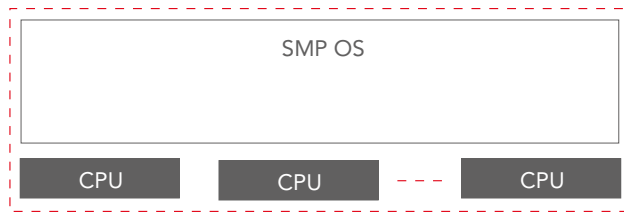


図8: 対称型マルチプロセッシングOSコンフィギュレーション

AMPコンフィギュレーション (図7) では、OSはマルチプロセッシング・システムの各プロセッサ上、あるいは各コア上で実行されます。それぞれのプロセッサ/OSの組み合わせは、その固有権限において実質的にシングルプロセッサ・コンピュータです。全体としてシステムは複数のこれらのコンピュータ (ノードとも呼ばれます) から構成され、これらが互いに呼応して与えられたジョブを処理します。これらの独立ノードを1つの大きなシステムにまとめるのが、プロセッサが互いに通信を行うために使用する相互接続メカニズムです。この相互接続メカニズムとして最も一般的なものはプロセッサ間にある共有メモリのバンクですが、ネットワーク接続やその他のペリフェラル・バスが使われる場合もあります。AMPシステムにおいては、個々のノードが実行するジョブを、システムのデザイン時に決定する必要があります。したがってシステム全体としてのジョブは、システム内の個々のノードに割り当てられた作業項目に分割されません。

AMPシステムは、さまざまな組み合わせで実現されます。このシステム内では、異なるプロセッサを選ぶことができます。また、AMPシステムでは、ユーザの必要に応じて異なるOSを選択できます。たとえば、VxWorksでリアルタイム性が求められるタスクを処理し、Wind River Linuxでメディア・インタフェースを処理することが可能です。場合によっては、完全なOSを必要とせず、ハードウェアを直接利用する簡単な実行型プログラムだけあればよいという、非常に限定的な機能しか実行しないノードもあります。

AMPシステムは通常、異なるOSとそのアプリケーションをすべて保持するために、SMPシステムよりも多くのメモリを必要とします。AMPシステムは、その本来の性質上、開発者がデザインするハードウェア・コンフィギュレーションの緻密さに大きく依存します。異なる価格帯のデバイス・ファミリーを作る時のように、プロセッサ数や相互接続メカニズムが異なる別のハードウェア・コンフィギュレーションに移行する際には、アプリケーションを再分割し、新しいハードウェア・コンフィギュレーション上でもシステムが機能することを再確認する必要があります。

SMPコンフィギュレーション (図8) では、1つのOSが、複数の同じプロセッサ (マルチコア・プロセッサの場合はコア) を制御します。シングルプロセッサ・システム同様、アプリケーションが「見る」OSは1つだけで、相互動作もそのOSを通じて行われます。複数のプロセッサがあるという事実は、OSの働きによってユーザからは見えなくなっています。その意味では、SMPオペレーティングシステムは、ハードウェアの詳細をユーザに対して抽象化する役割を果たします。また、SMPオペレーティングシステムは対称的でもありません。つまりこのシステムは、複数の同一プロセッサと連携する必要があります。それぞれが、システム内のすべてのメモリやデバイスに、同じ方法でアクセスできる必要があります。したがって、SMPシステム内のすべてのプロセッサ (またはコア) は、システム内の他のプロセッサが実行可能なタスクをすべて実行できます。ハードウェアにおけるこの対称性によって、SMPオペレーティングシステムは、システム内のどのプロセッサにでもすべての作業を割り当てることができます。SMPオペレーティングシステムはすべてのプロセッサをビジー状態にしてアプリケーション・スレッドを実行しようとはしますが、これは結局、システムの行う仕事のロード・バランシングを取ることであります。

システム内におけるハードウェア抽象化とロード・バランシングの結果として、SMPオペレーティングシステムは、SMPハードウェア上で動作するソフトウェアの開発の仕事容易なものにします。プログラマの視点からすると、アプリケーションの記述の対象となるOSは1つだけであり、このOSが、作業負荷を使用可能なすべてのプロセッサに自動的に分散させます。したがって、SMPオペレーティングシステムは、ユニプロセッサ・システムと同じプログラミング・セマンティクスを提供します。

SMPシステム内のすべてのプロセッサは、1つのOSの管理の下に1つのメモリ・プールを共有します。個々のプロセッサのメモリとローカル・キャッシュは、チップのハードウェアによって同期が保たれます。したがって、SMPシステム内のアプリケーションは、非常に容易かつ効率的にメモリ内で互いにデータを共有することができます。この特性が、SMPシステムを、少ない待ち時間でデータを大量に共有する必要のあるアプリケーションに適したものにしています。

SMPオペレーティングシステムは複数のプロセッサの同期と制御を行う必要があるため、1つのプロセッサだけを制御するOSに比べて必然的に内部オーバーヘッドが大きくなります。個々のベンチマーク比較では、SMPシステムはほとんど常にユニプロセッサ・システムよりも遅いという結果が得られますが、アプリケーションの並列性が高く、そのため所定の時間内にユニプロセッサ・システムよりも多くの作業をこなせる場合は、SMPシステムの方がユニプロセッサ・システムよりも優れた性能を示します。

SMPとAMPの選択

OSがさまざまな並列作業のロードバランスを効果的に取れると予想される時は、SMPオペレーティングシステムが最良の選択です。SMPを実行するためのハードウェアにとって不可欠な特性の1つは、ハードウェアによる強制的なキャッシュ・コヒーレンスです。これは、プロセッサのキャッシュ内にあるデータが別のプロセッサによって変更された場合に、そのデータのコピーをハードウェアが自動的に更新するものです。実行中のプログラムにとってこの更新は透過的に行われるので、変更されたデータはすぐにメモリに反映され、高速メモリによるアクセスが可能です。メモリ内にあるデータの多くをタスクが共有する場合、SMPオペレーティングシステム内では、メモリのアクセス速度でこのデータにアクセスするのが迅速かつ容易な方法です。アプリケーションをコア数の異なるシステム間で移植可能なものとする必要がある場合も、SMPが適切な選択です。これらいずれの場合も、SMPカーネルがハードウェアの抽象化とロード・バランシングを行うので、アプリケーション開発者が、異なるハードウェア・コンフィギュレーションに合わせてアプリケーションをチューニングする必要はなくなります。

AMPが最も適しているのは、異なるサブシステムに明確に分割されていて、それらがほとんど自律的に動作しながらも明確に定義され、制限された他のノードとのデータ通信の必要があるようなシステムです。AMPによってシステム・デザイナーは、システム内の各ノードが実行する機能に最も適した異種のプロセッサをミックスして、ハードウェアを設計することができます。AMPシステム内の各ノードは、それぞれが独自の権限を持つスタンドアロンのシングルプロセッサ・コンピュータなので、AMPは高い可用性が求められる冗長システムにも非常に適しています。

VxWorks SMP

ウインドリバーのVxWorks SMPは、すべてのWind River VxWorksプラットフォーム（6.6ベースおよびそれ以降）を対象としたアドオン製品で、VxWorksにSMP機能を提供します。VxWorks SMPはマルチコア・プロセッサを利用してアプリケーションの真の同時実行を実現し、並列性を通じてアプリケーションの性能を向上させることができます。

VxWorks SMPアドオンは、VxWorksプラットフォームと組み合わせることにより、ランタイム、ミドルウェア、市場をリードするウインドリバーの開発環境から構成される、完全な対称マルチプロセッシング対応プラットフォームをユーザに提供します。

それぞれのOSコンフィギュレーションの長所と短所を以下の表に示します。

AMP	SMP
選択理由 <ul style="list-style-type: none"> ● 緩結合ノードにアプリケーションを容易に分割できる ● 信頼性確保のために冗長性が必要 ● ハードウェア・コンフィギュレーションがSMPに向いていない ● 複数のプロセッサ/OSタイプが必要 ● ハードウェア・リソースへのアプリケーション割り当てを明示的に行う必要がある ● スケジューリング・オーバーヘッドを最小限に抑える必要がある 	選択理由 <ul style="list-style-type: none"> ● プロセッサをビジー状態に保てるだけの十分な並列性をアプリケーションが有している ● タスク/スレッドが頻繁に大量データの共有を必要とする ● ロード・バランシングが重要 ● 異種ハードウェア間でのアプリケーションの移植性が不可欠
注意点 <ul style="list-style-type: none"> ● 一般にAMPデザインにはより多くのメモリが必要（複数のOSイメージ） ● ハードウェア・コンフィギュレーションに合わせてAMPシステムをチューニングする必要がある ● コア間の通信設定とデバッグが複雑 	注意点 <ul style="list-style-type: none"> ● SMPカーネルは、ユニプロセッサ・カーネルよりオーバーヘッドが大きい ● OSコールを頻繁に行うアプリケーションは、ユニプロセッサ・システム上よりも実行速度が遅い ● ユニプロセッサ・システムからコードを移植する場合、同時実行を前提とすることによって、それまで隠れていたソフトウェアの欠陥が露呈する可能性がある

SMP機能を提供することに加え、VxWorks SMPは、ユニプロセッサ・バージョンのVxWorksとの互換性を維持できるようにデザインされています。これは、VxWorksユーザが、そのVxWorksアプリケーション、ドライバ、その他のレガシー・ソフトウェアを、ユニプロセッサ環境（VxWorks UP）からSMP環境へ容易に移植できるようにするためです。

VxWorks SMPは、Wind River VxWorks 6.6プラットフォーム（Wind River General Purpose Platformのバージョン3.6、Wind River Platform for Automotive Devices、Wind River Platform for Consumer Devices、Wind River Platform for Industrial Devices、またはWind River Platform for Network Equipment）用の個別のアドオン製品として販売されます。インストール後、ユーザは必要に応じて、ユニプロセッサ・アプリケーションまたはSMPアプリケーションの開発をどちらも行うことができます。



図9: VxWorks SMPインストールとWind River VxWorksプラットフォーム

図9は、VxWorks 6.6インストールとSMPを示したものです。

VxWorks SMPの機能

VxWorks SMPは、世界をリードするVxWorks RTOSに対称マルチプロセッシング機能を提供します。VxWorks SMPは、VxWorksと同じソース・ベースから作られています。具体的には、VxWorks SMPは以下の機能を提供します。

1. SMP機能

VxWorks SMPは、業界最先端を行くマルチコア・プロセッサ上のSMP環境内でアプリケーションを実行できる機能をVxWorks ユーザに提供します。これにより、シングルプロセッサ・システムが可能にしたもの以上に豊富な機能を備え、より高性能のデバイスを作り出すことが可能です。

2. 互換性

VxWorks SMPでは、ユニプロセッサ・バージョンのVxWorksと共通のAPIを使用することができます (一部例外あり)。APIの互換性は既存のVxWorksベース・ソフトウェアの再利用を促進し、企業は過去のVxWorksへの投資を生かすことができます。リアルタイム処理やエラー検出/レポートなどの主要な機能は、SMP環境においてもすべてサポートされています。

3. リアルタイム動作

VxWorks SMPは、VxWorksのリアルタイム動作や特性を犠牲にすることなく、VxWorksにSMP機能を追加します。このためVxWorks SMPは、ユニプロセッサVxWorksシステムと同じターゲット市場やターゲット・アプリケーションに応用することができます。VxWorks SMPは、優先順位に従ったタスク実行を保証する決定性スケジューラを備えています。このスケジューラは、実行準備の整った優先順位の高いN個のタスクをディスパッチします。ここで、Nはシステム内のプロセッサ数です。優先順位を基準にした決定性スケジューリングも、シングルプロセッサ・システムにおいてVxWorksを決定的に差別化する特長の1つで、これ

がRTOSと汎用OSとの相違点です。VxWorks SMPでは、割り込みレベルの並列性、具体的には、システム内の複数のコアが異なる割り込みを同時に扱う機能が実装されています。これは、システム全体としての応答性を向上させます。OS内の多くの重要な動作はスピンロックによって保護されています。これらの動作は決定された時間内に完了しますが、この時間は、スピンロックが保持される時間のみの関数です。スピンロックは、割り込み処理中の重要データの保護にも使われます。これらのスピンロックは決定された時間だけ維持されますが、システムの割り込み待ち時間も同様に決定されたものとなります。

4. 最新マルチコア・シリコンのサポート

VxWorks SMPは、現在市場に存在するすべての最新マルチコア・プロセッサ上で動作します。これには、ARM、Broadcom、Cavium、Freescale、Intel、Raza Microelectronicsのマルチコア・プロセッサが含まれます。これにより、ユーザは広範な選択肢の中からハードウェア・プラットフォームを選択して、次世代のデバイスを作り上げることができます。

5. 異種アーキテクチャ間での移植性

これが、発表以来のVxWorksの決定的特長です。ユニプロセッサ版と同様に、VxWorks SMPは移植性に優れたOSであり、さまざまなプロセッサ上で動作し、その特性も一定しています。SMPは、このユニークなVxWorksの機能をマルチプロセッシングの領域にそのまま拡張します。

6. 使い慣れた開発環境

Wind River Workbenchとこれを構成するツール群は、ユニプロセッサ版、SMP版、どちらのVxWorksでも使用できます。また、これらのツールは、すべてのコア内の並列実行とローディング・パターンを可視化するようにデザインされた、VxWorks SMP用の追加機能も備えています。以下の項では、これら各種ツールの機能を詳しく説明します。

VxWorks SMPが適するアプリケーション

VxWorks SMPは、決定性、少ない待ち時間、スモールフットプリントなどといった、SMPオペレーティングシステムとRTOSの機能が必要とされる問題に適しています。前の項では、SMPオペレーティングシステム・コンフィギュレーションの特長と固有機能を示しました。ポイントを言い換えると、VxWorks SMPは以下に示す条件のうち1つ以上があてはまる場合に適したソリューションといえます。

1. アプリケーションが並列スレッドの観点からデザインされており（あるいはされる予定であり）、システムが、ユニプロセッサ VxWorksシステムで可能な作業よりも多くの作業を同時に行えるようになっている。マルチコア・プロセッサを使用することによってしかるべき性能向上を実現できるようにするためには、アプリケーションが十分な並列性を備えている必要がある。アムダールの法則を考慮し、与えられた並列性のレベルに対してどれだけ性能向上を図れるか、上限を設ける方法に留意する。
2. スレッドが大量のデータを共有する、あるいは細粒度のデータ共有のデザインである。
3. システム・デザイナーが、個々のプロセッサに対してアプリケーションを明確に分割するのではなく、OSに依存して、実行準備が整ったスレッドを使用可能なすべてのプロセッサに割り当ててくれることを望んでいる（ロード・バランシングとも呼ばれる）。

たとえば、ネットワーク・ルータでは、複数のコアに多くの異なるストリームからのパケットを並列に処理させることによって、スループットを向上させることができます。また、いくつかのコアをセキュリティ処理用のハードウェア・オフロード・エンジンのように動作させ（その中の1つのコアがメインのTCP/IPスタック・ロジックを実行可能）、一方で他のコアをルーティングあるいは暗号化機能専用を使用することによって、ネットワーク・スループットの向上を実現することもできます。ネットワーク・ルータの例では、複数のコアが新たな独立したデータ・ストリームを並列に処理でき、もう1つの例では、ネットワーク処理負荷の一部が他のコアによって並列に処理されます。これらの例はいずれもマルチプロセッシングが提供する可能性を示すものです。マルチプロセッシングは、所定の時間内に、1個のプロセッサで可能な量よりも多くの処理を行います。

VxWorks SMPの新機能

ここでは、VxWorks SMPの主な新機能とAPIの一部を紹介します。これはすべての機能を網羅するものではありません。詳細については www.windriver.com を参照していただくか、ウインドリバーの担当者にご連絡ください。

1. スピンロック

スピンロックは、SMPシステム内の異なるプロセッサ上で実行されるスレッドと割り込みハンドラの同期に使われる、軽量・高速のマルチプロセッサセーフ・メカニズムです。スピンロックは、マルチプロセッサ環境用の軽量ミューテックス・セマフォと考えることができます。スピンロックには2つの基本タイプ（タスクレベルと割り込みレベル）があります。異なるタスク同士間、およびタスクと割り込みサービス・ルーチン間の両方の同期を取るためにスピンロックを使用することができます。

2. アトミック演算子

アトミック演算子は、簡単なメモリ変数に対するマルチプロセッサセーフな高速演算、たとえばインクリメント、デクリメント、ビットワイズ論理演算、単純算術演算などを提供するAPIクラスの1つです。これらの演算子はシステム内の他のプロセッサに対して変数をアトミックに更新し、割り込みロックやプリエンブション・ロック（これらは、ユニコア・プロセッサやユニプロセッサ・システムでよく使われます）といった低速で待ち時間の長いメカニ

ムを使用する必要をなくします。VxWorks SMP用に開発されたアトミック演算子は、ユニプロセッサ・バージョンのVxWorksでも使用できます。VxWorks 6.6によるすべてのVxWorksアプリケーションの開発で、これらの演算子の使用が推奨されます。

3. リーダ・ライター・セマフォ

リーダー・ライター・セマフォは、VxWorks 6.6における新しいタイプのセマフォです（ユニプロセッサ・システムとSMPの両方で使用可）。リーダー・ライター・セマフォを使用すれば、多数のリーダー・スレッドが、共有データ構造に迅速にリード・アクセスできます。データ構造の変更を許可されるライター・スレッドは、同時点において1つだけです。これらのセマフォは、リーダーが複数でライターが1つという状況に合わせて最適化されています。これらはProducer-Consumerタイプのソフトウェア・アルゴリズムにおいて使用が推奨され、マルチプロセッシング・システムには特に適しています。VxWorksミューテックス・セマフォ同様、リーダー・ライター・セマフォは、優先順位の逆転の保護もオプションでサポートしています。

4. スレッドローカル・ストレージAPI

スレッドローカル・ストレージは、スレッドローカル変数ストレージを管理するための新しいマルチプロセッサセーフ機能です。これは、従来からあるどちらもSMP環境では使用できないVxWorksカーネル・モードのtaskVarLib機能およびユーザ・モードのtlsLib機能に替わるものです。

Wind River WorkbenchとVxWorks SMP

VxWorks UPとSMPは、同じWind River Workbench開発環境を共有します。Workbench環境は、SMP環境のデバッグと解析をサポートする必要からアップグレードされました。Workbench 3.0は、システム・モードまたはタスク・モードのどちらでも、VxWorks SMPカーネルとリアルタイム・プロセス（RTP）をデバッグすることができます。Workbenchの機能強化により、マルチコア環境用のプログラム開発時に発生する競合状態とデッドロックの診断が容易になります。

Workbench内では、VxWorksのユニプロセッサ・バージョンのプロジェクトとまったく同じようにしてVxWorks SMPプロジェクトがセットアップされます。ユーザは、開発を行うにあたってVxWorks UPかSMPのどちらかをプロジェクトごとに選択できます。すべてのプロジェクトやアプリケーションがSMPに適しているわけではないので、ユーザは、そのニーズに最も適したOSコンフィギュレーションを選ぶことができます。Workbench Debugger, System Viewer, Workbench Performance Profiler, Workbench Code Coverage Analyzer, Workbench Memory Analyzer, Workbench Data Monitor, Workbench Function TracerなどのWorkbench機能が更新されたので、これらの機能は、SMP内でもユニプロセッサ・システム内と同様に使用できます。

SMPシステムのデバッグ

ほとんどのデバッグおよびチューニングのワークフローに関して言えば、SMPシステムとユニプロセッサ・システムの間には本質的な違いはありません。SMP固有のワークフローの1つは、最大限のスループットとCPU利用率が実現されているのを確認することです。この特別な問題のため、ユーザがCPU利用と同時実行に関わる相互動作を理解しやすくすることを目的として、System Viewerの機能が強化されました。

SMPシステムにおいて開発者が遭遇するバグと問題は、ユニプロセッサのマルチスレッド・システムにおいて発生する問題と同じ種類のもので、ある種のバグは、ユニプロセッサ・システムよりもSMPシステムで発生しやすくなります。特に、ソフトウェアが同時実行について一定の仮定をしている場合はその傾向が強くなりますが、これは厳密にはSMPの問題ではありません。SMPのデバッグは、ユニプロセッサ・システムと本質的に異なるものではありません。SMPシステムにおいて開発者が遭遇する可能性のある問題とその発見方法に関し、以下にいくつかご案内します。

- 競合状態やメモリ破壊など、保護されていないか、保護が十分でない重要セクションが原因で発生する問題は、スレッドが同時実行されている時に発生する可能性がより高くなります。同時実行するソフトウェアがこのようなエラーを持つコードを実行する可能性は、同時実行をほとんどしないか、まったくしないソフトウェアの場合よりもはるかに高くなるためです。ユニプロセッサ・システムの場合と同様に、メモリ・プロファイリングやSystem Viewerはこれらのエラーを見つける助けとなります。
- たとえば、優先順位の低いスレッドがリソース待ちをしている優先順位の高いスレッドよりも高い優先順位を継承してしまう、といった優先順位の逆転は、他のスレッドがスレッドの優先順位に関する仮定をしている場合に発生する可能性があります。System Viewerは、ユニプロセッサ・システムの場合と同様に、これを可視化する助けとなります。
- タスク割り込みの同時実行に関する問題は、1つのコアがあるタスクを実行中に別のコアで行われる割り込み処理が原因となります。レガシーVxWorksアプリケーションでは、タスクと割り込みの処理は相互に排他的なものだという前提に立っていることが考えられますが、SMPではこの前提は成り立ちません。System Viewerは、ユニプロセッサ・システムの場合と同様に、これを可視化する助けとなります。

通常、ランタイム解析ツールが監視や分析を可能にするすべての一般的な問題は、ユニプロセッサ・システムでもSMPシステムでも同じように確認できます。これらのツールは、マルチコア環境でもユニプロセッサ環境と同じように容易に使用できます。

System Viewer

System Viewerには、SMPのコアとアプリケーションの開発者を支援する新しい解析機能が含まれています。SMPの場合、System Viewerのイベント・グラフにはコアに関する情報が表示されるのでコア内部の状態を知ることができ、ユニプロセッサ・システムの場合と同じように、競合状態、デッドロック、資源の枯渇などを開発者が確認して、これを集中的に検討する助けとなります。これは、性能損失を最小限に抑える上で有効です。System Viewerを使用すれば、ユーザは、システム内で進行中の並列動作を確認できます。タスクレベルの並列性、タスク割り込みの並列性、割り込み同士の並列性など、すべてを共通のタイム・ソースに同期して正確なタイミングを実現できます。これは、アプリケーションの実行特性を可視化する上で有効な手段となり、同時実行が原因で発生するソフトウェアの欠陥を診断する助けとなります。さらにSystem Viewerは、システム内のすべてのコアについて、CPUローディングのレベルを表示します。これは、システム内のすべてのコアを対象に、性能のチューニングと効果的なロード・バランシングを行う上で有効です。

VxWorks Simulator

VxWorks Simulatorを使用すれば、シングルCPUまたはマルチCPUのホストマシン上でSMPアプリケーションをシミュレートすることができます。ユーザはSimulatorを使用して、実際のマルチコア・ハードウェアに移行する前に、基本的なレベルから高度なレベルに至るまで、アプリケーションの移行、シミュレーション、特性付けを行うことができます。Simulatorを実行中でも、ユーザはホストマシン上でSystem Viewerを使用してSMPデータを収集し、これを表示できます。ただし、SimulatorはSMPアプリケーションの動作を正確に示すことはできませんが、基礎となるホストハードウェアが対象となるターゲット・ハードウェアと大幅に異なることが考えられるので、SMPアプリケーションの性能を正確に予想することはできません。

Wind River Analysis Tools

Wind River Run-Time Analysis Tools (これまでScopeToolsと呼ばれていた) は、VxWorks SMPにおいても、ユニプロセッサ・バージョンのVxWorksの場合と同様に機能します。

おわりに

過去においては、プロセッサのクロック周波数を上げることと微細化が、CPU性能の大幅向上への道を開いてきました。しかし最近では、この戦略が物理的に限界に近づきつつあります。これまで以上の性能を追求する中で、半導体産業は単一チップ上に複数のプロセッシング・コアを置く方向に変わってきました。

コンピュータ・サイエンスにとってマルチプロセッシングは新しいものではありませんが、これをデバイス・ソフトウェアに使用することは新しい試みであり、マルチコア・プロセッサの利用増加によって、この傾向は強まりつつあります。高性能で豊富な機能を持つデバイスの実現が確実に期待される一方で、マルチコア技術はソフトウェア開発者に課題と混乱をもたらすものでもあります。シングルプロセッサ・システムとは異なり、マルチプロセッシング・システムは真の同時実行という特性を備えています。これは、ほとんどのレガシー・ソフトウェアが対象としていたシングルプロセッサ・システムでは問題とならなかった、数多くのプログラミング上の仮定と潜在的欠陥を露呈させます。シングルプロセッサ・システムからマルチプロセッサ・システムへのソフトウェアの移植は、1つの課題です。

複数のプロセッサ使用によるソフトウェアの速度向上は、システム内でソフトウェアを並列に実行できる量に大きく依存します。ソフトウェア内のシリアル処理とパラレル処理の本来的な比率によっては、複数プロセッサを使用しても、必ずしもすべてのアルゴリズムで著しい性能向上を実現できるわけではありません。

マルチプロセッシング・システム用のOSは、その性質からSMPまたはAMPに分類できます。SMPシステムはハードウェアの対象性に依存し、複数のプロセッサ上で対称的にソフトウェアを実行します。AMPシステムは、多数の個別ノードが集まって構成される1つのシステム内で、複数のOSを使用します。どのコンフィギュレーションを使用するかは、解決すべき問題によって異なります。AMPシステムと比較するとSMPシステムの方がプログラミングは容易ですが、いくつかの問題の性質を考慮に入れると、性能的にはAMPコンフィギュレーションの方が望ましい選択です。

VxWorks SMPは、世界をリードするこの商用RTOSに対称マルチプロセッシング機能を提供します。SMP化にあっても、VxWorksは、シングルプロセッサ・システム版と同じコアと非常に優れたリアルタイム特性を維持しています。APIの互換性や、同じWorkbench環境内の単一ツールセットによって、最小のラーニングカーブと既存のVxWorksアプリケーションの再利用が保証されます。これらのツールは、SMP環境用の新機能をサポートしています。VxWorks SMPは、シングルプロセッサ上のVxWorksと同じプラットフォーム環境内で共存できるようデザインされています。VxWorks SMPのランタイムとツール技術が、幅と深さの両方を備えた唯一のRTOSプラットフォーム・ソリューションを実現し、さらにデバイス・ソフトウェア業界最良のサポート・チームとサービス・チームがこれをバックアップします。

ウインドリバーはスマートデバイス搭載ソフトウェアの最適化 (DSO) をワールドワイドに提供するリーディングカンパニーです。企業がスマートデバイスに搭載するソフトウェアを、品質および信頼性のさらなる向上を実現しつつ、リーズナブルなコストで開発することを可能にし、早期にマーケットへ投入することを支援します。

WIND RIVER ウインドリバー株式会社

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) FAX.03-5778-6002
大阪営業所 〒532-0011 大阪市淀川区西中島7-5-25 新大阪ドイビル TEL.06-6100-5760 (代表) FAX.06-6100-5761
E-mail:info-jp@windriver.com <http://www.windriver.co.jp>

登録商標: Wind River, Wind Riverロゴ, Tornado, VxWorksは、Wind River Systems, Inc.の登録商標または商標です。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。