

# マルチプロセッシングとマルチコアのための ウィンドリバー・ソリューション

ロブ マックヤモン

## 拡大するマルチプロセッシングの利用

2つのトレンドが、デジタルデバイスにおけるマルチプロセッシングの利用を拡大させています。第一のトレンド、すなわち現在進行中の機能拡張はビジネス主導によるもので、これは、これまでデジタルデバイスの開発とは常に不可分の関係にありました。現在デバイスが提供している機能がどのようなものであっても、次世代製品は市場での競争力を維持するために、より高い機能を提供しなければなりません。このことがあらゆるアプリケーション向けデジタル・デバイスの品質証明になりつつあります。デジタル技術の進歩は、一定レベルのコスト、電力あるいはスペースの面で、ますます高い処理能力をもたらしています。この事実を活かし、性能向上や新機能追加、さらにコスト管理などによって自社の製品とサービスを改善できる企業が、今後の市場をリードしていくことになるでしょう。

第二のトレンドはマルチコア・プロセッサ・アーキテクチャで、これは比較的新しい傾向です。プロセッサの設計者は、より高性能のプロセッサを市場に投入するために、もはやクロック速度の増加に頼ることができなくなっています。代わりに複数の処理エンジン（「コア」と呼ばれる）を単一のマルチコア・プロセッサに組み込み、より高い性能を実現しています。

多くのプロジェクトにおいて開発チームは、マルチコア・プロセッサを使用することによって、マルチプロセッシングに伴う課題に直面することになるでしょう。一方それ以外のプロジェクトでは、マルチプロセッサ・システムに関する開発者の過去の経験を活かすことができます。いずれのケースにおいても、マルチコア・プロセッサを採用する場合は、性能面での利点を活かすために製品の設計と実装形態を変更する必要があります。

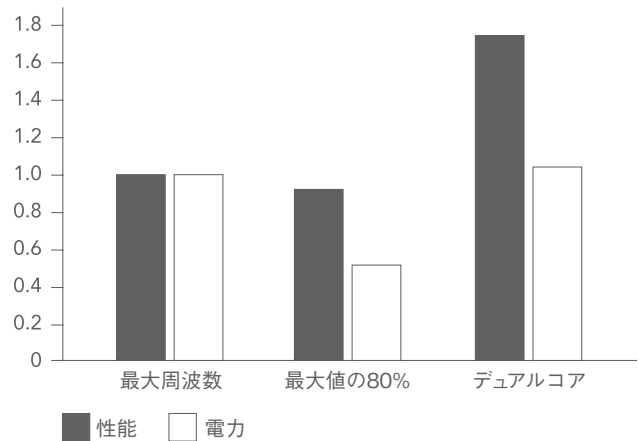


図1: マルチコア・プロセッサにより  
同じ電力消費でより高いコア性能を発揮

## 1つのプロセッサに複数のコアを置く理由

電力消費と温度に関する制約があるため、クロック周波数の増加に依存してプロセッサの性能を向上させることは現実的ではありません。マルチコア・プロセッサ・アーキテクチャは、プロセッサのクロック周波数と電圧を少し下げれば、わずかな性能低下と引き換えに大幅な電力節減が可能になるという事実を利用しています。図1は、シングルコア・プロセッサを使用した場合、同じプロセッサを20%低いクロックで使用した場合、デュアルコア・プロセッサを同じく20%低いクロック速度で使用した場合の電力と性能を比較したものです。2個のプロセッサコアを組み込み、クロック周波数を20%下げた場合は、ほぼ同レベルの電力消費で、オリジナルのシングルコアプロセッサの1.6倍のピーク性能を得ることができます。

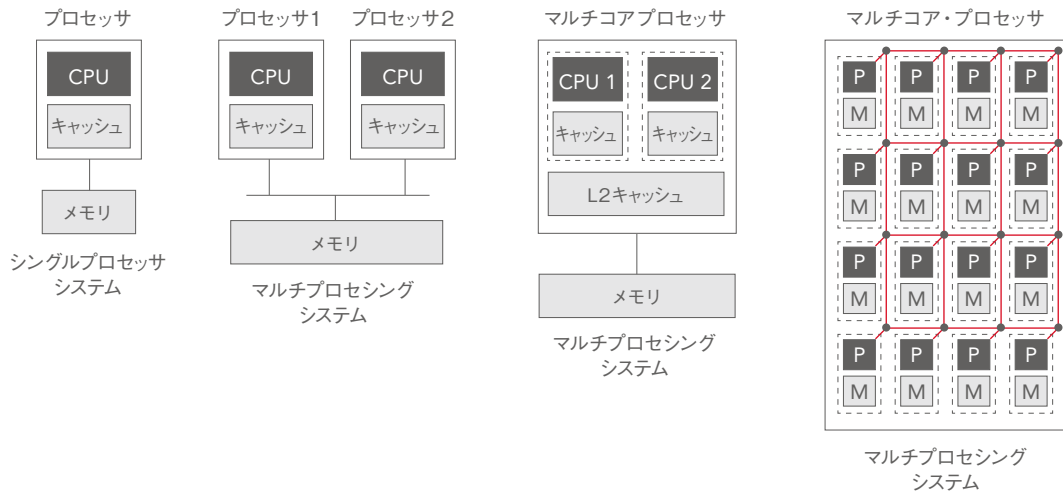


図2: マルチプロセッシングとマルチコア・プロセッサの例

### マルチコア、マルチプロセッシングという用語

本資料では、「マルチコア」および「マルチプロセッシング」という用語が繰り返し使われています。いずれも、複数の処理エンジンを使用してデジタルデバイスの処理を行うことを表しています。処理エンジンは、ソフトウェアプログラムを独立して実行できるようにするプログラムカウンタと演算器セットの組み合わせです。従来、デジタルデバイスに使用されるプロセッサの大多数は処理エンジン（「コア」と呼ばれる）が1つだけでした。したがって、タスクを完了させるために複数の処理エンジンが必要な場合には、そのシステムには複数のプロセッサが組み込まれていました。しかし近年では、1個のチップまたはパッケージに複数の処理エンジンを組み込んだマルチコア・プロセッサを利用できる機会が増えてきました。図2にシングルプロセッサ・システム、2個のプロセッサを使用したマルチプロセッシング・システム、マルチコア・プロセッサを使用した2つの異なるマルチプロセッシング・システムのサンプル図を示します。マルチコア・プロセッサを使用したデバイスはすべてマルチプロセッシングですが、すべてのマルチプロセッシング・デバイスがマルチコア・プロセッサを使用しているわけではないという点に注意してください。

### マルチプロセッシングの課題

デバイス開発者が最初にマルチプロセッシング実装を行う場合、アプリケーションの適合性、設計、開発に関する新たな問題に直面します。

マルチプロセッシングによる数多くの潜在的な利点として、機能の拡大、性能と信頼性の向上などが挙げられます。マルチコア・プロセッサ利用の背後にある最大の共通要因は、計算能力の強化によって新たに実現される機能拡大や性能向上でしょう。

一部のアプリケーションではマルチプロセッサの使用によって容易に性能を向上させることができますが、それ以外の場合、性能向上はほぼ不可能です。特定のデバイスにおけるマルチプロセッシングの利点を予測するための第一ステップは、そのアプリケーションがマルチプロセッシングにどの程度適しているのかを判断することです。マルチプロセッシングによって実現される最大性能向上率の初期分析は、アムダールの法則 (Amdahl's Law) を使用して行うことができます。この法則は、並列ではなくシリアルで行う必要のある処理の割合に基づいて、実現可能な最大速度向上率を定義します。

図3は、異なる量のシリアル処理が必要なアプリケーションと異なる数の処理エンジンを使用したシステムに対して、アムダールの法則を適用した結果を示したものです。この図からは、問題あるいは実装がシリアル処理を必要とする場合は、たとえそれがわずかな量であっても、マルチプロセッシングによって可能な性能向上率が大きく低下することが分かります。アムダールの法則を適用する際は、アプリケーションと、そのアプリケーションを実行するために使用するアルゴリズムの両方を考慮することが重要です。マルチプロセッシングに適したアプリケーションが関係するいくつかの例では、そのアプリケーションの並列性をより良く実現する新しいアルゴリズムが必要になります。

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

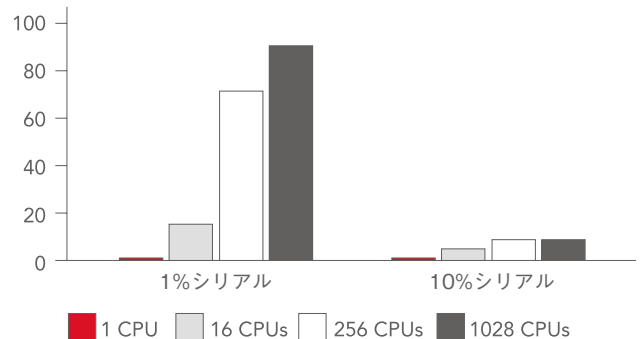


図3: アムダールの法則の適用

大量のジョブを完了させることが求められるアプリケーションはマルチプロセッシングに向いていますが、1つのジョブの完了が他のジョブの完了に依存していない、あるいは依存していたとしてもわずかなことが条件です。アプリケーションが1つの大きなジョブであってもその各部分が互いに依存しており、シリアル化が必要な度合いが高い場合は、マルチプロセッシングを行っても利点は限られたものとなります。ほとんどのデバイス・アプリケーションは、これら両極端の間に位置します。マルチプロセッシングによる潜在効果を評価するために重要な第一ステップは、並列で行う処理の量をできるだけ多くするため、前もって作業を分解する方法（機能面やデータ、マルチセッションなど）を決定することに努めることです。

次の課題は、アプリケーションの特性に適したシステム・デザインの開発に関するものです。以下の点を含め、いくつかの重要な設計上の決定が必要です。

- プロセッサの数とタイプ
- メモリシステムの設計
- プロセッサ、メモリ、I/O間の相互接続の設計
- システムのマルチプロセッシング・アーキテクチャ(対称型マルチプロセッシング、非対称型マルチプロセッシング等)
- ソフトウェア・アプリケーションの分割方法(プロセス、スレッド等)
- 個々の処理エンジンに対するハードウェア・リソースのマッピング(メモリ、I/O、割り込み)
- 分割されたソフトウェア・ワークロードのハードウェア・リソースへのマッピング

これらの各事項は、システムとそのアプリケーションを総合的に理解した上で決定する必要があります。アプリケーションの特性とハードウェアおよびソフトウェアの構造をうまく合わせることができないと、システム性能の低下を招き、システムが複雑化してソフトウェア開発努力の成果を脅かすことになります。

第三の課題は、デバイスを作動させるソフトウェアの開発に関することです。クロック速度の増加に基づく過去の性能向上とは異なり、マルチプロセッシングを使用して性能を大幅に向上させるという行為は、デバイスソフトウェア開発に大きな影響を与えます。マルチプロセッシング・ソフトウェアの開発に関する一般的な課題をいくつか以下に挙げます。

- ソフトウェアの実装開始前に開発される優れたマルチプロセッシング・システムとソフトウェアの設計に関する作業
- 使用可能なすべてのプロセッサを使用するために十分な同時実行を実現するソフトウェアの分割と、将来、より多くのプロセッサが追加された場合のスケール・アップ
- 複数のタスク、スレッド、プロセスの同時実行を可能にする環境に適したソフトウェアの開発とデバッグの習得
- 競合状態およびデッドロックの検出と是正
- 性能の評価と、ボトルネックおよび未使用プロセッササイクルをなくすことによる最適化
- 十分な性能を持つ信頼できるプロセス間通信とプロセッサ間通信の確立
- 分割されたソフトウェアシステムに対するプロセッサその他のハードウェアリソースの割り当ての最適化
- 共有データと非共有データの適切なバランスの実現
- マルチプロセッシング環境で使用する既存の社内開発ソフトウェア、市販ソフトウェア、またはオープンソース・ソフトウェアの準備。これには以下の作業が必要となります。
  - 同時実行の実現
  - マルチプロセッシングに関連する欠点や性能上の問題の排除

## 競合状態

複数の実行スレッドが同一のデータにアクセスする場合、ソフトウェア内に競合状態が生じる可能性があります。ソフトウェアの実行が特定イベントの発生順序に依存している場合に、競合状態が発生します。競合状態が発生すると、通常、変数が不正な値に設定される結果となります。競合状態の反応はタイミングによって異なるので、その検出とデバッグには困難が伴います。シングルプロセッサ・システムの場合は問題とならなかったような競合状態でも、マルチプロセッサ・システムでは障害となる可能性ははるかに高くなります。

初めてマルチプロセッシングに取り組む開発チームは、これらの課題に対応できる態勢を整えておかなければなりません。十分な経験を積んだマルチプロセッシング・チームでも、これらの問題に遭遇することがあります。例えば、複数のシングルコア・プロセッサを使用したマルチプロセッシング・システムでは良好に作動した設計であっても、性能特性の異なるマルチコア・プロセッサでは設計を変更しなければならないことがあります。マルチプロセッシング・プロジェクト用の新しいソフトウェアの開発は、シングルプロセッサ・システムの場合よりも多くの課題が伴います。ですから、マルチプロセッシングに関する開発チームの過去の経験は、大きな財産となります。

## Device Software Optimization

DSO (Device Software Optimization) は、より迅速に効率よく低コストで、しかもより信頼性の高い方法で、企業がデバイスソフトウェアを開発、実行、管理することを可能にする戦略です。マルチプロセッシング・デバイスの開発は、通常、シングルコア・プロセッサを使用したデバイスの開発よりも複雑でコストがかかり、しかも高いリスクが伴うので、DSO戦略はより多くの価値を提供します。この戦略には以下の4つの要素が含まれています。

- 標準化
- 選択と柔軟性
- パートナーとの関係
- 業界最良事例の採用とその堅持

## マルチプロセッシングのためのウインドリバー・ソリューション

この項では、マルチプロセッシング・デバイス開発チームに対するウインドリバーのサポート機能について、その概要をまとめて示します。ここに述べる機能の一部は現在開発中で、その具体的な詳細は今後決定されます\*。

開発者がDSOの利点を実現するための支援を行うアプリケーション用プラットフォームという形で、ウインドリバーは技術的ソリューションを提供します。5つの産業分野と数多くの市場におけるデジタル・デバイスへの幅広い要求に対応するために、多くのプラットフォームが用意されています。現在ウインドリバーのプラットフォームが対応している産業には、自動車用エレクトロニクス、コンシューマ・デバイス、産業用および医療用機器、ネットワーク機器、航空宇宙用および防衛用アプリケーションなどがあります。

それぞれのウインドリバー・プラットフォームには、適切なプロセッサやボードに組み込まれたデバイスOSとミドルウェア、そしてハードウェアの立ち上げからシステムテストに至るまでのデバイスソフトウェア開発プロセスのすべての側面に対応した、完全な開発環境 (Wind River Workbench) が含まれています。これらのプラットフォームは、マルチプロセッシング・デバイスのソフトウェア開発をあらゆる側面で支援します。

## OSの選択と柔軟性

ウインドリバーは、VxWorksリアルタイムOS (RTOS) とWind River Linuxに基づくマルチプロセッシング対応プラットフォームを提供します。このプラットフォームを使用すれば、ユーザはアプリケーションに合った最良のOSを選択したり、マルチプロセッシング・デバイス内でVxWorksとLinuxを組み合わせて使用することができます。開発チームがマルチプロセッシング・デバイス用のさまざまなOSのオプションを評価する際に支援するのは、ウインドリバーだけです。

## マルチプロセッシング・システムアーキテクチャの選択

マルチプロセッシング用システムアーキテクチャの中でハードウェアとソフトウェアを結合する方法は数多くあります。この目的のための2つの主要なアーキテクチャが、対称型マルチプロセッシング (SMP) と非対称型マルチプロセッシング (AMP) です。

図4はSMPシステムの例で、このシステムには以下のような特性があります。

- すべてのプロセッサ(またはコア)が同じ
- すべてのメモリが共有されており、すべてのプロセッサによるメモリアクセスが一樣
- 1つのSMP OSが、すべてのプロセッサ、割り込み、アプリケーションを管理
- そのOS環境内では複数のスレッドが同時に実行されるが、これはシングルプロセッサ・システムでは不可能
- OSがプロセッサにスレッドを割り当て
- OSがプロセッサ間のワークロード・バランスを取ることを試行する
- CPUアフィニティがサポートされている場合、OSによってタスク/スレッドを特定のプロセッサに割り当て、常にそこで実行するようにスケジューリングすることができる
- このシステムは、複数のスレッドが頻繁にデータを共有する必要があるようなアプリケーションに適している
- このシステムでは、マルチプロセッサの性能を活かすためにアプリケーションをマルチスレッド化する必要がある

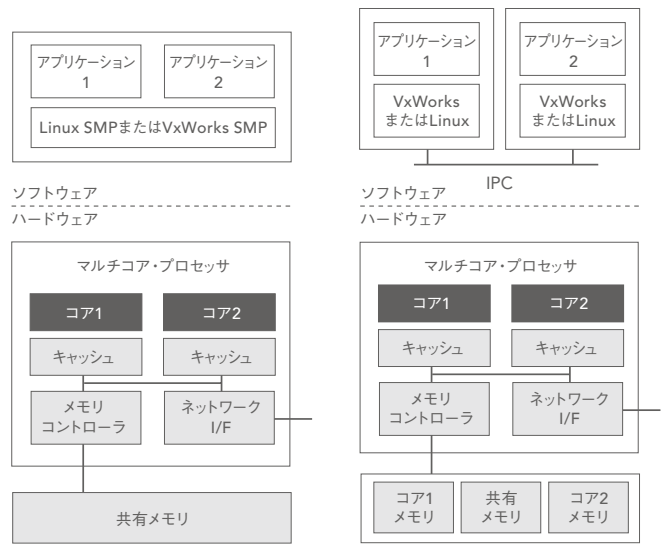


図4: 対称型マルチプロセッシング (SMP) システム

図5: 非対称型マルチプロセッシング (AMP) システム

図5はAMPシステムの例で、このシステムには以下の特性があります。

- すべてのプロセッサ(またはコア)を同じにする必要はないが、同じにすることも可能
- すべてのプロセッサは、ある程度の非共有のローカルメモリを持っている
- すべてのプロセッサは、OS、実行型オブジェクト、またはスタンドアロン・ソフトウェア・アプリケーションのインスタンスを個別に独立して実行する
- すべてのOSを同じにすることができるが、必ずしも同じにする必要はない。例えば、AMP内の異なるプロセッサ上でVxWorksとLinuxの両方を実行することができる
- プロセス間通信 (IPC) が提供される。これはプロセッサ間でのデータ転送を可能にし、動作の同期を可能にする
- サブシステムを特定のプロセッサに割り当てると共に、設計時にシステム・ワークロードを分割する必要がある
- このシステムは、サブシステム間のデータ共有と相互動作の必要性が比較的限定されたアプリケーションに適している
- 各アプリケーションはシングルプロセッサ・サブシステム上で実行できるので、このシステムではアプリケーションをマルチスレッド化する必要はない。複数のプロセッサの動作の調整と結合にはIPCが使われる

AMPアーキテクチャとSMPアーキテクチャは、異なるデバイスおよびアプリケーションのニーズに適します。設計チームが複数のプロセッサを使ってアプリケーションの速度向上を図る方法を検討するプロセスを開始する場合は、両方のアーキテクチャを使用できるという点に留意する必要があります。システムによってはSMPとAMPを組み合わせて使用する場合もあります。例えば、4個のマルチコア・プロセッサが搭載された1枚のボード上で、ボード上の4個のプロセッサはAMPコンフィギュレーションで結合する一方、各プロセッサのすべてのコアでSMP OSを実行することができます。

\*上記の内容はウインドリバー製品の一般的な方向性の概要を示すことを意図したものです。これはあくまで参考のために示したに過ぎず、何らかの契約に組み入れられたり、購入判断の根拠とすることができものではありません。ウインドリバー製品について示した特性あるいは機能の開発と発売、そのタイミングは、ウインドリバー独自の判断に基づいて決定されます。

## Linuxマルチプロセッシング

Linuxベースのデバイスに関係する重要なマルチプロセッシング・アーキテクチャの中で、最も多く使われるのが対称型マルチプロセッシングです。その理由はさまざまですが、最も大きな点は、既存の企業／ITシステムの多くがSMPをサポートしていることと、多くのLinuxアプリケーションがSMPに対応しているということです。Wind River Linuxプラットフォームには、最初のリリースから対称型マルチプロセッシングのサポートが含まれていました。Wind River Linuxに付属しているTCP/IPスタックは、複数のプロセッサを使用してネットワーク性能を向上させるために、SMP用に最適化されています。ウインドリバーのLinuxディストリビューションの他のすべての面と同様、SMPサポートは、デバイス・ソフトウェア開発用の汎用プラットフォーム・ソリューションの一部として提供されています。

Wind River Linuxプラットフォームは、非対称型マルチプロセッシングもサポートしています。AMPシステムにおいては、Wind River Linuxの複数のコピーを互いに組み合わせたり、VxWorksのような別のOSと組み合わせたりすることができます。AMPには、複数プロセッサ間の情報伝達や動作の同期に使用できるIPCが必要です。Wind River Linuxはプロセッサ間のEthernet接続を使用してIPCをサポートしていますが、これは複数のプロセッサを使用してマルチプロセッシング・システムを構成する一般的な方法です。マルチコア・プロセッサ内でAMPを実装する時は、IPCをサポートするために、より頻繁に共有メモリが使われます。ウインドリバーは現在、Linux用に共有メモリベースのIPCサポートを開発中です\*。ウインドリバーによる共有メモリサポートの詳細については、以下の「プロセッサ間通信のビルディングブロック」に示します。

## VxWorksマルチプロセッシング

ウインドリバーのVxWorksは、長年にわたってマルチプロセッシング・デバイスの作成に使われています。VxWorksを使用するシステムは、ネットワーキング、産業／医療、航空宇宙、防衛を含め、多くの産業分野のさまざまなアプリケーションを支えてきました。これらのデバイスは、すべてAMPアーキテクチャを使用して作成されています。VxWorksのプロセッサ間通信機能は、最も一般的に使われる共有メモリとEthernetを含め、物理的転送方法の選択肢を提供します。結果として現在のVxWorksは、マルチコア・プロセッサまたはマルチプロセッサどちらによるAMPシステムにも使用することができます。VxMPと呼ばれるオプションのランタイム・コンポーネントは、プロセッサ境界を越えて使用できる共有のメッセージキューとセマフォを提供するので、これらの共通OS機能を使用して複数タスクの調整を行うことができます。タスクが同じプロセッサ上で実行されているか否かは問いません。

最も広く使われているリアルタイムOSであるVxWorksは、SMPに適したアプリケーションをマルチコア・プロセッサに移行する際に使用されます。VxWorks SMPは現在開発中ですが、デバイス開発者の選択範囲を広げ、柔軟性を高める存在となるでしょう\*。

Linux			VxWorks	
Linux SMP	Linux AMP	混用 AMP	VxWorks AMP	VxWorks SMP

図6：マルチプロセッシング・システムのオプション

図6は、VxWorksとLinuxを使用して実現できる5つのマルチプロセッシング・コンフィギュレーションを示したものです。ウインドリバーは現在のうちの4つをサポートしており、5つめ(VxWorks SMP)のサポートに向けて開発を進めています\*。

## SMPでもAMP?

AMPおよびSMPという略語は、マルチプロセッシング・ハードウェアの設計と、マルチプロセッシング・ソフトウェア・アプリケーションをハードウェアにマップする方法の両方を表すために使われます。このことは、ある種の混乱を引き起こす可能性があります。ハードウェアの領域では、対称型マルチプロセッサとは、すべての処理要素が同一である設計のこと。一方、ソフトウェアの領域では、対称型マルチプロセッシングとはOSによってサポートされる機能を指します。その仕事は、複数のプロセッサをOSの単一インスタンスによって管理することです。このインスタンスは、アプリケーションを構成するプロセスやスレッドを個々のプロセッサに割り当てる役割を果たします。対称型マルチプロセッシングは対称型マルチプロセッサ上でしか実現できません。しかし、対称型マルチプロセッサ・ハードウェアを使用して非対称型マルチプロセッシングを実現することは可能です。さらにややこしいことには、ハードウェアの領域では非対称型マルチプロセッサというものも存在します。これらのプロセッサと対称型マルチプロセッサの違いは、非対称型マルチプロセッサには複数のタイプの処理要素が含まれているという点です。したがって、ソフトウェア開発者は次の点に留意する必要があります。すなわち、誰かがSMPあるいはAMPと言った場合は、それがプロセッサ・ハードウェア設計のことなのか、システムソフトウェア・コンフィギュレーションのことなのか、あるいはその両方のことなのかを、間違いないように確認することが必要です。

## プロセッサ通信のビルディングブロック

ウインドリバーは、マルチプロセッサ・デバイスに必要なIPCの作成用に、数多くのビルディングブロックを提供しています。これには以下のものが含まれます。

- AMPシステム内におけるメッセージ受け渡しのための豊富なネットワークサポート
- ネットワークベースIPCに代わるものとして必要な手段を提供する共有メモリ通信。特にマルチコア・プロセッサに関連
  - 現在はVxWorksを実行するプロセッサに使用可能
  - 将来Wind River Linuxで使用可能なものを開発中\*

\*上記の内容はウインドリバー製品の一般的な方向性の概要を示すことを意図したものです。これはあくまで参考のために示したに過ぎず、何らかの契約に組み入れられたり、購入判断の根拠としたたりできるものではありません。ウインドリバー製品について示した特性あるいは機能の開発と発売、そのタイミングは、ウインドリバー独自の判断に基づいて決定されます。

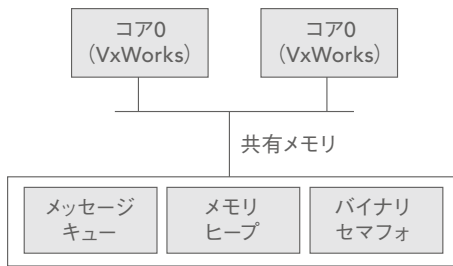


図7: 現場で実証されたVxMPによるAMPサポート

- VxMPは、AMPシステムで使用するための分散 VxWorksオブジェクトを提供します(図7参照)。
  - VxMPは、標準のVxWorksセマフォ、メッセージキュー、メモリ分割を使用して、異なるプロセッサ上で実行中のタスクを連携させることができます。
  - VxMPは、マルチプロセッサ環境およびマルチコア・プロセッサ環境での使用に適しています。
  - 現在、将来のリリースを目標に共有スピンロックおよび共有アトミックメモリ演算子を含む追加機能を開発中です\*
- VxWorksとLinux向け透過プロセス間通信(TIPC)のサポートは、ソケットベースIPC用の理想的なプロトコルを提供します。

ウインドリバーはTIPCの開発と使用を積極的にサポートしています。それはTIPCがマルチプロセッシング・システム用のIPCメカニズムとして他に例を見ない利点を提供するからです。図8にTIPCクラスタの例を示します。

TIPCの利点としては、以下のようなものがあります。

- TCPやUDPと同じソケットAPIを使用しますが、TCPやUDPの場合よりもはるかに小さく、待ち時間も短縮されます。
- 物理的なネットワーク・レイアウトを設計者から見えないようにすることによってマルチプロセッサ・アプリケーションの設計を簡略化するという、透過的なアドレッシング・メカニズムを備えています。
- リコンフィギュレーションや新しいプロセッサの追加によってネットワークに変更が生じた場合でも、通常、ソフトウェアを大幅に作り直す必要はありません。
- マルチプロセッサ・システムをスタートアップする際に必要なアプリケーションによる同期問題への対応を、ネットワーク・トポロジ・メカニズムが容易にします。
- プロセッサや通信リンクの障害に対処するために容易に変更を加えることができます。
- 同一プロセッサ上のプロセス間、同一プロセッサ内のプロセッサコア間、同一ボード上/バックプレーン間/ネットワーク上の複数プロセッサ間のIPCをサポートしています。
- デュアルライセンス(BDS、GPL)のオープンソース技術。
- ロードマップはMulticore Association TIPC Working Groupが管理。
- ベアラは、Ethernetや共有メモリを含むさまざまな物理的相互接続方法によるTIPCの使用を可能にします。
- OSやハードウェアに依存しません。
- 現在はVxWorksやLinuxを含む複数のOSをサポートしています。

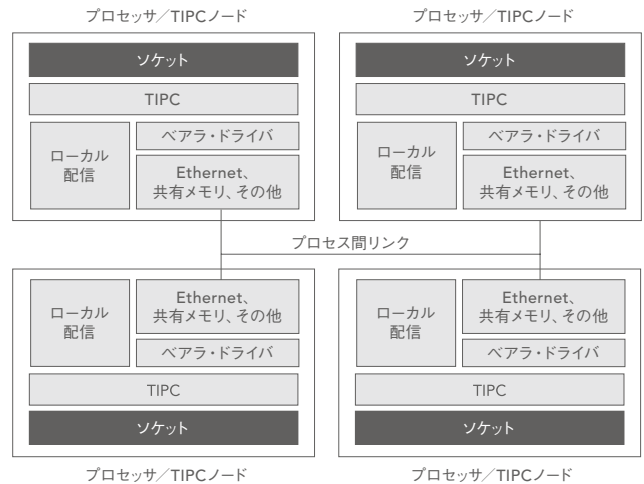


図8: 透過プロセス間通信(TIPC)

### ハードウェアの統合

ドライバ、ボードサポート、OSを今日の複雑なマルチコア・プロセッサと統合することは容易ではありません。しかし、ウインドリバーのアプリケーション用プラットフォームは、さまざまなプロセッサ、チップセット、周辺機器、ボードとあらかじめ統合されています。さらに、ウインドリバーのプロフェッショナル・サービス(受託開発)やパートナーがその他のハードウェアサポートを提供しています。広範なハードウェア用のマルチプロセッシング・ソリューションを提供するウインドリバーの機能は、要求時にサポートを提供する機能とオフザシェルフサポートを組み合わせることによって、顧客に製品市場投入までの時間を大幅に短縮するという利点を提供します。

### 開発ツール

マルチプロセッシングを有効に利用するソフトウェア・アプリケーションがなければ、マルチプロセッシング・プラットフォームの真価を発揮することはほとんどできません。図9に示すように、WindRiverWorkbench開発環境はEclipseフレームワークをベースにし、幅広い機能を統合します。Workbench開発環境とこれに付属するツールは、マルチプロセッシング・デバイス用ソフトウェア開発の課題を解決する幅広い機能を提供します。

\*上記の内容はウインドリバー製品の一般的な方向性の概要を示すことを意図したものです。これはあくまで参考のために示したに過ぎず、何らかの契約に組み入れられたり、購入判断の根拠としたりできるものではありません。ウインドリバー製品について示した特性あるいは機能の開発と発売、そのタイミングは、ウインドリバー独自の判断に基づいて決定されます。

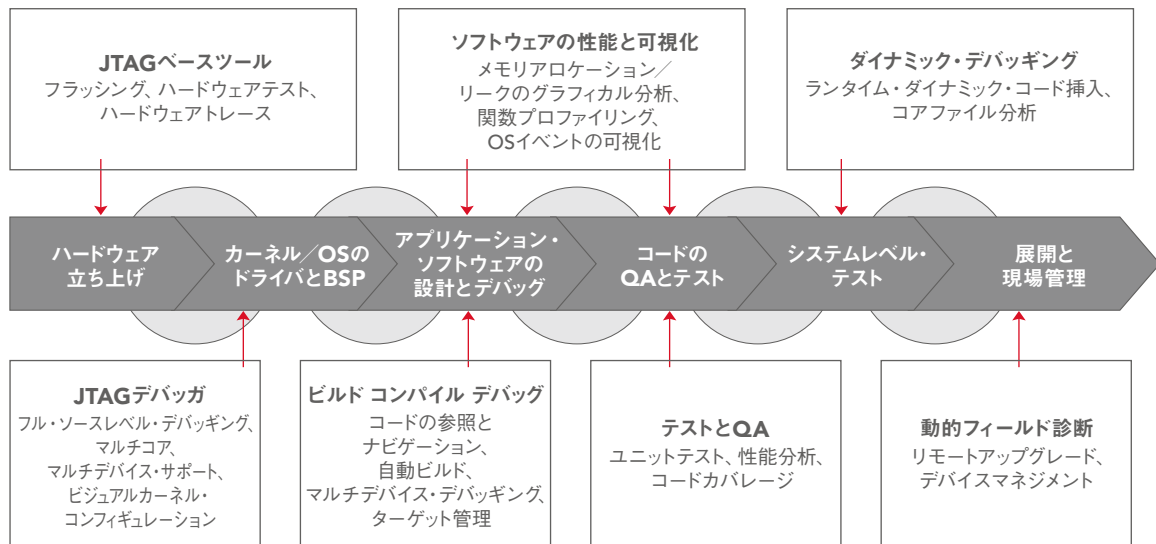


図9: Wind River ワークベンチ

## 開発の課題とソリューション

### 複数プロセッサの接続と制御

マルチプロセッシングにおいては、複数のプロセッサやプロセッサ・コアを物理的に制御するための手段が必要です。Wind River Workbenchには、マルチプロセッサ・システムを調整するためのターゲットマネージャが含まれています。

デバッガやその他の開発ツールを有効なものとするためには、それらのツールを対象プロセッサに接続できるようにしなければなりません。多くのマルチプロセッサ・システムでは、プロセッサが開発者のデスクトップコンピュータと同じネットワーク上にないこともあります。デバッグ時に対象プロセッサが開発者のデスクトップと同じネットワーク上にないが、ネットワークを共有している中間プロセッサを通じて接続が可能な場合は、Workbench Debuggerに含まれるプロキシを使用した接続が可能で、必要であればプロトコルを変換することもできます(図10参照)。

オンチップデバッグを使用する場合は、1つのWind River ICEを使用して複数のプロセッサまたはプロセッサコアを制御することができます。他の多くのオンチップ・デバッグ・ソリューションでは、プロセッサまたはコア1つごとに別体のデバッグ・ハードウェアユニットが必要となり、コストが大幅に上昇して複雑さも増します。ウインドリバーがサポートするマルチプロセッサ・オンチップ・デバッグには複数プロセッサ間の同期実行制御も含まれていますが、基礎となるハードウェアがこの機能をサポートしている必要があります。

### 負荷の大きいプロセッサと小さいプロセッサの特定

ウインドリバーでは現在、Wind River System Viewerのマルチプロセッシング専用機能を強化する開発作業を新しく行っています\*。例えば、SystemViewerは、時間の経過に伴う各プロセッサ・コアの利用状況やペンディングタスクの数を表示することが可能。SMPシステムの場合は、すべてのプロセッサ・コアのデータが1つの時間相関ディスプレイにまとめて表示されます。

さらに、VxWorks Spy()コマンドを使用すれば、各タスクによるCPU使用時間、割り込みレベルとカーネルによる使用時間、CPUの総アイドル時間を知ることができます。

### 競合状態の特定と診断

Wind River System Viewerは、ユーザ定義イベントの表示をサポートしています。開発者は、適切なイベントをファイルに記録することによって、競合状態が疑われるイベント・シーケンスを特定し、スクリプトまたはプログラムを使用してそのファイル内のデータを分析することができます。

### 複数プロセッサ間のスレッドの相互動作のデバッグ

Workbench Debuggerは複数のコンテキストで同時に動作し、ソース/アセンブリ・ビュー、レジスタ・ウィンドウ、スタックのバックトレース・ビューを含む自身のデバッグ・コンテキストに各スレッドを割り当てます。各スレッドはスレッド限定のブレークポイントを使用して個別に制御することもできるので、複数プロセッサの相互動作に関する開発者のデバッグ機能がさらに向上します。



図10: マルチコア・デバッグにおけるデバッガ・プロキシ・エージェントの使用

\*上記の内容はウインドリバー製品の一般的な方向性の概要を示すことを意図したものです。これはあくまで参考のために示したに過ぎず、何らかの契約に組み入れられたり、購入判断の根拠としたりできるものではありません。ウインドリバー製品について示した特性あるいは機能の開発と発売、そのタイミングは、ウインドリバー独自の判断に基づいて決定されます。

## ブレイクポイントを使わない複数プロセッサのデバッグ

AMPシステムでは、問題によってはブレイクポイントを使ったデバッグができないことがあります。いくつかあるプロセッサの中の1つを停止すると、システムに異常が発生したりシステムの動きが大きく変わったりして、解決しようとしている問題が隠されてしまう可能性があるからです。Workbench Diagnosticsは実行中のアプリケーションと動的に連携させることができます。つまり、診断コードを追加してデータを収集し、プログラムの実行を修正。プロセッサを停止したりアプリケーション・コードをリビルドしたりすることなくこれらの機能を利用できるWorkbench Diagnosticsは、AMPシステムへの干渉が従来のデバッグよりもはるかに少ないソリューションです。

## システム全般にわたるデータ値追跡

マルチプロセッサ・システムの動作を理解するには、多くの場合、複数プロセッサの相互動作によって生じるシステム状態の変化を理解し、時間の経過に伴うデータ値の変化を追跡できるようにする必要があります。Data Monitor (旧StethoScope)を使用すれば、SMPシステムの重要なデータ値や、AMPシステムの複数プロセッサ間で共有されるデータの値を容易に追跡することができます。

Workbenchのターゲット・エージェントとハードウェア・ブレイクポイント機能によるプロセッサのオンチップ・デバッグ・サポートは、もう1つのデータ値追跡方法を提供します。これは、指定アドレスでのデータアクセスをトリガするブレイクポイントを設定することによって行います。

## 速度向上に最も適した機能の特定

並列性を利用することによる性能向上は、プログラムのうちで最もCPUサイクルを消費する部分から開始します。ウインドリバーのLinuxおよびVxWorks用開発環境は、この情報を収集するいくつかの方法を提供します。先に述べたVxWorks Spy()機能と、Linux環境で使用できるトップコマンドはその一例です。Performance Profiler (旧 ProfileScope)とSystem Viewerを使用すれば、アプリケーションのどの機能が最もCPUサイクルを消費しているかを容易に特定することができます。SMPシステムでは、スレディングの追加がシステム性能向上の助けとなります。AMPシステムにおける同様の機能としては、タスク分解や負荷の小さいプロセッサへのタスク再割り当てによって速度を向上させる機能があります。

## 性能を低下させるリソース競合の特定

Wind River System Viewerの豊富なデータ収集機能を使用すれば、事後分析によってさまざまなリソース競合関連問題を明らかにすることができる情報を記録することが可能です。

## プロセッサ間通信と同期の追跡

マルチプロセッサ・システムにおけるプロセッサ間の相互動作は、少なくとも単一プロセッサ内で何が起きているのかということと同程度に重要です。通常、マルチプロセッシング・システムの同期にはスピンロックが使われます。スピンロックはVxWorks SMPでサポートされる予定であり<sup>\*</sup>、スピンロック動作はSystem Viewerの表示によって追跡が可能になります。System ViewerによるTIPCメッセージの計測によっても、プロセッサ間の相互動作についての観察が可能です。

## プロセッサへの割り込みの静的または動的ルーティングの表示

割り込みを含むハードウェアリソースを特定のプロセッサに割り当てることは、マルチプロセッシング設計の重要な決定要素です。VxWorksカーネルオブジェクトの状態に関するクエリ機能を利用すれば、開発者は、割り込みがプロセッサに対してどのように割り当てられているのかの確認が可能。この機能は、静的割り当てと動的割り当てに利用できます。

## プロフェッショナル・サービスを利用したソリューションの完成、拡張、速度向上

マルチプロセッシング・デバイスの開発努力を成功裏に完了させるには、一連のオプションを検討して新たなスキルや経験を獲得する必要があります。ウインドリバーのプロフェッショナル・サービス(受託開発)はさまざまな方法によってこのプロセスを支援します。例えば、プロフェッショナル・サービスを利用すれば、ウインドリバーのエンジニアリング部が現在開発中の技術にアクセスすることができます。エンジニアリング部のプロジェクトには、さまざまなユーザのニーズに対応する幅広いハードウェア・コンフィギュレーションとソフトウェア・コンフィギュレーションのサポートが含まれています。プロフェッショナル・サービスを利用すれば、ウインドリバー・プラットフォームの予定リリース日よりもかなり早い時期に、新しい機能(特定のプロジェクトに必要とされる特定のコンフィギュレーションで)入手することができます。

ウインドリバー・プロフェッショナル・サービスは、ウインドリバーの既成製品によってサポートされていないプロセッサやボード関連のウインドリバー・プラットフォームを提供することによって、アプリケーション開発に要するユーザの労力を軽減します。これは、プロジェクトに要する時間を短縮してプロジェクト・チームの注力を増加させます。また、マルチプロセッシング・ハードウェアのサポートが非常に複雑なものであるという点から見て特に有効です。

多くの場合、既存のソフトウェア・アプリケーションは、マルチプロセッシング環境での使用に十分対応できるものではありません。例えば、シングルプロセッサ環境で開発とテストが行われたソフトウェアをマルチプロセッシング・システムで実行するには、普通、変更を加えなければなりません。ソフトウェアが正しく動作したとしても、そのソフトウェアが性能向上を実現できるような形で複数のプロセッサを使用できるようにするには変更が必要です。あるプロジェクト用のソフトウェアが社内開発されたものか、オープンソースによるものか、あるいは商業プロバイダによるものかを問わず、ウインドリバー・プロフェッショナル・サービスは、マルチプロセッシング・システムの性能を向上させるような形で、そのソフトウェアを正しく実行できるようにすることができます。

\*上記の内容はウインドリバー製品の一般的な方向性の概要を示すことを意図したものです。これはあくまで参考のために示したに過ぎず、何らかの契約に組み入れられたり、購入判断の根拠としたりできるものではありません。ウインドリバー製品について示した特性あるいは機能の開発と発売、そのタイミングは、ウインドリバー独自の判断に基づいて決定されます。

## マルチプロセッシングおよびマルチコアへのDSOの適用

マルチコア・プロセッサ・アーキテクチャの出現、そしてそれに伴うデバイス・ソフトウェア開発におけるマルチプロセッシング利用の拡大によって複雑さとリスクが増大し、さらにそのことによってDSO戦略の利点が強調されるようになりました。マルチプロセッシングを必要とするデバイスソフトウェア開発という背景の中で、以下に示すようなDSO原則の適用について考えてみましょう。

### 標準化

企業全体に共通する技術、ツール、プロセスを使用して効率を向上させ、資源の活用と再利用を促進します。

- 新しい技術、ツール、プロセスが、マルチプロセッシング・デバイスソフトウェアに関する課題への開発チームの対応を支援します。
- 開発チームはオプションの評価に力を集中することができ、選択したアプローチの使用を標準化することができます。
- このような前提がないと、さまざまな選択肢が統一されことなくバラバラに開発プロジェクトに使われることになり、下流部分の非効率を招く結果となります(例えば、開発者がさまざまなプロジェクトで異なるツールや技術、あるいはプロセスを使用すると、トレーニングが非効率的になり生産性が失われます)。

柔軟性を保ち選択の余地を残すためのオープンスタンダードの採用。

- オープンスタンダードのインターフェースと技術を最大限に使用すれば、統合化のコストを減らすことができ、実装をより良いものに置き換えることが容易になります。
- ウィンドリバーは、TIPCのようなマルチプロセッシング標準の開発と使用を積極的にサポートしています。

### 選択と柔軟性を可能にする

OSを選択できれば、より適切なソリューションを得ることができます。

- ウィンドリバーは幅広いマルチプロセッシング・オプションを提供することによって、ユーザがそのアプリケーションに最も適した方法を見つけることができるようにします。
- ウィンドリバーは、VxWorksとLinux、AMPとSMP、さまざまなプロセッサ間通信機能をサポートしています。

### パートナーとの提携

強力なパートナー・エコシステムがより完全なソリューションを提供し、製品市場投入までの時間短縮、開発コストや開発リスクの低減を実現します。

- さまざまなプロセッサや商用既成ボードを含むデバイスハードウェアの広く深いサポート。
- ハードウェアおよびソフトウェアのパートナーが、幅広いカテゴリのウィンドリバー・プラットフォーム用に、統合化された拡張機能を提供します。

### 業界最良事例の採用

知的資産を再利用し、使用可能なサービスやサポートを活用します。

- 商用ソフトウェアコンポーネントがマルチプロセッシングの利点を生かせば、その価値は増大します。
- ウィンドリバーのアプリケーション用プラットフォームは再利用可能な幅広いマルチプロセッシング対応のIPを提供しており、その範囲は広がっています。
- ウィンドリバーは世界中で製品のサポートとサービスを提供しており、これには、マルチプロセッシング・システムの開発に伴う複雑な課題への特別なサポートも含まれています。

### 最初の一步

ウィンドリバーは比類のない柔軟性、機能、サポートをユーザに提供し、マルチプロセッシング・デバイスのDSOに関する課題の解決を支援します。マルチプロセッシングのあらゆる問題に対応できるオールマイティなソリューションというものは存在しません。マルチプロセッシング技術探求の最良のスタート点は、ユーザ自身のアプリケーションとその特性にあります。ウィンドリバーにご連絡の上、問題解決のための話し合いを始められることをお勧めします。ウィンドリバーは、お客様のニーズに合った最適のソリューション発見のお手伝いができることを望んでいます。

ウィ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 ウィ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。