WP474 (v1.0) 2016 年 3 月 31 日

Xen ハイパーバイザーと Zynq UltraScale+ MPSoC を使用した仮想化の実現

Zynq® UltraScale+™ MPSoC 上で Xen ハイパーバイザーを使用すると、ハードウェア アクセラレーションを使用した仮想化と使いやすさが実現するため、エンベデッド システム設計者はハードウェア投資を最大活用することができます。

概要

仮想化はデスクトップ システムにとって欠かせない存在ですが、SoC システムの使用率と性能を最適化する必要のあるエンベデッド システム設計者にとっては、長らく、複雑な問題でした。

これまで、エンベデッド領域に仮想化を導入することが困難であったのは、十分な性能を実現しながらも、ソリューションを容易に実装できる適切なハードウェア リソースが欠如していたためです。したがって、ヘテロジニアス ソフトウェア スタックを同じプロセッサで実行する必要がある場合、各種のソフトウェア スタックを手動で管理するか、またはアクセラレーション機能のない仮想化が持つ、高レイテンシで低性能な特性を受け入れるかしかありませんでした。

Zynq UltraScale+ MPSoC の中核をなす ARM®v8 アーキテクチャは、ハードウェア アクセラレーション機能を利用した仮想化を実現することで、このような実装上の障害を軽減します。さらに、Xen ハイパーバイザーによって、デスクトップ クラスの性能と生産性をエンベデッド システムで実現するための土台となる使いやすさが提供されます。Xen ハイパーバイザーを Zynq UltraScale+ MPSoC で実行すると、エンベデッド システム設計が持つ可能性を最大限に引き出せる完全なソリューションが実現します。

はじめに

仮想化は、複数のソフトウェア スタックを同一プロセッサで同時実行できるようにすることで、デスクトップに広く普及しており、今後はエンベデッド システムにも広がる勢いを見せています。デスクトップ ユーザーの間に仮想化を定着させたのは、VMWare や VirtualBox などの定番パッケージです。これらのソフトウェアはデスクトップ コンピューターの生産性向上ツールと見なされていますが、同じ原則を適用することで、Xilinx Zynq UltraScale+ MPSoC などのエンベデッド システム オン チップ (SoC) を最大活用することができます。

仮想化が果たす役割はシステムによってそれぞれ異なります。あるシステムでは、仮想化を使用してプロセッサの使用率を常に高く維持することで、消費電力の削減と性能の最大化を実現しています。また別のシステムでは、独立性や冗長性を確保するために各種のソフトウェア スタックを分割するための手段として、仮想化を利用しています。

これまで、エンベデッド領域に仮想化を導入することが困難であったのは、十分な性能を実現しながらも、ソリューションを容易に実装できる適切なハードウェア リソースが欠如していたためです。したがって、ヘテロジニアス ソフトウェア スタックを同じプロセッサで実行する必要がある場合、各種のソフトウェア スタックを手動で管理するか、またはアクセラレーション機能のない仮想化が持つ、高レイテンシで低性能な特性を受け入れるかしかありませんでした。

Zynq UltraScale+ MPSoC の中核をなす ARMv8 アーキテクチャは、ハードウェア アクセラレーション機能を利用した仮想化を実現することで、このような実装上の障害を軽減します。さらに、Xen ハイパーバイザーによって、デスクトップ クラスの性能と生産性をエンベデッド システムで実現するための土台となる使いやすさが提供されます。Xen ハイパーバイザーを Zynq UltraScale+ MPSoC で実行すると、エンベデッド システム設計が持つ可能性を最大限に引き出せる完全なソリューションが実現します。

エンベデッド システムを仮想化する理由

仮想化するかどうかの判断は、通常、以下の 3 つのシステム設計特性によって左右されます。

- パフォーマンス基準を満たすため、プロセッサの使用率をできるだけ高く維持する必要がある
- 安全性、スケーラビリティ、信頼性を確保するため、ソフトウェアの分離または分割が必要である
- 信頼性要件を満たすため、スケーリングまたは冗長化が必要である

Zynq UltraScale+ MPSoC の ARMv8 アーキテクチャの中心にあるのは、ハードウェア アクセラレーションを使用した仮想化です。相乗効果をもたらすこのアーキテクチャによって、上記のシナリオが実現できるだけでなく、労力を最小限に抑えて実装することができます。これは、それぞれのゲスト ソフトウェア スタックをサンドボックスに分離することで可能になります。ハードウェア アクセラレーションを使用しない場合、これらのシステムの複雑度は大幅に上がり、実装と管理が非常に困難になります。

システム負荷の最適化

システム負荷を綿密に管理することはエンベデッド システムにとって共通の課題ですが、ハイパーバイザーを活用しない場合はさらに多数の大きな課題が引き起こされます。Linux などの従来型 OS は、対称型マルチプロセッシング (SMP) タスクの処理に非常に長けており、すべてのプロセッサ コアをその制御下に置きます。しかし、すべてのプロセッサに割り当てるタスクがない場合はどうなるのでしょうか。Linux では、使用されていないプロセッサはアイドル状態になります。複数の Linux インスタンスをそのままで (Linux を変更することなく) 仮想化し、インスタンスごとに固有のタスクを実行することができます。この場合、必要があれば新しい Linux インスタンスがオンラインになり、逆に需要が低い場合は、一部のインスタンスが停止されます。その結果、プロセッサ自体は必要に応じて回転数を上げるかアイドル状態になりますが、システム全体としてはほぼ常に使用率の高い状態が維持されます。同様の戦略を採用して、複数の異なるソフトウェア スタックを管理することができます。すべての仮想化 OS が同じである必要はないため、エンド システムの柔軟性とモジュール性を高めることができます。

ソフトウェアの分離と分割の最適化

ソフトウェアの分離と分割も、アンマネージド環境に大きな課題をつきつけるテーマです。システム負荷シナリオとよく似ていますが、このシナリオでは追加要件として、各ソフトウェア スタックが、同時に稼働する別のスタックに悪影響を与えないようにする必要があります。このシナリオを最も簡単に示す例は、2 つのリアルタイム OS (RTOS) スタックを並列で実行するケースです。従来の手法でこれに対処すると、それぞれの RTOS が限りある割り当てリソースのみを使用するように制限するため、ソフトウェアの複雑度が高くなります。反対に、仮想化システムで使用されるサンドボックス方式では、各 RTOS がサンドボックス内にあるすべてのリソースを完全に制御下に置くことができます。サンドボックスが 1 つのシステムであるかのように処理でき、同じシステム上で稼働するその他のソフトウェアを意識することなく、リソースを利用できます。このサンドボックス方式では、特定のハードウェア プラットフォームに関連するコードの量が非常に少なくなるため、コードの移植性が非常に高くなります。また、RTOS はシステム上のその他のソフトウェアを意識する必要がまったくないため、非常にシンプルな処理になります。このように、特定のプラットフォームへの依存から解放されることで、開発者がコードを 1 回書くだけで、さまざまなシステムにデプロイできるという極めて大きなメリットがあります。

スケーリングと冗長性の最適化

SoC のパフォーマンスと性能が上がるにつれて、スケーリングと冗長性に対するエンベデッド システムの要求も高くなっています。

スケーリングを実現するには、プロセッサにロードするソフトウェア数をシステム要求に合わせて増やす必要があります。たとえば、高性能なコンピューティング環境でユーザーからの追加要求に応えるには、Linux ベースの OS インスタンスを追加でオンラインにする必要があります。仮想化を利用すると、まったく同一の Linux システムのコピーを必要に応じてオンラインにすることができ、後でシステム要求が減少したら、これらの Linux インスタンスを停止することができます。

冗長性を提供するには、システムに過度な負荷がかかった場合も特定のサービスの可用性を維持する必要があります。たとえば、RTOS には特定のシステム機能を監視する重要な機能があります。では、何らかの理由で RTOS 自体に障害が発生した場合はどうなるのでしょうか。仮想化を使用している場合、システム モニターが障害を検出し、RTOS を再起動するか、または新しいインスタンスを起動するため、クリティカルなシステム サービスのダウンタイムはまったく (またはほとんど) 発生しません。

XEN を選ぶ理由

ハイパーバイザー ソリューションを選ぶときに重要になるのはロバスト性と信頼性です。また、関連分野の変化に遅れを取らないようにするため、活発な開発が実施されているソリューションを選ぶ必要もあります。Xen ハイパーバイザーはまさにこの条件に当てはまるソリューションです。

Xen の開発は 1990 年代後半に、より大規模な Xenoserver プロジェクトの一環としてケンブリッジ大学で始まりました。2000 年代前半にはオープンソース コミュニティに公開され、2013 年に Linux Foundation の傘下に入りました。Linux Foundation という後ろ盾を得たことで、Xen は Linux ベースの OS にとってデファクトのハイパーバイザー ソリューションとなっています。

従来からの Xen アーキテクチャは x86 との互換性を持っていましたが、最近のホスト開発により、ARM アーキテクチャでも堅牢なソリューションになりました。Xen はシステム メモリ管理ユニット (SMMU) を含めて、ARMv8 の下位にある仮想化ハードウェアを全面的に活用しています。

Xen は標準 GPLv2 ライセンスに従って無料で提供されており、活発なユーザー コミュニティによる新機能の開発や、さまざまなテクニカル サポートが行われています。商用のサービスとサポートを必要とするインテグレーター向けには、DornerWorks などのベンダーがプロフェッショナルなサポートと体制を提供しています

ハイパーバイザー ソリューションの選択においては、ソフトウェア サポートも重要な差別化要因になります。Xen は下位のハードウェア上で直接実行されるタイプ 1 のハイパーバイザーであり、VMWare や VirtualBox などのホスト OS で稼働するタイプ 2 とは異なります。

Xen ハイパーバイザーは、ゲスト OS を複数のドメインに分割し、特別な管理インターフェイスである Dom0 を使用して、ハイパーバイザーの実行時動作を制御します。このドメインは、Xen ハイパーバイザー専用のソフトウェア インフラストラクチャを提供しており、Xen が正しく機能するためにはこの動作構造が不可欠です。Dom0 はカーネル内の特権ソフトウェアと、下位のハードウェアに直接アクセスできる特別な管理ドライバーを利用します。最も一般的な Dom0 OS の 1 つが Linux であり、Zynq UltraScale+ MPSoC に対して堅牢なサポートが提供されています。

すべての標準ゲスト OS は非特権ドメイン内にあり、まとめて DomU と呼ばれます。Xen では、Linux などの高水準 OS や FreeRTOS などのリアルタイム OS、ベアメタル コードまでの各種ゲスト ソフトウェアに対して、DomU での使用がネイティブ サポートされています。ホストとゲストの組み合わせは、全面的にシステム設計のニーズに合わせて決定できます。対照的に、ほとんどの商用ハイパーバイザー ソリューションがサポートするのは、限られた数のゲストと、(たいていはハイパーバイザー プロバイダー自身によって開発された) 非常に特殊な構成のみです。

XEN ハイパーバイザーと ZYNQ ULTRASCALE+ MPSOC の容易な統合

技術的には最高のコンポーネントであっても、ユーザー システムへの実装が複雑なハイパーバイザー ソリューションは使用される可能性が極めて低くなります。Xen は Linux カーネルと密接に関連付けられているため、この点では非常に有利です。Xen ハイパーバイザーのサポートを有効化する作業は、Linux カーネルに含まれる機能の有効化と何ら変わりありません。また、DomU サンドボックスの管理は手動でも、Xen Tools と呼ばれるツール スイートからでも実行できます。

Linux を Dom0 として使用している場合、Xen Tools のインストールはコマンド ラインから非常に簡単に実行できます。RPM や APT などの一般的なパッケージ マネージャーかソース コードを使用して、Xen Tools をビルドおよびインストールすることも可能です。

Xen Tools をインストールした後は、xl と呼ばれるツールを使用して、Linux ユーザー空間から DomU を作成、管理、破棄することができます。

新しい仮想マシンを作成する場合、仮想化環境を定義したプレーン ASCII のコンフィギュレーション ファイルを作成するだけで作業は完了です。ファイルに指定する内容には以下が含まれます。

- 仮想マシンに割り当てるメモリのサイズ
- 仮想化 CPU の数
- ネットワーク情報
- ディスク イメージ ファイル

コンフィギュレーション ファイルは、テキスト解析ツールやリビジョン管理ソフトウェアを使用して簡単に管理できます。以下にコード例を示します。

```
# This configures an HVM rather than PV guest
builder = "hvm"

# Guest name
name = "My Virtual Machine"

# Initial memory allocation (MB)
memory = 128

# Number of VCPUs
vcpus = 2

# Disk Devices
# A list of `diskspec' entries as described in
# docs/misc/xl-disk-configuration.txt
disk = [ '/dev/vg/guest-volume,raw,xvda,rw' ]
```

仮想マシンの管理に使用するコマンドは xl と呼ばれており、起動から停止までにおよぶ Xen 仮想マシンのライフサイクル全体の管理に使用できます。既存のコンフィギュレーション ファイルを使用して新しい仮想マシン インスタンスを作成する場合、次の簡単なコマンドで Xen ハイパーバイザーを作成できます。

```
xl create <path_to_configuration_file>
```

list、reboot、shutdown を含むその他の xl オプションを使用すると、仮想マシンのライフサイクル全体を管理できます。

まとめ

仮想化からメリットを得るシステムにはさまざまな種類がありますが、ARMv8 プロセッサを搭載したザイリンクス Zynq UltraScale+ MPSoC などのマルチプロセッサ プラットフォームと Xen を組み合わせると、その他の方法であれば非常に手間のかかる多くの複雑なシステム要件を簡単に実装することができます。仮想化を使用することで、以下の有益で重要な特長を実現できます。

- 分割
- 分離
- 信頼性
- 冗長性

仮想化は上記の特長を可能にするだけでなく、容易に実現します。最も費用効果の高い最善の結果を確実に得るためには、仮想化ソリューションを考慮に入れる必要があります。ARMv8 マルチプロセッサを搭載したザイリンクス Zynq UltraScale+ MPSoC は、Xen ハイパーバイザーおよび Xen Tools に最適な設計プラットフォームです。

ソフトウェア開発者は仮想化を使用することで、移植性と信頼性に優れたソフトウェア スタックを、必要な性能を発揮するためのグルー ロジックを最小限に抑えて開発できます。また、Xen は、Linux Foundation からの全面的な後援に加えて、さまざまなオペレーティング システムとソフトウェア スタックの統合およびサポートを通じて高いロバスト性を実現しています。Xen ベースの仮想マシンは、使いやすいコマンド ライン インターフェイスとリビジョン管理に適したプレーン ASCII テキストのコンフィギュレーション ファイルを使用することで簡単に管理できます。

GitHub のザイリンクス ページ (https://github.com/Xilinx/linux-xlnx) で提供している Linux カーネルは、Xen サポートが有効化されています。Xen ハイパーバイザおよび Xen Tools のソース コードと Xen Project の詳細情報については、http://www.xenproject.org/ をご参照ください。

DornerWorks はザイリンクスのパートナーであり、Xen ハイパーバイザーのサポートを提供しています。詳細情報は以下の Web サイトをご参照ください。
http://dornerworks.com/services/xilinxxen

改訂履歴

次の表に、この文書の改訂履歴を示します。

日付	バージョン	内容
2016 年 3 月 31 日	1.0	初版

免責事項

自動車用のアプリケーションの免責条項