4 組み込みマルチコアプロセッサのソフトウェアプラットフォーム

Software Platform for Embedded Multi-core Processor

酒井淳嗣
NEC システムデバイス研究所
jsakai@bc.jp.nec.com

井上浩明
NEC システムデバイス研究所
h-inoue@ce.jp.nec.com

枝廣正人
NEC システムデバイス研究所
eda@bp.jp.nec.com

プロセッサのマルチコア化が進んでいる．PC やサーバ等で Intel 社のデュアルコア CPU が使われ始めているのはよく知られているが，情報家電，ゲーム機，携帯電話など，組み込み機器へのマルチコア CPU 導入も検討されつつある．本稿では，組み込み機器向けマルチプロセッサでのソフトウェアプラットフォームについて述べる．

組み込み機器のニーズ

PC と比べたときの組み込み機器の特徴とは何だろうか？　いくつもある差異のうち最も自明な点は，

(a) 発熱量やバッテリー動作という制約から低消費電力にしなければならないこと

であるが，もう 1 つ重要な点として，

(b) アプリケーションの動作性能を保証すること

があげられる．一般に，組み込み機器では PC と異なり，搭載されているアプリケーションが固定されている反面，搭載されている各機能（たとえばブラウザ，動画，音声認識など）は，もしそれが機器の仕様として認められているのであれば，どのような状況下でも決められた性能を維持して動作することが暗に期待されている．また，予想外のバグであれ悪意あるウィルスであれ，アプリの不正な動作によって機器全体がダウンするなどということもあり得ない，と信じられている．組み込み機器メーカはこういった暗黙ニーズに応えつつ機器開発を行わなければならない．組み込み機器に求められるこのような要求を，ここでは性能保証要求と呼ぶことにする．

他方で組み込み機器に搭載される機能はどんどん高度化している．図 -1 は携帯電話に搭載される機能のロードマップの一例を示したものである．高機能化にはメディア性能と汎用 CPU 性能の 2 つの方向がある．前者は動画のエンコード，デコード等に代表されるもので，専用ハードウェアエンジンやメディアプロセッサ，メディア拡張命令等によって対処されることが多い．後者は Java，セキュリティ処理，認識処理などであり，基本的には今後も CPU 性能の一層の向上が求められる．この汎用 CPU 能力向上ニーズに対し，前記 (a) の観点からマルチコアアプローチを考えるのは自然な流れである．

組み込み機器向けマルチコア

一口にマルチコアと言ってもさまざまな方式が考えられる．比較的少数のコアを用いる場合について，OS と

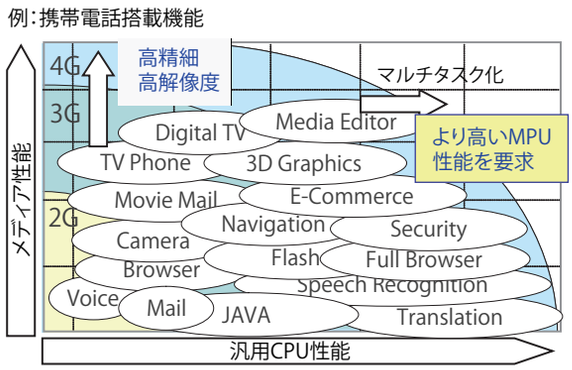


図 -1　携帯電話搭載機能ロードマップ

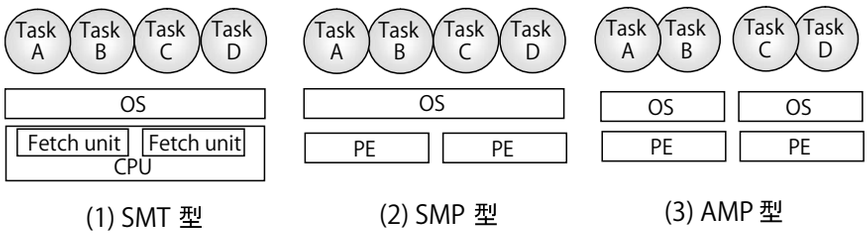


図 -2　マルチコア CPU と OS

CPU の独立性観点から図 -2 にいくつかの方式を比較して示す．

（1）**SMT（Simultaneous Multi-Threading）型**：CPU 内の演算リソースはシングル CPU と同等だが，命令フェッチと発行のユニットが複数あり，複数スレッドの命令流を並行に実行できる [1]．これは厳密にはマルチコアとは呼ばないが，他との比較のためここにあげている．

（2）**SMP（Symmetric Multi-Processing）型**：CPU 内に独立した PE（Processing Element）が複数あり，異なるスレッドやプロセスを同時に実行する．OS は SMP 用のものを使い，全 PE がその OS で管理される．

（3）**AMP（Asymmetric Multi-Processing）型**：PE だけでなく OS も独立になったものを SMP に対比して AMP と呼ぶことにする．CPU 内にはシングル CPU 相当の PE が複数あり，各 PE で独立してシングル CPU 用 OS を走らせる．

（1），（2），（3）はこの順に独立性が高くなる．前述の（b）すなわちアプリケーションの性能保証の観点から言えば，アプリケーションの実行環境が互いにできるだけ独立であることが望まれる．すると図 -2 の（3）AMP 型が最もふさわしいことになる．PE ごとに独立に OS を走らせることはやや極端に映るかもしれないが，アプリケーションに含まれる不本意なバグや，機器出荷後に追加されるダウンロードアプリケーション（これらは出荷時組み込み済みアプリケーションに比べ信頼性の面でやや劣る）に対して組み込みシステム全体がどれぐらい堅牢であるべきか，という点を考えると，組み込み機器ではこれぐらいの独立性確保が必要であると言うこともできよう．

ただし，この AMP 型は PE や OS といった比較的下位のレイヤがほぼ完全に独立してしまっている．これは分散並列マシンに似た構造であり，アプリケーションからは使いにくい．これまでシングル CPU 環境で膨大なソフトウェア資産を構築してきた組み込み分野にこの AMP 方式を適用するには，ソフトウェアからマルチコアを利用しやすくする何らかの工夫が必要である．

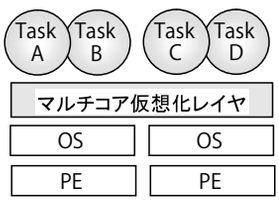


図 -3　タスク並列方式

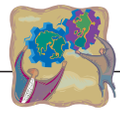
タスク並列方式

我々は，AMP 型ハードウェアおよび OS の上に，アプリケーションからのマルチコア利便性を改善するソフトウェアレイヤを構築し，組み込み機器に適したマルチコアプラットフォームとした．このソフトウェアレイヤを「マルチコア仮想化レイヤ」と呼び，マルチコア仮想化レイヤを用いた AMP 型並列実行方式を「タスク並列方式」と名づけた [2]．

タスク並列は図 -3 のように，シングル CPU コア，シングル CPU 用 OS の上に，全 PE をまたぐようにマルチコア仮想化レイヤを載せた構成になっている．各アプリケーションタスクは物理的にはいずれか 1 つの PE に固定的に割り当てられる．

メインメモリは，物理的には全 PE で 1 つを共有するが，そのメモリ領域を分割し，各 PE には論理的に独立した空間を割り当てる．メモリを物理的に共有するのはひとえにハードウェア実装の都合であり，メモリ共有によりバス競合など不都合が生じないよう，実装においてはメモリバス設計に注意を払う必要がある．

マルチコア仮想化レイヤは，アプリケーションがマルチコアを意識せずに済むようにするためのもの，言い換えれば，アプリケーションをシングル CPU 環境からマルチコア環境に移行しやすくするために存在するレイヤである．ここには，複数 PE の存在を隠蔽する「OS



Wrapper」と，PE につながる周辺ハードウェアを統合管理する「アクセスサーバ」の 2 つの機構を導入している．

OS Wrapper

OS Wrapper はタスク並列方式を現実的なものにした中心技術の 1 つである．アプリケーションと OS の間に位置し，PE をまたがるタスク間通信をシームレスに実現する．ここでシームレスとは，通信相手のタスクが自タスクと同一 PE 上にある場合も別 PE 上にある場合も，同じ API（Application Program Interface）が使える，という意味である．タスク間通信がシームレスになることで，アプリケーションをマルチコア環境に移植することが容易になる．

OS Wrapper は大きく分けて，OS 標準のタスク間通信 API をフックするクライアントライブラリ CLlib，OS に代わってタスク起床制御を行う代理タスク，そして PE 間の最もプリミティブなイベント通知を行う ipi（Inter-PE Interrupt; PE 間割り込み）ドライバの 3 要素からなる．図 -4 を用いて，異なる PE 上のタスク A，B 間で，どのようにしてシームレスなタスク間通信が実現されているかを説明する．

1）タスク A が送信 API（Send() とする）を用いてデータ送信する．
2）タスク A にリンクしてある OS Wrapper クライアントライブラリ CLlib がそれをフックし，送信相手が自 PE 上タスクか他 PE 上タスクかを判断する．
3）他 PE である場合，デバイスドライバ経由で割り込みコントローラ INTC を操作して ipi を起こし，相手 PE に知らせる．
4）相手 PE では ipi 割り込みを受け，代理タスクが呼び出される．
5）受信側タスク B は，受信 API（Recv() とする）を呼んでブロックしている．OS Wrapper クライアントライブラリ CLlib のため，この API はフックされ，タスク B は OS Wrapper 管理下で待機している．
6）ipi 割り込みがくると，代理タスクは CLlib 内でブロックしていたタスク B にシグナルを送る．
7）タスク B は実行再開し，CLlib がタスク A からの送信データを受け取り，タスク B 本体に返す．

シームレスなタスク間通信は OS カーネル内部に実装することも可能であるが，このように OS カーネルから独立した実装にしておくことで，OS のバージョンアップに対する追随が容易になる．さらに，AMP 型マルチコアプロセッサの製品適用でしばしば検討にあがる，ヘテロ OS 環境，すなわち，異なる種類の OS を各 PE に対して搭載する場合にも，このようなユーザランド実装によりシームレスタスク間通信の実現が容易になると考えられる．

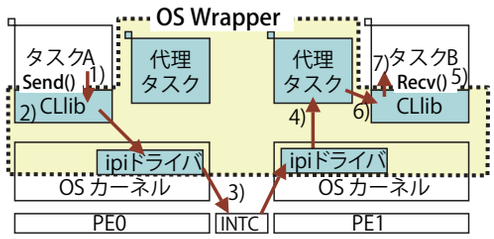


図 -4　OS Wrapper の仕組み

我々はタスク並列方式を Linux システム上に構築した．評価に用いるアプリケーション群（後述）を事前に精査し，タスク間通信としてサポートすべきものは，SystemV IPC（メッセージ，セマフォ，共有メモリ）と UNIX ドメイン socket であることが分かった．そこで，今回の試作ではこれらの API のみを上記 OS Wrapper として実装することにした．クライアントライブラリは各プロセスにリンクしておく必要があるが，これは Linux のダイナミックライブラリ プリロード機能（環境変数 LD_PRELOAD に指定したライブラリは，プロセス起動時に自動的にリンクされる）を用いることで，各プログラムの再ビルドなしで行えるようにした．

アクセスサーバ

AMP 型マルチコア環境では各 OS は独立であり，各 OS はそれが動作する PE 内でしかハードウェアリソース（例 : 画面 , キー入力デバイス , ファイルシステム等）の管理を行わない．そのため，異なる OS 上のタスクから同一ハードウェアリソースへアクセスしようとすると，OS による排他制御が働かず，結果として正しくない動作を引き起こす可能性がある．しかし，OS レイヤで排他制御機構を導入すると，たとえばある OS がロックを獲得したままハングアップしてしまうと別 OS もロックを獲得できなくなるなど，タスク並列の良さである OS 間の独立性が失われてしまう．

そこで，PE 間で共有するハードウェアリソースの管理は，アプリケーションレイヤでの特別なタスクに行わせることにした．すなわち，あるハードウェアリソースへのアクセスは特定 PE 上の特定タスクに限定するとともに，他のタスクはこの特定タスクとタスク間通信を行うことで間接的にハードウェアリソースにアクセスする．これはハードウェアリソース管理に関するクライアント・サーバ方式そのものであり，我々はこの特定タス

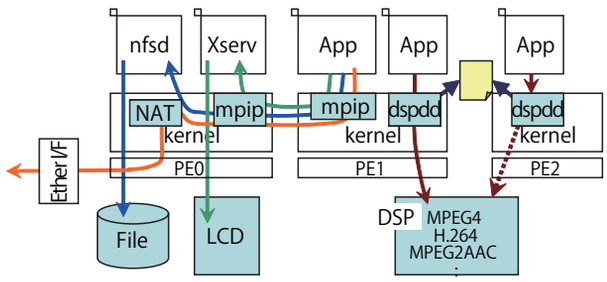


図-5　アクセスサーバとダイレクトアクセス

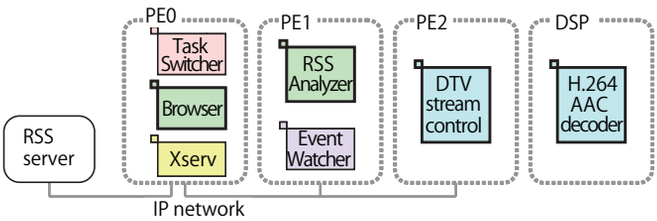


図-6　アプリケーションの配置

クのことを「アクセスサーバ」と呼んでいる．

我々の Linux 上実装では，画面制御のための X サーバ，ファイルシステム管理のための NFS デーモンを，それぞれアクセスサーバとして利用している（図-5）．このように，Linux で一般的なハードウェアリソースに関しては Linux で普通に用いられているソフトウェアツールが流用できる点が，アクセスサーバ方式の利点である．

このクライアント—サーバ間通信はオンチップの高速な通信機構を介して行われるため，通常はアクセスサーバ経由アクセスによるオーバヘッドは無視できる程度である．しかし，メディア処理を DSP で行わせる場合の制御など，高バンド幅，低レイテンシの両面からアクセスサーバ方式では不十分な場面が存在する．このような場合はダイレクトアクセス方式，すなわち，各 PE 上のデバイスドライバから直接ハードウェアリソースをアクセスする方式を採用する．PE 間の排他制御はデバイスドライバを呼び出す前のユーザ空間（ライブラリ層）かデバイスドライバ内で行うが，OS の相互独立性を維持するためには前者の方式のほうが望ましい．

後述の実証実験では，動画音声デコードを行う DSP 上のミドルウェアが数十 ms 単位での制御を必要とするため，このダイレクトアクセス方式でアクセスを行っている．PE 間排他制御はユーザ空間ライブラリ内で行っている．

実証実験：地デジ表示＋ニュースリーダ

タスク並列方式によってアプリケーションの性能保証がうまく行えるようになったかどうか，地上ディジタル TV 放送（地デジ）表示と RSS ニュースリーダの 2 つのアプリケーションを含むタスク群を用いて実験した．このアプリケーションミックスは近い将来の携帯電話搭載機能を想定したものである．地デジ表示アプリケーションは DSP 上で動作する H.264+AAC デコーダと CPU 上で動作するメディアプレーヤからなり，もう 1 つのニュースリーダは CPU 上で動作する 2 タスク，RSS アナライザーと HTML ブラウザからなる．

タスク並列実行環境は，アプリケーションプロセッサ MP211 と Linux OS である．MP211 は携帯機器での実績の多い ARM プロセッサの 1 つ ARM926（192MHz）を 3 個搭載した NEC エレクトロニクス社のシステム LSI であり[3), 4)]，その上に ARM 用 Linux（シングル CPU 用の Linux 2.4 がベース）を載せてタスク並列環境とした．MP211 にはまた，NEC エレクトロニクス社製の DSP SPXK6（192MHz）を 1 つと，グラフィックアクセラレータなどいくつかの IP コアを搭載しており，メインメモリとして DDR SDRAM（今回は 64MB 使用）を直結して使用する．

上記アプリケーションミックスを MP211 の 3CPU+1DSP に載せた版（以下 3PE 版）と，このうち 2CPU をとめて 1CPU+1DSP としたハードウェア環境に載せた版（以下 1PE 版）をつくり比較した（図-6）．1PE 版では，ニュースリーダが表示するコンテンツを更新する際に，地デジ表示画面の動画と音声が一瞬停止するのが確認された．ニュースリーダが動き出さない場合は地デジ表示も問題なかった．それに対し，3PE 版はニュースリーダの動作状態にかかわらずスムーズな動画再生を続け，音の途切れも確認されなかった．

1PE 版で動画再生が一瞬止まるときのプロセス動作状態図を図-7 に示す．ニュースリーダ動作時に動画ストリームの demux スレッド（図中で h264d_a と示されているもの）の周期動作が欠落していることが分かる．つまり，1PE 版ではニュースリーダアプリケーションが内容更新する際の一時的な処理負荷が，地デジアプリケーション中のリアルタイム処理スレッドの周期的動作に悪影響を与えていたことになる．3PE 版ではこのような性能干渉がなくなり，各アプリケーションの性能が保証されたかたちになっている．

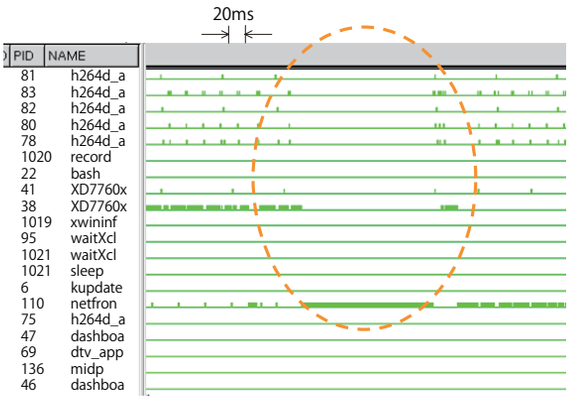


図 -7　1PE 実行時のプロセス動作状態

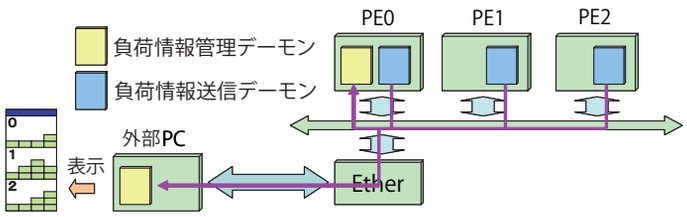


図 -8　PE 負荷バランス情報の収集

タスク並列でのソフト開発

アプリケーションから見ると，タスク並列環境はシングル CPU 上のマルチタスク環境と同様である．シングル CPU 環境で OS 標準のタスク間通信によってマルチタスク処理を記述し動作確認しておけば，それをそのままタスク並列環境に持っていくだけで，タスク並列環境で動作するアプリケーション群ができる．この移行容易性はマルチコア仮想化レイヤの働きによるものである．デバッグはまずシングル CPU 上で十分行っておき，タスクの PE 配置のみ指定した上でタスク並列環境でのデバッグを再開する．タスクの PE 配置はそれぞれのシステムに依存するが，標準的な Linux なら，/etc/rc 等で特定 PE 上にのみタスクが起動するように記述することになる．

タスク間の微妙なタイミングが関係する部分のデバッグには，LTT（Linux Trace Toolkit; カーネルのスケジューリング動作をトレースする機能）をマルチコア対応させたツールを使用する．これにより，PE をまたがる並列関連 API の振る舞いを観察できる．また，PE 間の負荷バランスを確認するために，負荷の重さなどのシステム情報を常駐タスク（デーモン）で取得し，1 カ所に収集した上で時系列表示するツールも開発，利用している（図 -8）．

おわりに

本稿で紹介したタスク並列方式は，現在および近い将来の組み込み機器を想定し，純粋な技術的課題の側面のみならず，ハードウェア／ソフトウェアの実装容易性，入手容易性，想定されるアプリケーションの種類や数，などの現実的な側面も勘案して生み出したものである．

今後，より性能保証を重視するならタスク並列のままコア数を増やすなどの強化が必要だろうし，アプリケーションの自動分散配置をより良くしようとすると SMP 技術との融合が必要だろう．また，リアルタイム OS など，純粋なソフトウェアによる性能保証との組合せについてもあわせて検討していく必要があろう．

参考文献

1）Tullsen, D. M. et al.: Simultaneous Multithreading: Maximizing on-chip parallelism, The 22nd Annual International Symposium on Computer Architecture, pp.392-403 (June 1995).
2）Sakai, J. et al.:Multi-Tasking Parallel Method on MP211 Multicore Application Processor, IEEE Symposium on Low-Power and High-Speed Chips (COOLChips VIII), pp.198-211 (Apr. 2005).
3）Torii, S. et al.: A 600MIPS 120mW 70μA Leakage Triple-CPU Mobile Application Processor Chip, 2005 IEEE International Solid-State Circuits Conference, Digest of Technical Papers, pp.136-137 (Feb. 2005).
4）NEC エレクトロニクス , MP211 製品紹介, http://www.necel.com/ja/techhighlights/application_processor/product.html

（平成 17 年 12 月 9 日受付）