既存のソフトウェア資産をマルチコア環境で生かす

RPC技術を利用した関数並列型マルチコア・プログラミング

マルチコアCPUの性能を引き出すためには，マルチコア対応OSの導入に加えて，アプリケーション側のタスク分割などの対応が必要となる．本稿で提案されているAsynchronous Remote Procedure Call（ARPC）ベースのプログラミング・モデルでは，OSやメイン関数はメインのプロセッサで，そこから呼び出される関数はサブプロセッサで動作させるという方法を採る．これにより，既存のソフトウェアをマルチコアに対応させている．（編集部）

須賀 敦浩，鈴木 貴久

プロセッサには，サーバ系で使われるものから機器に組み込まれるものまで，さまざまな種類があります．筆者らは，プロセッサをさまざまな角度から研究しており，組み込み向けのVLIWアーキテクチャのメディア・プロセッサ，およびそのマルチコア版を開発しました[(1),(2)]．最近では機器メーカの開発者が使いやすいプロセッサのアーキテクチャについて，プログラミング言語からハードウェアまで幅広く取り組んでいます．

マルチコア・プロセッサを使いこなすためには，ハードウェアだけでなくソフトウェアからのアプローチも必要です．本稿では，組み込みシステムの現状について概観したあと，筆者らの会社の研究部門と半導体事業部門が一体となって開発している，関数並列型マルチコア・プログラミングについて解説します．

1 組み込みシステムの現状

かつて組み込みシステムの製品開発では，ハードウェアが製品の機能や性能を決めていました．ソフトウェアは，機能を実現するための制御などの補助的な役割を担っていました．ハードウェアとしてのLSIは，機能および性能がカタログ・スペックから分かるので，設計の初期段階で機能や性能を議論できたのです．

ところが，最近では組み込みプロセッサの性能が向上したりマルチコアが使用できるようになり，システム製品において，ソフトウェアが機能および性能を決める重要な要因になりつつあります．製品の機能や性能について，ハードウェアだけでは決定できず，ソフトウェアを改造するのか，従来のものと互換にするのかなどの方針を第一に考える必要があります．

このような状況下で製品を開発するためには，「ソフトウェア資産の継承」，「適切な価格と機能の実現」，「短期間での製品開発」という課題を解決する必要があります．ソフトウェア資産の継承というのは，製品開発を進める上で，最も重要な問題です．ソフトウェア資産の継承については，コア数が変化しても既存のソフトウェアに手を加える必要がなく，なおかつコア数に応じた性能向上が可能でスケーラブルなマルチコア技術が重要であると考えます．

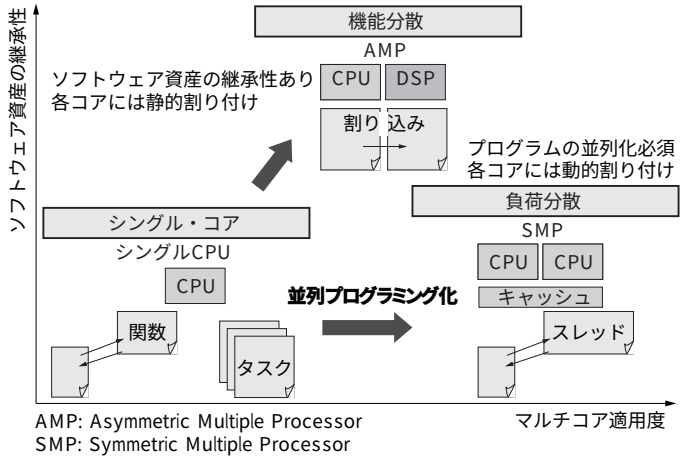

図1 並列プログラミングとマルチコア適用度

2 マルチコアと並列プログラミング

まず，マルチコア・プロセッサと並列プログラミングについて説明します．

図1では，シングル・コア上のソフトウェアを原点とし，横軸にマルチコア適用度，縦軸にソフトウェア資産の継承性をとっています．ここで言うマルチコア適用度とは，マルチコアの性能をどれだけ引き出せているかの指標です．

● SMP と AMP の 2 種類がある

マルチコアの技術として，コアのアーキテクチャやコアからアクセスするリソースが対称的に配置される SMP（Symmetric Multiple Processor）型と，SMP の制約を受けない非対称な AMP（Asymmetric Multiple Processor）型があります．

AMP 型では，それぞれのコアが得意とするソフトウェアを実行します．各コアにこれまで開発したソフトウェア（例えば画像処理ソフトウェアや音声処理用ソフトウェアなど）をそのまま搭載できるので，ソフトウェア資産の継承性を保てます．しかし，各コアへの割り付けはプログラマが静的に行うので，コアの数を変更する場合にはプログラムの変更が必要です．つまり，ソフトウェアはプロセッサの種類，およびコアの数などの物理制約に依存したものとなります．

一方の SMP 型は，実行するスレッドがどのプロセッサ・コアで実行するのかを OS などが動的に割り付けます．その反面，プログラムの並列化が必須となります．C 言語主流の組み込みシステム・ソフトウェアにおいて，プログラムが広域変数やポインタを多用している場合，並列化に時間がかかります．広域変数やポインタを使ったプログラムを解析してマルチコアで動作できるような機能を搭載したコンパイラは，なかなか商用化されません．また，既にプログラムがタスクを用いてシングル・コア向けに分割されていた場合でも，タスクを実行する際の優先度を，暗黙の優先度で使用している記述から明示的に指定する記述に変更する必要があります．その結果，デッドロックが起きる可能性もあります．

3 関数並列型 マルチコア・プログラミング

筆者らは，組み込みマルチコア・システムを考えた場合に最も簡単にソフトウェアを記述できる手法について考えました．その結果，動的負荷分散機能を搭載した AMP 型プロセッサ・プラットホームが組み込みマルチコア・システム向けの並列プログラミングとして最も適切であるという結論に達しました．これは，クライアント-サーバ間でよく使われている遠隔手続き呼び出し（RPC，下掲の**コラム 1** を参照）[3] の技術をチップ内のコア間に適用するものです．

コア間の通信を実現するプログラミング手法としては，Asynchronous Remote Procedure Call（ARPC）ベースのプログラミング手法を用います．このプログラミング手法では，シングル・コア上のソフトウェア資産を少ない労力で並列プログラム化することが可能です．

● ARPC を用いたプログラミング

それでは，ARPC を用いたマルチコア・プログラミング・モデルについて説明しましょう．

コラム 1　RPC とは

RPC（Remote Procedure Call）は通常の関数呼び出しと同じ形式で異なるマシン間で通信するための技術です．もともとは分散コンピューティングの分野においてネットワークで接続された異なるマシン間の通信に利用されてきました．

RPC を利用するとコンピュータ間の通信プログラムが簡単に作成できるため，古くから UNIX などでは NIS（Network Information System ；複数マシン間におけるユーザの一括管理などで利用される）や NFS（Network File System ；複数マシン間におけるファイル共有に利用される）などで使われてきました．

RPC を利用したプログラミングでは，ILD コンパイラを利用して開発を行います．IDL とは Interface Definition Language（インターフェース定義言語）の略です．開発者が RPC で呼び出す関数を作成すると，その関数のインターフェース情報として，関数名や引き数の型などの情報を記述した IDL ファイルを作成します．

この IDL ファイルを IDL コンパイラにかけると，ネットワークを介してこの関数を呼び出すためのコードが生成されます．煩雑なネットワーク・コードも IDL コンパイラが生成してくれるので，これを利用するとコンピュータ間で通信を行うプログラムを簡単に開発できます．

RPC には，別のマシンに関数を呼び出した後，関数の実行が完了するまで待ち続ける同期型と，別のマシンに関数を呼び出した後も現在の処理を継続する非同期型（ARPC ：Asynchronous RPC）の 2 種類があります．今回はこの ARPC 技術を利用してマルチコアのコア間通信を実現しました．

この場合，IDL コンパイラがネットワーク・コードの代わりにコア間通信の処理を生成することで，煩雑なコア間通信の処理を隠ぺいすることが可能となります．