Cell Broadband Engine を利用したホログラム計算

マルチコア・プロセッサのプログラミング手法

柘植 宗範，伊藤 智義

ここではマルチコア・プロセッサ向けプログラムの作成手法について解説する．マルチコア・プロセッサの例として，家庭用ゲーム機であるPLAYSTATION3に搭載されている「Cell Broadband Engine」を取り上げる．このプロセッサの上で動作するプログラムの開発手順と，計算機合成ホログラムのプログラムをCell Broadband Engineに移植した例を紹介する．なお，ここで紹介するソース・コードの全文は，本誌のWebサイトからダウンロードできる．（編集部）

昨今，プロセッサの動作クロック周波数の向上やプロセスの微細化が進んでいます．そして，このことがトランジスタのリーク電流の増大を引き起こし，消費電力量や発熱量の増加という問題を生んでいます．これらの問題を回避しながら性能を引き上げる手法として，プロセッサのマルチコア化や並列処理技術の導入が提唱されています．

このような流れの中で，いくつかのプロセッサ・メーカは同じコアを複数個搭載したホモジニアス（対称）型のマルチコア・プロセッサを発表しています．さらに，異なるタイプのコアを一つのプロセッサ内に搭載するヘテロジニアス（非対称）型のマルチコア・プロセッサについても，アプリケーションに特化した処理機構を有するコアを組み合わせたものがいくつか発表されています．

本稿でとり上げるマルチコア・プロセッサ「Cell Broadband Engine（以後，CBE）」は，ソニー・コンピュータエンタテインメント，IBM社，東芝の3社が共同開発したヘテロジニアス型マルチコア・プロセッサです．2006年末に発売された家庭用ゲーム機「PLAYSTATION3」に搭載されたプロセッサであることから，皆さんも耳にしたことがあるのではないかと思います．筆者らが開発に使用した機材を**写真1**に示します．

写真1　CBE（Cell Broadband Engine）の開発機材
右上が家庭用ゲーム機「PLAYSTATION3」．

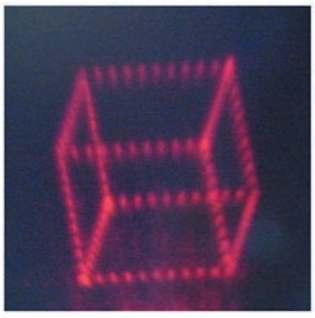
本稿では，まずCBEの構造や特徴を簡単に紹介し，ある処理に対してそれを並列化する作業，および実際のプログラミングについて解説します．また，計算機合成ホログラム（**写真2**）のプログラムをCBEに移植した例を紹介します（p.160の**コラム1**を参照）．

● すべてのプログラムを並列化できるわけではない

並列処理とは，一つの処理を幾つかの小さな処理に分割し，それらを複数の機構で個別に処理することを言います．そして，それら小さな処理を同時並行で行うことにより，全体の実行時間の短縮を目指します．大手プロセッサ・メーカから最近出荷され始めたマルチコア・プロセッサは，その名のとおり複数の処理機構（CPUコア）を利用して並列処理を行っています．

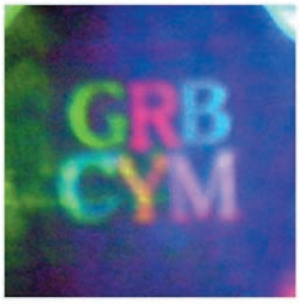
なぜ大手プロセッサ・メーカがシングル・コアからマルチコアへとかじを切ったのかについては，さまざまな理由があるようです．例えばその一つとして，昨今のCPUでは，そのピーク性能で稼働し続けている時間が非常に短くなってきている，ということが挙げられます．多くの場合，CPUで実際に動かそうとしているプログラムはそれほど難しい計算を行っているわけではなく，さらに，そういった単純なプログラムを複数同時に起動していることがほとんどです．そういった場合，一つ一つの処理に多少の手間がかかっても，同時に複数の処理を実行できる方が都合が良い，ということになります．そこに，シングル・コアの性

KEYWORD ── PLAYSTATION3，CBE，マルチコア，計算機ホログラム，電子ホログラフィ，FPGA，SIMD，XDR DRAM，GNU，Cell SDK，Fedora Core

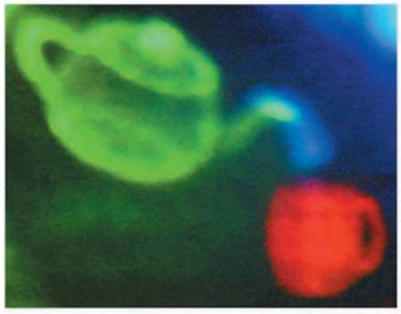

(a) ワイヤ・フレーム・モデル


(b) サーフェイス・モデル


(c) カラー・モデル



(d) 混色モデル


(e) カラー面モデル


(f) 複雑な立体モデル

写真2
電子ホログラフィの表示例

能向上の限界も相まって，「高価な一つのプロセッサよりも（製造歩留まりなどのコストから見て）安価な複数のプロセッサ」という流れが出てきたのだろうと思います．

これとは別に，一つの大きなプログラムを動かす場合も，「それらを分割してできる限り同時に処理する」という観点から，並列処理の重要性が高まっています．この手法は昔からスーパコンピュータの世界でよく利用されています．現在，計算速度の上位を占めるスーパコンピュータは，少なからず並列処理を考えて構成されています．

しかし，すべてのプログラムを並列化できるわけではありません．例えば，A，B，C という三つの処理を実行する場合，効率良く処理したいのであれば，それぞれを別の計算機（CPU コア）の上で処理すれば良いように思えます．しかし，A と B の処理に依存関係がある場合（例えば A で計算した結果を B で使用するなど），A と B は同時に処理できません（**図 1**）．つまり，並列化によって性能向上を期待する場合，その効果は並列化するプログラムの構造に大きく依存すると言えます．さらに，各計算機の間の通信がボトルネックになるなどの問題点も挙げられます．

また，最近増えてきたマルチコア，マルチスレッドの環境では，まだまだプログラミング環境が整備されていないのが実情です．並列処理を行おうとする場合，マルチスレッドなどを意識して，プログラマが明示的にプログラムを書く必要があります．そうなっていないプログラムは，マルチコア環境でもシングル・コアと同じようなの性能しか発揮できません．実際にマルチコア・プロセッサを搭載したパソコンで一つのアプリケーションを走らせてみても，今までのパソコンとあまり差がないと感じることが多いのはそのためです．

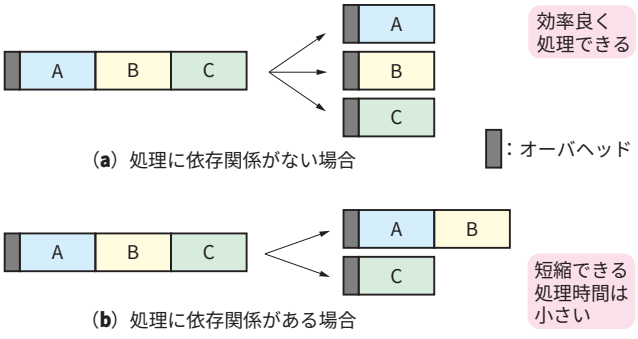


図 1　処理の並列化に伴う性能向上

並列処理を行うと，プログラム全体の処理時間が短縮される．しかし，すべてのプログラムが並列化できるわけではない．例えば，A，B，C という三つの処理を行うとき，A と B の処理に依存関係があると（例えば A で計算した結果を B で利用する場合など），A と B を同時に処理することができない．また，多くの場合，処理のはじめに多少のオーバヘッドが存在する．