Cell Task Manager Cerium のSPU内データ管理

多賀野海人[†4]　宮　國　　渡[†1]
河　野　真　治[†2]　小　林　佑　亮[†4]

PlayStation 3 では、搭載されている Linux を用いてゲームを開発することができるが、GPU の詳細が公開されていないため、Frame Buffer 上に描画する必要がある。Frame Buffer 上の描画は非常に低速である。本研究では Cell 用の Task Manager を実装し、Frame Buffer 上で高速な 3D Graphics Renderer を開発する。

Kaito TAGANO,[†4] Wataru MIYAGUNI,[†1] Shinji KONO[†2]
and Yusuke KOBAYASHI [†4]

In PS3, the game can be developed by using installed Linux. However, because details of GPU are unpublished, it is necessary to draw on Frame Buffer. Drawing on Frame Buffer is very low-speed. In this research, we implement Task Manager for Cell and develop 3D Graphics Renderer high-speed on Frame Buffer.

1. 研究の目的

PS3 上の Cell に搭載されている SPE は Local Store (256KB) にしかアクセスできず、メインメモリには直接アクセスすることができない。メインメモリにアクセスするには Memory Flow Controller を用いて Direct Memory Access 命令を送らなければならない。またこの DMA には待ち時間が発生する。待ち時間の間 SPE が動作しなければマルチコアプロセッサのパフォーマンスが極端に下がる。

本研究では、SPE 内のデータ管理を行うことによって Cell プログラミングの並列度を確保する手法を提案する。

Cerium という独自の Rendering Engine を用いたゲームプログラミングを例題とする。描画するオブジェクトに用いられる Texture データは、SPE の Local Store に収まりきらない場合があるので、データを分割して転送、処理する必要がある。処理に必要なデータの管理を SPE 内で行うことによって、メインメモリとの通信頻度を減らし、Cell プログラミングの並列度の確保を実現する。

†1 琉球大学理工学研究科情報工学専攻
Interdisciplinary Infomation Engineering, Graduate School of Engineering and Science, University of the Ryukyus.
†2 琉球大学工学部情報工学科
Infomation Engineering, University of the Ryukyus.
†3 琉球大学理工学研究科情報工学専攻
Interdisciplinary Infomation Engineering, Graduate School of Engineering and Science, University of the Ryukyus.
†4 琉球大学工学部情報工学科
Infomation Engineering, University of the Ryukyus.

2. Cell

Cell[1] は、マルチコア CPU の 1 つで、構成は「ヘテロジニアス・マルチコアプロセッサ構成」です。汎用的な用途に対応した 1 種類のコアを用意するのではなく、制御系プロセッサコア (PPE) と演算系プロセッサコア (SPE) という異なるコアを用意しています。

2.1 Cell の構成

Cell はメインプロセッサである 1 基の PowerPC Processor Element (PPE) と 8 基のデータ処理プロセッサアーキテクチャー Synergistic Processor Element (SPE) からなる非対称なマルチコアプロセッサであり、高速リングバスで構成されている。[1)]

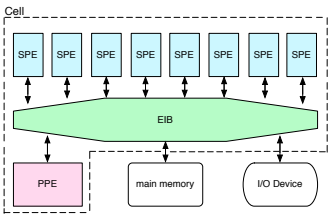

図 1　Cell の構成要素

2.2 PPE

PPE は 複数の SPU をコアプロセッサとして使用することができる汎用プロセッサである。オペレーティングシステムの役割であるメインメモリや外部でバイスへの入出力制御に加えて、SPE を制御する役割も担っている。PPU (PowerPC Processor Unit) は、PPE の演算処理をおこなう核となるユニットで、PowerPC アーキテクチャをベースとした命令セットを持つ。PPSS (PowerPC Processor Storage Subsystem) は、PPU からメインメモリへのデータアクセスを制御するユニットである。

本研究で用いた PS3Linux では、6 個の SPU を制御することができる。

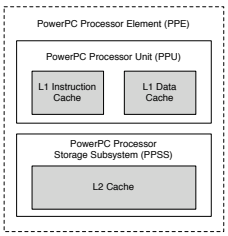

図 2 PowerPC Processor Element (PPE) の構成要素

2.3 SPE

SPE は、 PPE のような複雑なプログラム制御よりも、計算を単純に繰り返すマルチメディア系の処理を得意とする演算系プロセッサである。SPU (Synergistic Processor Unit) は、SPE の演算処理をおこなう核となるユニットで、各 SPE 上に搭載されている。SPU は、PPU とは異なる独自の命令セットを持つ。また、LS (Local Store) という 256KB のメモリを持ち、メインメモリへのアクセスは MFC (Memory Flow Controller) へチャンネルを介して DMA (Direct Memory Access) 命令を送ることで行われる。SPU は直接メインメモリへアクセスすることはできない

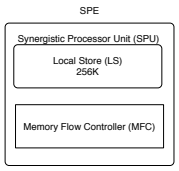

図 3 Synergistic Processor Element (SPE) の構成要素

3. Cell の基本機能

3.1 SIMD (Single Instruction Multiple Data)

Cell は基本的にあらゆるものを並列的に計算できるような構造になっている。SPU に実装されている 128 ビットレジスタも SIMD を行う為に設計されている。SIMD 演算とは 1 つの命令で複数のデータに対して処理をおこなう演算方式である。以下にスカラ演算と SIMD 演算を図で示す。

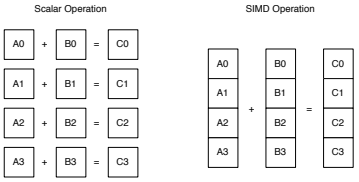

図 4 スカラ演算と SIMD 演算

スカラ演算では、4 個の処理結果を得るために 4 回の加算命令を逐次的に実行しなければならない。一方、SIMD 演算では、1 回の加算命令で 4 個の処理結果を得ることができる。このように SIMD 演算はスカラ演算に比べて同じ処理を少ない命令で実行することができる。SIMD 演算が行えるのは複数のデータに対して同じ処理の場合である。以下のようなデータによって処理が異なる場合には SIMD 演算を行うことができない。

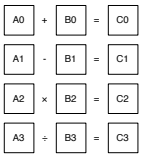

図 5 SIMD 演算を行えない場合

3.2 Mailbox

Mailbox は、PPE と SPE 間で通信するための機構の一つである。DMA 転送はメインメモリと LS との間で最大 16KB の大きなデータの受け渡しを行う。それに対し、Mailbox はステータスの変化などの小さなデータの受け渡し向けで、PPE と SPE との間で双方向のデータの受け渡しが可能で、FIFO キュー構造になっており、3 つの振る舞いができるように設計されている。[2)]

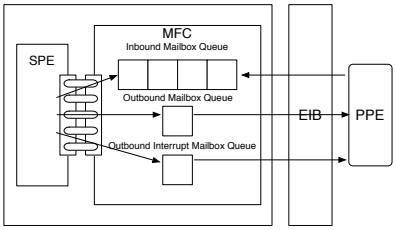

図 6 Mailbox

(1) SPU Inbound Mailbox

PPE から SPE へデータを渡すためのキューで、最大 4 個までのデータを蓄積できる。もし、SPE がキューを読むときにキューにデータがない場合は、キューにデータが書き込まれるまで待ち続ける。

(2) SPU Outbound Mailbox
SPE から PPE へデータを渡すためのキューで SPU INbound Mailbox と同様にデータがない場合はデータが書き込まれるまで待ち続ける。データが 1 個までしかキューに格納できない点が異なる。

(3) SPU Outbound Interrupt Mailbox
SPU Outbound MMailbox とほとんど同じだが、SPE からキューにデータが書き込まれると、PPE に対して割り込みイベントが発生しデータの読み出しタイミングを通知することができる

4. 開 発 環 境

4.1 libSPE2

libSPE2 とは PPU が SPE を扱うためのライブラリ群である。libSPE2 は SPE Context Creation、SPE Program Image Handing、SPE Run Control、SPE Event Handing、SPE MFC PRoblem State Facilities、Direct SPE Access for Applications という基本構成できている。Cell に基本プログラムは次のようになる。

(1) create N SPE context
(2) Load the appropriate SPE executable object into each SPE context's local store
(3) Create N threads
(4) Wait for all N threads to terminate

4.2 SPU C/C++言語拡張

SPE は基本的な C 言語の一部の機能しか使えないが、拡張も行われている。以下にその一部を示す。[3)]

表 1 SPU C/C++ 言語拡張 API

spu_mfcdma32	DMA 転送を開始する
spu_read_in_mbox	PPE からの mail を取得する
spu_write_out_mbox	PPE へ mail を送信する
spu_add、spu_sub、spu_mul	SIMD 演算 (加算、減算、乗算)

このように Cell 特有の関数やアセンブラ命令を学ぶ事が必要となる。

4.3 SPURS

ここでは現在発表されている Cell の開発環境 SPURS について説明する。SPURS[4)] とは閉じた並列分散システムと考えることができる。Cell の環境でいかに効率よく動作させるかということを考えたシステムである。Cell の性能を存分に生かすためには SPE を効率よく使い切ることと、あらゆるレベルで並列処理を行うことである。

Cell の性能を最大限に生かすためには、SPE を止めることなくデータのアクセスを最小限に留めて行うことが重要になる。

そこで SPURS では、SPE を効率よく利用するために、PPU に依存せずに SPU コードを選択し、実行することと機能は効率重視で割り切ることを挙げている。そのために SPE にカーネルを組み込んでいる。

アプリケーションを複数 SPE で実行するとき、アプリケーションプログラムをできるだけ小さな Task に分割し、通信ライブラリを用いて Task 間を依存関係で結合する。LS 常駐のカーネルは実行可能な Task を選んで実行する。

また、アプリケーションを分割するとき、プログラムがデータを伴うとき、ジョブに分割し、並び替えた上で、LS 常駐のカーネルはジョブリストからジョブをとってきて実行する。

また、これたはデータを扱うため、SPURS はパイプライン実行を行う。

SPURS は確かにライブラリとして優れた物であると思われるが、公開はされてない。

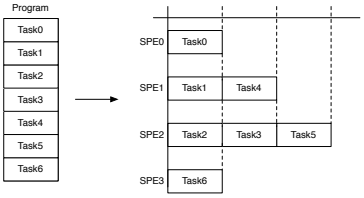

図 7 Task

5. Cerium

6. Texture

7. 評価と考察

8. ま と め

参 考 文 献

1) Sony Corporation: Cell broadband engine architecture (2005).
2) Akira KAMIZATO: Cell を用いたゲームフレームワークの提案, 琉球大学理工学研究科情報工学専攻平成 19 年度学位論文 (2008).

3) : SourceForge.JP: Project Info - Cerium Rendering Engine, https://sourceforge.jp/projects/cerium/.
4) Keisuke Inoue: SPU Centric Execution Model (2006).
5) Shinji KONO: 検証を自身で表現できるハードウェア、ソフトウェア記述言語 Continuation based C と、その Cell への応用，電子情報通信学会 VLSI 設計技術研究会 (2008).
6) International Business Machines Corporation, Sony Computer Entertainment Incorporated, Toshiba Corporation: *SPE Runtime Management Library* (2006).