タイトル

Title

平成27年度 3月 学位論文(修士)

琉球大学大学院　理工学研究科
情報工学専攻

古波倉 正隆

要 旨

目 次

図 目 次

表 目 次

第1章　正規表現の並列化

第2章　Cerium

Cerium は、Cell 向けに開発された並列プログラミングフレームワークである。Cell は Sony Computer Entertainment 社が販売した PlayStation3 に搭載されているヘテロジニアスマルチコア・プロセッサである。本章では Cerium の実装について説明する。

2.1　Cerium の概要

Cerium は当初 Cell 向けに開発され、C/C++ で実装されている。現在では Linux、MacOS X 上で動作する並列プログラミングフレームワークである。

Cerium は TaskManager、SceneGraph、Rendering Engine の 3 要素から構成されている。本研究では汎用計算フレームワークである TaskManager を利用して文字列の並列計算を行なった。

図 2.1 は Cerium が Task の生成/実行する場合のクラス構成図である。TaskManager で依存関係が解消され、実行可能になった Task は ActiveTaskList に格納される。ActiveTaskList に格納された Task は、依存関係が解消されているのでどのような順番で実行されても問題はない。Task は転送を行いやすい TaskList に変換され、CpuType に対応した Scheduler に転送される。なお、転送は Synchronozed Queue である mail を通して行われる。

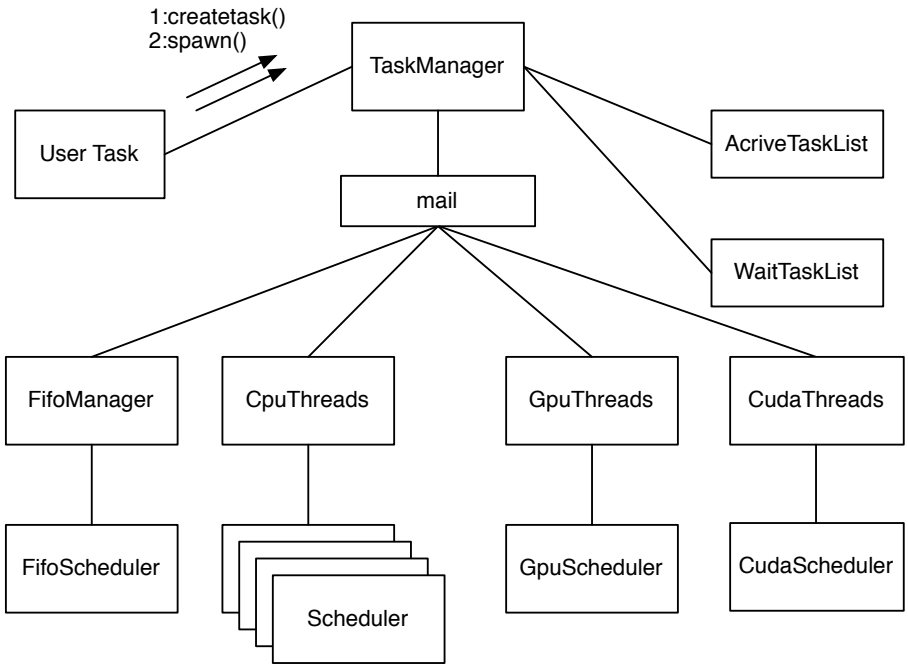

図 2.1: Task Manager

2.2 Cerium TaskManager

Cerium TaskManager では、処理の単位を Task として記述していく。関数やサブルーチンを Task として取り扱い、その Task にて Input Data/Output Data 及び Task の依存関係を設定する。そして Task は設定された依存関係を考慮しながら実行される。

Input Data で格納した 2 つの数を乗算し、Output Data に演算結果を格納する multiply という例題のソースコード 2.1 を以下に示す。

また、Task の生成時に用いる API 一覧を表 2.2 に示す。

ソースコード 2.1: Task の生成

```
1  multi_init(TaskManager *manager)
2  {
3      float *A, *B, *C;
4
5      // create Task
6      HTaskPtr multiply = manager->create_task(MULTIPLY_TASK);
7
8      // set device
9      multiply->set_cpu(SPE_ANY);
10
11     // set inData
12     multiply->set_inData(0, (memaddr)A, sizeof(float)*length);
13     multiply->set_inData(1, (memaddr)B, sizeof(float)*length);
14
15     // set outData
16     multiply->set_outData(0, (memaddr)C, sizeof(float)*length);
```

```
17
18      // set parameter
19      multiply->set_param(0,(long)length);
20
21      // spawn task
22      multiply->spawn();
23  }
```

create_task	Task を生成する
set_inData	Task への入力データのアドレスを追加
set_outData	Task への出力データのアドレスを追加
set_param	Task へ値を一つ渡す。ここでは length
set_cpu	Task を実行するデバイスの設定
spawn	生成した Task を TaskList に set

表 2.1: Task 生成における API

次に、デバイス側で実行される Task のソースコードを 2.2 に示す。

ソースコード 2.2: Task

```
1  static int
2  run(SchedTask *s) {
3      // get input
4      float *i_data1 = (float*)s->get_input(0);
5      float *i_data2 = (float*)s->get_input(1);
6
7      // get output
8      float *o_data = (float*)s->get_output(0);
9
10     // get parameter
11     long length = (long)s->get_param(0);
12
13     // calculate
14     for (int i=0; i<length; i++) {
15         o_data[i] = i_data1[i] * i_data2[i];
16     }
17     return 0;
18 }
```

また表 2.2 は Task 側で利用する API である。Task 生成時に設定した Input Data や parameter を取得することができる。

表 2.2: Task 側で使用する API

get_input	Scheduler から input data を取得
get_output	Scheduler から output data を取得
get_param	set_param した値を取得

Task 生成時に設定できる要素を以下に列挙する。

- Input Data

- Output Data
- Parameter
- CpuType
- Dependency

Input/Output Data、Parameter は関数の引数に相当する。Cpu Type は Task を動作させるデバイスを設定することができ、Dependency は他の Task との依存関係を設定することができる。

2.3 並列 I/O

第3章　Cerium による文字列処理の例題

3.1　WordCount

3.2　Boyer Moore Search

第4章　オートマトン

4.1　オートマトンの定義

4.2　非決定性オートマトン

4.3　決定性オートマトン

4.4　SubSet Construction

第5章　正規表現

第6章　ベンチマーク

第7章　結論

第8章　Chapter

参考文献

[1] 金城裕. 並列プログラミングフレームワーク cerium の改良. 琉球大学工学部情報工学科平成 24 年度学位論文 (修士), March 2012.

[2] 渡真利勇飛. マルチプラットフォーム対応並列プログラミングフレームワーク. 琉球大学大学院理工学研究科情報工学専攻平成 26 年度学位論文 (修士), 2013.

[3] 河野 真治新屋 良磨. 動的なコード生成を用いた正規表現マッチャの実装. 第 52 回プログラミング・シンポジウム, January 2011.

[4] 新屋 良磨, 鈴木 勇介, 高田 謙. 正規表現技術入門 (技術論評社), 2015.

[5] Michael Sipser 著, 太田 和夫・田中圭介監訳. 計算理論の基礎 [原著第 2 版]1. オートマトンと言語, 2008.

[6] Regular Expression Matching Can Be Simple And Fast. https://swtch.com/ rsc/regexp/regexp1.html, 2007.