CodeGear,DataGear による GPGPU 処理

135704C 氏名 東恩納琢偉 指導教員：河野 真治

1 Gears OS

本研究室では並列プログラミングフレームワーク Cerium[1] と分散フレームワーク Alice[2] の開発を行なってきた。

Cerium と Alice を開発して得られた知見から、並列分散処理の信頼性を向上させるには Code の分割だけでなく Data の分割も必要であることがわかった。それををふまえ Gears OS[3] を設計、実装している。

Alice では処理の単位である Code Segment、 データの単位である Data Segment を用いてプログラムを記述 [4] する。Code Segment は使用する Input Data Segment, Output Data Segment を指定することで処理とデータの依存関係を解決する。Gears OS では Gear という単位を用いてプログラムを Code Gear, Data Gear に細かく分割する。これは Alice の Code Segment、 Data Segment にそれぞれ対応する

従来の OS が行う排他制御、メモリ管理、並列実行などは Meta Computation に相当する。Meta Computation は本論の Computation を支える Computation のことである。関数型言語では Meta Computation に Monad を用いる手法 [5] がある。Gears OS では、Meta Code Gear, Meta Data Gear を Monad として定義し、Meta Computation を実現する。

また、接続する Gear を変更することでプログラムの振る舞いを変更することを可能にする柔軟性 、Monad に基づくメタ計算による並行制御を用いた信頼性の確保を目的とする。

2 Continuation based C

CbC のプログラムでは C の関数の代わりに Code Segment を用いて処理を記述している。Code Segment は C の関数と異なり戻り値を持たない。Code Segment の宣言は C の関数の構文と同様に行い、型に __code を使うことで宣言できる。

Code Segment から Code Segment への移動は goto の後に Code Segment 名と引数を並べて記述するという CbC 独自の構文で行う。この goto による処理の遷移を継続と呼ぶ。図 1 は Code Segment 間の継続関係を表している。

C では関数呼び出しを繰り返し行う場合、呼びだされた関数の引数の数だけスタックに値が積まれていくが、戻り値を持たない Code Segment ではスタックに値を積んでいく必要が無くスタックを変更する必要が無い。このようなスタックに値を積まない継続、つまり呼び出し元の環境を持たない継続を軽量継続と呼び、軽量継続による並列化、ループ制御、関数コールとスタックの意識した最適化がソースコードで行えるようになる。

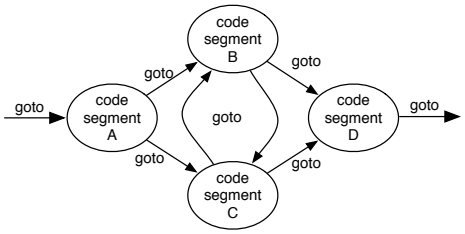

図 1: goto による Code Segment 間の継続

3 Code Gear と Data Gear

Code Gear はプログラムの実行コードそのものであり、OpenCL[7]/CUDA[8] の kernel に相当する。

Code Gear は処理の基本として、 Input Data Gear を参照し、一つまたは複数の Output Data Gear に書き込む。また、接続された Data Gear 以外には参照を行わない。これは Alice の Input Data Segment と Output Data Segment の関係に対応しており、依存関係を解決し、 Code Segment の並列実行を可能とする。

Code Gear は function call ではないので、呼び出し元に戻る概念はない。その代わりに、次に実行する Code Gear を指定する機能 (軽量継続) を持つ。

Data Gear には、int や文字列などの Primitive Data Type が入る。

Gear の特徴の一つはその処理が Code Gear, Data Gear に閉じていることにある。これにより、Code Gear の実行時間、メモリ使用量を予測可能なものにする。

4 Context

ある Code Gear から継続するときには、次に実行する Code Gear を名前で指定する。Meta Code Gear が名前を解釈して、処理を対応する Code Gear に引き渡す。これらは、従来の OS の Dynamic Loading Library や Command

呼び出しに対応する。名前と Code Gear へのポインタの対応は Meta Data Gear に格納される。この Meta Data Gear を Context と呼ぶことにする。これは従来の OS の Process や Thread に対応する。

Context には以下のようなものが格納される。

- Code Gear の名前とポインタの対応表
- Data Gear の Allocation 用の情報
- Code Gear が参照する Data Gear へのポインタ
- Data Gear に格納される Data Type の情報

5 Red Black Tree

Red Black Tree(赤黒木) は平衡二分木の一種で、主に連想配列の実装に使われる。Red Black Tree の Stack Pop,Stack Push している部分に Stub を仲介させることによって、Stack pop と Stack psuh を Data Segment に分けることで CbC による記述に直した。

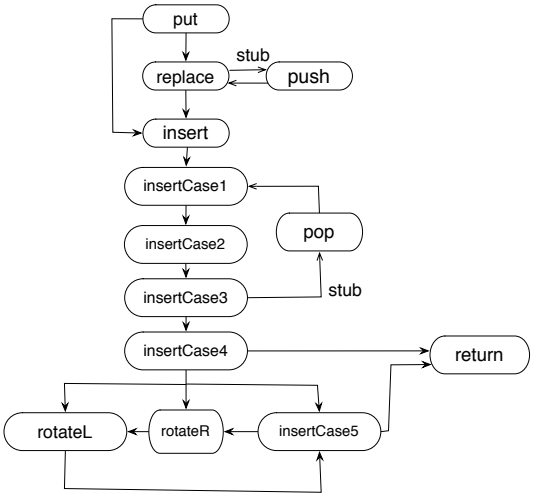

図 2: CbC での Red Black Tree

6 GPGPU

GPGPU とは画像処理に使われる GPU を画像処理以外で利用する技術である。GPU は CPU に比べコア数が圧倒的に多く、単純な計算しかできないが並列処理において一度に大量の計算ができるため利用されている。Gears OS は Code Gear に分割され処理の依存関係が明確になるので、並列な処理を書きやすい。また Data Gear へのアクセスは接続された Code Gear からのみであるから、処理中に変数が書き変わる事がない。

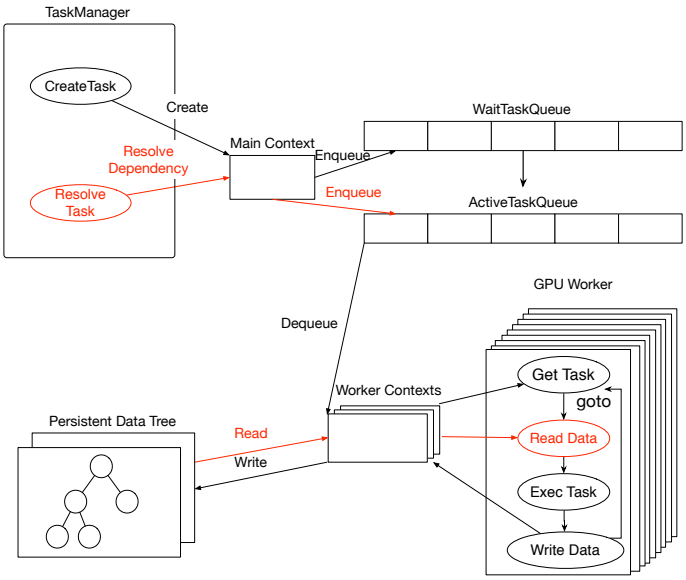

図 3: Gears OS による GPGPU

7 今後の課題

今回、Gears OS の GPGPU 処理についての基本的な設計と必要な機能の一部を紹介した。今後の課題としては、Gears OS による GPGPU 処理を実装する。並列化処理に最適な Mapping をほどこす。Cerium、Gears での Many CPU の並列化処理と比較しながら、GPGPU 処理の測定、評価を行う。

参考文献

[1] 宮國 渡, 河野真治, 神里 晃, 杉山千秋：Cell 用の Fine-grain Task Manager の実装, 情報処理学会システムソフトウェアとオペレーティング・システム研究会 (OS) (2008).

[2] 赤嶺一樹, 河野真治：DataSegment API を用いた分散フレームワークの設計, 日本ソフトウェア科学会第 28 回大会論文集 (2011).

[3] 伊波 立樹, 東恩納 琢偉, 河野 真治:Code Gear、Data Gear に基づく OS のプロトタイプ、情報処理学会システムソフトウェアとオペレーティング・システム研究会 (OS) (2016).

[4] 河野真治, 杉本 優：Code Segment と Data Segment によるプログラミング手法, 第 54 回プログラミング・シンポジウム (2013).

[5] Eugenio Moggi, Notion of Computation and Monads(1991)

[6] 徳森海斗, 河野真治：Continuation based C の LLVM/clang 3.5 上の実装について, 情報処理学会シ

ステムソフトウェアとオペレーティング・システム研究会 (OS) (2014).

[7] Aaftab Munshi, Khronos OpenCL Working Group: *The OpenCL Specification Version 1.0* (2007).

[8] : CUDA, https://developer.nvidia.com/category/zone/cuda-zone/.