Code Segment と Data Segment によるプログラミング手法

河野真治[†]　杉本優[†]

本研究室では分散プログラミングにおいて、データを Data Segment、タスクを Code Segment という単位に分割して記述する方法を提唱している。しかし、前述した方法をプログラマーが一から記述することは大変である。そこで、本研究室で分散ネットフレームワーク Alice を作成した。本論文では実際に Alice を用いて Code Segment と Data Segment によるプログラミング手法の例を示す。

How to Programming with Code Segment and Data Segment

Shinji Kono[†] and Yu Sugimoto[†]

1. 歴史的経緯

本研究室では、並列タスク管理フレームワーク Cerium の管理を行なっているが、その設計と実装を行うにあたり、並列プログラミングと分散プログラミングは本質的には同じことを行なっていることが分かった。特にヘテロジーニアスマルチコアであり、ローカルストアをそれぞれのコアが持っているという Cell の特異な環境は、分散プログラミング環境と告示している。それらを踏まえて、Cerium のタスク管理手法を分散に応用できないかと考えた。

Cerium に置いても、タスクとそれの入出力データの取り扱いは難しい課題である。データを操作する API を考えて、タスクを自然に記述し、効率良く実行する方法を考えてきた。そこで考えられた手法が、データを DataSegment、タスクを CodeSegment という単位に分割して記述する方法である。

2. 分散ネットフレームワーク Alice

2.1 Alice とは

Alice は本研究室の卒業生である赤嶺一樹氏が、本研究室で開発を行なっている並列タスク管理フレームワーク Cerium と先行研究である Federated Linda の開発を通して得られた知見を生かされている。Federated Linda の設計はシングルスレッドで行われている。しかし、近年ではマルチコアのマシンが主流となっている。将来的にはメニーコアのマシンが主流になってくると考えられるそのような背景を踏まえて Alice はマルチスレッド向けに設計されている。

Alice は Data Segment と Code Segment という単位でデータと処理を細かく分割し、それぞれの依存関係を記述して分散プログラムを作成する。また、他のマシンとの接続トポロジーの構成の機能も有しているのでユーザーはトポロジー構成後の処理を記述するだけでよい。

2.2 Data Segment

Alice では Data Semgnet をデータベースとして利用している。KeyValueStore で実装されており、キーごとにリストを持っている。Data Segment API を用いることで、リストにデータを追加、削除を適宜行うことができる。

2.2.1 Data Segment Manager

大量の Data Segment を管理するのが Data Segment Manager である。Data Segment Manager は文字列のキーで Data Segment を整理する。各キーごとにキュー構造を持っている。それらを Data Segment API を用いて操作する。データの読み出し (”peek” または ”take”) 時に、希望のデータがなかった場合、ブロッキングを行う機能を持つ。しかし、ブロッキングといってもそこで同期するわけではない。非同期でデータを通信する。そのため、 ”peek” と ”take” は他の API とは違い、レスポンスが発生する。

2.2.2 Data Segment API

表番号) が用意されている Data Segment API である。これらを用いてデータの送受信を行う。

† 琉球大学
University of the Ryukyu

- `void put(String key, Value val)`
- `void update(String key, Value val)`
- `void peek(Receiver receiver, String key, int id)`
- `void take(Receiver receiver, String key, int id)`

”put”

”put” はデータを追加するための API である。

”put” は受け取ったデータ val を Data Segment 内のキューに対してエンキューする。この時、キーごとに重複しない連番の ID を受け取った順に振る。(図 **??**)

”update”

”update” はデータを置き換えるための API である。

”update” はキューの先頭にあるデータをひとつだけ削除する。その後は ”put” と同じく、 受け取ったデータ val を Data Segment 内のキューに対してエンキューする。この時、キーごとに重複しない連番の ID を受け取った順に振る。(図 **??**)

”peek”

”peek” はデータを読み込むための API である。

”peek” は前回読み込んだデータの id を引数で指定する。省略した場合は、 0 が id として渡される。id よりも値の大きい id のデータがキューに含まれていれば、そのデータを receiver に返す。もし id 以下のデータしか無いならば、データの更新が前回の ”peek” 発行時から更新が無いものと考え、リストに格納されて保留される。(図 **??**)

”take” や ”update” によりデータの更新があれば、”peek” が直ちに実行される。

”take”

”take” もデータを読み込むための API である。基本的な id に関する部分は ”peek” と同じである。

”peek” との決定的な違いは、読み込まれたデータは Data Segment 内のキューから取り除かれるということである。(図 **??**)

2.3 Code Segment

Code Segment はタスクのことである。Code Segment をユーザーが記述するときに、Code Segment 内で使用する Data Segment を記述し、依存関係を作る。依存関係により、実行される順番が一意に決まる。実際に使用する Data Segment は Code Segment の入出力に相当する。それぞれ、Input Data Segment、Output Data Segment とする。

2.3.1 Code Segment の実行方法

Code Semgnet を実行するためには Start Code Segment という Code Segment を実行させる必要がある。Start Code Segment はどの Data Segment にも依存しない。つまり Input Data Segment を持たない。この Code Segment を main メソッド内で new し、execute メソッドを呼ぶことで実行を開始させることができる。

2.3.2 Code Segment の記述方法

Code Segment をユーザーが記述する際には CodeSegment を継承して記述する。その CodeSegment は InputDataSegmentManager と OutputDataSegmentManager を利用することができる。

InputDataSegmentManager

InputDataSegmentManager は Code Segment の ids というフィールドを用いてアクセスする。

- `Receiver create(CommandType type)`

create でコマンドが実行された際に取得される Data Segment が格納される受け皿を作る。引数には CommandType が取られ、指定できる CommandType は PEEK または TAKE である。

- `void setKey(String managerKey, String key, int id)`

setKey メソッドにより、どこの Data Segment のある key に対して peek または take コマンドを実行させるかを指定することができる。コマンドの結果がレスポンスとして届き次第 Code Segment は実行される。

OutputDataSegmentManager

OutputDataSegmentManager は Code Segment の ods というフィールドを用いてアクセスする。OutPutDataSegmentManager は”put”または”update” を実行することができる。

- `void put(String managerKey, String key, Value val)`
- `void update(String managerKey, String key, Value val)`

2.4 Topology Manager

TopologyManager は Alice 同士の接続トポロジーを管理する。TopologyManager 関連の通信処理は Code Segment で実装してある。TopologyManager はトポロジーファイルを読み込み、参加を表明したクライアント (以下、Topology Node) に接続するべきクライアントの IP アドレスやポート番号、接続名を送り、トポロジーファイルに記述された通りにトポロジーを作成する。

2.5 Topology Manager の設定ファイル

Topology Manager はトポロジーファイルを読み込むが、トポロジーファイル自体は DOT Language と

いう言語で記述される。DOT Language とはプレーンテキストを用いて、データ構造としてのグラフを表現するための、データ記述言語の一種である。この DOT Language のグラフを利用して、クライアント間の接続を表現する。DOT Language ファイルは dot コマンドを用いて、グラフの画像ファイルを出力することができるので、記述したトポロジーが正しいことを可視化して確認することができる。

クライアント間の接続には label を用いて名前が割り振られており、この接続名を用いてユーザーは Data Segment Manager にアクセスすることができる。前述した Receiver に setKey を行う際、ods で put または update する際の引数の managerKey がこれにあたる。

2.6 Topology Manager の使用方法

Topology Node を起動する際にコマンドライン引数として Topology Manager の IP アドレスとポート番号を指定をする。そして main 関数内で TopologyNode を new を行えば良い。TopologyNode の第一引数は Alice デーモンの設定オブジェクト、第二引数は Start Code Segment である。ここで指定した、Start Code Segment がトポロジーが完成した後実行される。

3. ゲームの例題

3.1 水 族 館

今回作成した例題は水族館である。複数のクライアントのディスプレイを複数の魚が移動していくものである。魚は画面の端まで移動すると自分の画面上からは消え、別のクライアントの画面の端から魚が出てくる。また、魚のうち一匹はクライアントが直接操作することができる。トポロジーは TopologyManager によりツリー状に構成してある。

3.2 データの伝搬

(1) ユーザーが魚を操作するまたは Code Segment により魚の座標が更新される。
(2) 画面に表示させるための SetLocation (Code Segment) が実行され実際に魚のオブジェクトにセットされ画面に反映される。
(3) Update(Code Segment) に FishPosition(魚の座標データ) が渡される。
(4) Update に list(送信者リスト) が渡される。
(5) Update が実行され、list を元にデータが送信される。ただし、この時に FishPosition には送信元情報が付加されているので、送信元には送信されない。
(6) 各 client で 2 - 4 が実行される。

4. 評 価

5. まとめと今後の課題