分散フレームワーク Christie を用いた remote editor
Remote editor using distributed framework Christie

165713F 一木貴裕　指導教員：河野真治

1　複数人によるファイルの同時編集

ペアプログラミングを行う際に有効的手法の一つとして, 同じファイルを複数人が場所を問わずに同時編集することができるリモートエディタをあげられる.

複数人の編集がリアルタイムに同期されるリモートエディタには, 既存の物として Visual Studio Code(以下 VScode) の remote session がある. しかし, 編集のセッションに参加するユーザ全員が VSCode を使うことになり, また VSCode の環境を導入する必要がある.

セッションに参加するユーザー全員が各々好きなエディタを使用することができるリモートエディタアプリケーションを作成したい. アプリケーションの形に作成することにより, 開発のための環境に手間を使うことなく, ユーザーが慣れ親しんだエディタを利用できるようしたい.

先行研究ではネットワークをリング型で構成しトークンを巡回させていたが, ノードごとの整合性の確立が難しい, ネットワーク全体の障害に対する脆弱性の弱さといった問題点が見られた. これらの反省点を踏まえ本研究では スター型ネットワークを用いることで remote editor の障害耐性を高める.

また新しく, 本研究室で開発している分散フレームワーク Christie を使用することにより簡潔な実装と, Christie 自体の性能と信頼性の向上も目指す.

2　remote editor

リモートエディタは共通プロトコルが対応するエディタが保持するバッファを開いて編集することができる. ネットワーク上の一つのバッファが編集されると他のバッファにも変更が反映され, お互いのバッファを編集し合うことができる. バッファの変更のやり取りでは, 互いのバッファに起こった変更を一つの命令として他のノードへ送信することで実現する.

3　コマンドパターンでの実装

命令はコマンドパターンを用いて実装した. コマンドパターンとは命令を一つのオブジェクトとして扱うものであり, 命令に必要な情報が含まれたクラスのインスタンスを作成することで命令を作成する.

- インスタンスを利用して命令を作成するため, 後述の Christie の Gear の概念と相性が良い.
- 命令に必要な内容をまとめて送信するため, 相違の発生を防ぐことができる.
- 命令の管理が行いやすい, 行列に並ばせ命令の順番を管理したり, 命令の際, 実行, 取り消しが容易になる.

と言った利点がある. クラスのインスタンスを作成し, msgpack パッケージを用いてシリアライズしてから送受信する。

4　msgpack の対応修正

msgpack を使用し, コマンドを送信しようとしたところエラーの発生が見られた. 原因は msgpack のバージョンの進行によりシリアライズ機能が削除されたことによることが判明した.

以上の修正のため javassist のバージョンを上げる, シリアライズするクラスに対してフィールドを public にするといった処理を行うことで対処した. msgpack はバージョンの更新が途絶えているため, 最新の java で開発を進めるためには msgpack に対応する機能を自身で開発する必要が生じた.

5　編集位置の相違とその解消

エディタ間の通信で生じる相違について説明する. エディタ同士のコマンドの送信はそれぞれが独立して行うため、編集対象の領域にエディタ間で相違が生じる場合がある。すれ違いが発生したか否かを検知するために、ノード同士が送信し合うコマンドにそれぞれ番号を割り振る方法を考案した. 図 1 はコマンド番号を用いてバッファの相違を解決する過程の図である.

コマンドとコマンド番号について以下の特性が存在する.

- 各ノードは最後に実行した数値を変数 (図では server が cNum, node が nodeCNum) に保持している.
- いずれかのノードがコマンドを発行したら, そのコマンドのコマンド番号は前に実行したコマンドの番号+1 となる. そしてそれは送信されてきたコマンドか自分が発行したコマンドであるかは問わない.

- 送信されてきたコマンドのコマンド番号が自身が保持しているコマンド番号+1 でなければ、自身が先に発進したコマンドとすれ違いが発生していることが判明できる. (保持コマンド番号より同値以下の場合)

以上の仕組みで相違の解消を図る.

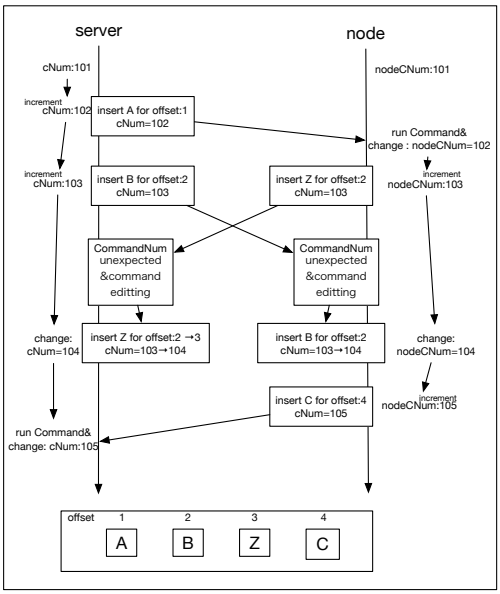

図 1: コマンド番号による編集相違の解消

6　スター型ネットワーク

リモートエディタのセッションに参加するノード (ユーザ) はスター型で接続を行い, リモートエディタの通信部分の障害に対する耐性を保障する. スター型とは中心となるノードから放射状に他のノードにそれぞれ一対一の接続を行う接続であり, リング型と比較した際のスター型の利点として,

- ノードの中心 (サーバー) が正しいファイル状況を保持するため, 整合性を保つことが容易である.
- どこかのノードの接続が切断されても, 障害の範囲をそのノードのみに抑えることができる.
- 新しいノードが参加した, もしくはノードの再接続の際にはサーバーのファイル状況を参照するのみで参加, 復帰ができる.

が挙げられる.

7　分散フレームワーク Christie

Christie は Java 言語で構成された本研究室独自の分散フレームワークである. 同じく本研究室で開発を行っている GearsOS のファイルシステムに組み込む予定があるため, GearsOS を構成している本研究室の独自の言語 Continuation based C (以下 CbC 言語) とにた, Gear というプログラム概念が存在する.

- CodeGear(以下 CG)
- DataGear(以下 DG)
- CodeGearManager(以下 CGM)
- DataGearManager(以下 DGM)

CodeGear はクラスやスレッドに相当する. DataGear は変数データに相当し, java のアノテーション機能を用いて記述する. CG 内に記述した Key に全ての DG が揃った際に初めてその CG が動作するという仕組みになっている. CodeGearManager はいわゆるノードに相当し, CG, DG ,DGM を管理する. DataGearManager は DG を管理するもので, put という操作により DG , つまり変数データを格納することができる.

DGM の put 操作を行う際には Local と Remote と 2 つのどちらかを選び, 変数の key とデータを引数に書く. Local であれば, Local の CGM が管理している DGM に対し, DG を格納していく. Remote であれば接続した Remote 先の CGM の DGM に DG を格納できる. put 操作を行ったあとは, 対象の DGM の中に queue として保管される. DG のアノテーションには Take, Peek, TakeFrom, PeekFrom の 4 つがあり, Take は DG が一度読み込まれると中身が削除されるが, Peek は削除されない. From をつけるとリモート先を指定して DG を得ることができる.

8　今後の課題

本研究ではリモートエディタを制作する上で必要となる構造と, 土台となる分散フレームワーク Christie について述べた.

現時点ではリモートエディタのコマンドを送り合う上での基盤となるコマンドパターンでの送受信を実装し, 命令コマンドのやり取りにて問題となってくる編集の相違の発生をテスト作成することまでができた. リモートエディタを動かすまでの最低限の構造制作までは終わっていない. 現時点から最低限のセッションができるようになるためには, 複数人が参加できる Star 型 Topology を構成すると編集ファイルの共有方法を作成しなければならない. 加えて, 本研究の制作物が実用的になるには既存のエディタを動作できるようになることは絶対条件である. これらを踏まえて, 今後も制作を続け, 本研究で述べた構成が実際に実用性に応えられるか, ユーザに苦痛なく使ってもらえる処理速度を得られるか検証していきたい.

参考文献

[1] 河野真治. 分散フレームワーク christie と分散木構造データベース jungle. IPSJ SIG Technical Report, May 2018.

[2] Kaito TOKKMORI and Shinji KONO. Implementing continuation based language in llvm and clang. *LOLA 2015*, July 2015.

[3] 照屋のぞみ. 分散フレームワーク christie の設計, March 2018.

[4] 安村恭一. 巡回トークンを用いた複数人テキスト編集とセッション管理, Merch 2004.

[5] 宮城健太. Remote editing protocol の実装, Merch 2008.

[6] 結城浩. デザインパターン入門, 2004.