4J-06

# Rust 言語による並列分散処理システムの開発と評価

比嘉 隆貴 † 市川 嘉裕 ‡ 山口 智浩 ‡
奈良工業高等専門学校 専攻科 システム創成工学専攻 †奈良工業高等専門学校 情報工学科 ‡

## 1 背景と目的

並列分散処理システムは，複数台のコンピュータおよび複数のスレッドを利用し，大量のデータを高速に処理するシステムである．代表的な並列分散処理システムである MapReduce[1] を用いた Apache Hadoop や RDD[2] を用いた Apache Spark は初期リリースから年月が経ち，改良が日々施されている．しかし，言語システムやアーキテクチャの技術向上に沿って，コンパイル時の最適化と柔軟性の担保やリソース使用の最適化が行われず，最大限の性能が発揮できていないと考えられる．

本研究では所有権とライフタイムと呼ばれる概念を備えた Rust 言語を用いて開発を行い，低レベルな制御とメモリを安全かつ効率的に使用することで，速度向上が可能か検証する．

## 2 提案システム

提案システムでは，RDD[2] を処理システムの基礎に採用し，Rust 言語を用いて並列分散処理システムの開発を行う．Rust 言語は比較的新しい言語で，所有権とライフタイムによって，安全でかつ高速に処理するコードを書くことができる言語である．また，並列処理や非同期処理に tokio ライブラリを使用する．

### 2.1 RDD

RDD は読み込み専用の分割されたインメモリのレコードであり，決定的な操作によって RDD から新たな RDD を生成し，その連鎖反応によって分散処理を行うことが可能である．

### 2.2 処理方法

ユーザーは，Listing 1 のように手順を定義し，AppClient を通じてシステムに処理を依頼する．

Listing 1 処理の定義

```
use proposed::{AppClient, Map, map::function};

fn main() {
    let client = AppClient::new();

    let map = Map::no_input()
        .create("x", vec![function::random(0, 1),
             function::mul(2), function::sub(1)])
        .create("y", vec![function::random(0, 1),
             function::mul(2), function::sub(1)])
        .target("x", vec![function::pow(2)])
        .target("y", vec![function::pow(2)])
        .add_target("z", "x", "y")
        .output("z");

    let rdd = client.generate_rdd(1000000, 300, map)
}
```

手順データは，シリアライズされてシステムに送信される．送信されたデータはデシリアライズされて，その手順に従ってシステム内で定義されている関数をスケジューラーを通して実行する．なお，定義済み関数は`#[target_feature]` 属性を用いて，LLVM において AVX 命令を用いた最適化が行われる．最終的に，RDD からデータを取り出す際は，ファイルシステムを通じて取り出す．

### 2.3 スケジューリング

システムの起動時に事前に高負荷の処理を行い，スケジューラーに処理時間をノードのスコアとして保持する．スケジューラーは，分散処理を行う際に，手順データの依存関係から有向非巡回グラフを生成する．その後，そのグラフを元に，別の RDD を必要としない依存関係の少ない処理から，ノードのスコアが小さい順に処理を割り当てる．

### 2.4 データの管理

RDD や生成された DAG を含むデータは，すべてスケジューラーが管理する．分散処理を行う際，処理の依存関係によって RDD がメモリに乗らない場合がある．その場合は，依存関係が少ない順に RDD をファイルシステムに書き込む．

## 3 実験

2 章にて述べた提案システムの性能を評価するため，Apache Hadoop 3.3.6 と Apache Spark 3.5.0 との性能比較を行った．実験環境は表 1 の通りで，通信には 10Gbps のイーサネットを，OS は Arch Linux を用いた．

並列分散システムは Ansible を用いて構築を行い，ファイルシステムには HDFS を用いた．なお，Apache Hadoop と Apache Spark はメモリとスレッド関連の設

---

Proposal and Evaluation of a Parallel and Distributed Processing System Using Rust

† Ryuki Higa・Department of Systems Innovation, Faculty of Advanced Enginnering, National Institute of Technology, Nara College
‡ Yoshihiro Ichikawa, Tomohiro Yamaguchi・Department of Information Enginnering, National Institute of Technology, Nara Collage

表1　クラスタ構成

| Hostname | CPU | Memory | M.2 SSD | HDD |
|---|---|---|---|---|
| cluster-01 | Ryzen9 7950X | 64GB | 1TB | 2TB |
| cluster-02 | Ryzen5 5600X | 32GB | 512GB | - |
| cluster-03 | Core i9 11900K | 32GB | 1TB | 2TB |
| cluster-04 | Core i9 11900K | 32GB | 1TB | 2TB |

定以外はデフォルトの設定を使用し，Apache Hadoop YARN を用いてリソース管理を行った．また, cluster-01 をマスターノードとして用いた．

### 3.1　円周率計算

ここでは，モンテカルロ法を用いて円周率の計算を行う．ランダムに 1 億点打つことを 1 単位の処理とし，300 単位を分散処理で行う．すべてのシステムで 100 回測定した結果を箱ひげ図として，図 1 に示す．

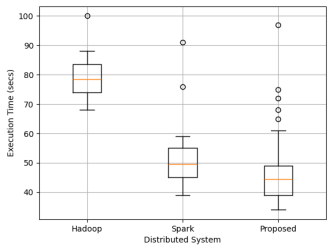


図 1　円周率計算の実行時間

図 1 より, 提案システムは Apache Hadoop や Apache Spark に比べて，高速に処理を行うことができた．しかし，いくつかのケースにおいて，提案システムは遅くなることがあった．また，すべてのケースで小数点第 5 位まで円周率と一致することを確認した．

### 3.2　分散 Grep

ここでは，1 行 1024 文字で構成された 1GB のファイル 400 個に対して，文字列'foo' を含む行を検索する．なお，文字列探索には Knut-Morris-Pratt 法を用いた．すべてのシステムで 100 回測定した結果を図 2 に示す．

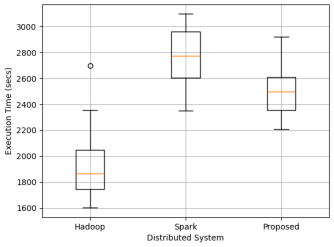


図 2　分散 Grep の実行時間

図 2 より，提案システムは Apache Spark に比べて，やや高速に処理を行うことができたが，Apache Hadoop に比べて，遅いことがわかる．これは，メモリ上にデータが乗り切らず，ファイルシステムに書き込む必要があるためと考えられる．

### 3.3　任意の演算

ここでは，40 億個の 64bit 整数が改行で区切られた，約 30GB のファイル 2 つを読み込み，それらを同一行で加算を行った後に，その結果の重複を削除して出力を行う．Apache Spark と提案システムで 100 回測定した結果を図 3 に示す．なお，Apache Hadoop では重複削除の処理が適切に行えなかったため，結果を示さない．

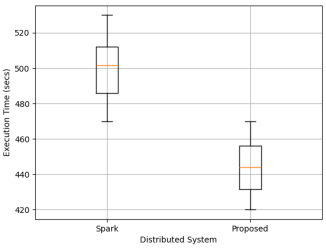


図 3　演算と書き込みの実行時間

図 3 より，提案システムは Apache Spark に比べて，高速に処理を行うことができた．

## 4　まとめと課題

本研究では，Rust 言語を用いて並列分散処理システムを開発した．そして，提案システムの性能を評価するために，Apache Hadoop と Apache Spark との性能比較を行い，特定の処理においては，高速に処理が行うことができることを確認した．

ただ，既存の並列分散処理システムと比較して，システムに対して柔軟な処理を行えない点や，機能が乏しい点などの課題点があり，全体的に優位性があるとは言い難い．今後は，RDD の機能を拡張し，より高度な処理を行えるようにする．また，groupBy などの依存関係が複雑な演算について実行できるように，検証を行う．

## 参考文献

[1] J. Dean, S. Ghemawat, MapReduce: Simplified Data Processing on Large Clusters, OSDI' 04, 2004.

[2] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. Franklin, S. Shenker, I. Stoica, Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing, NSDI' 12, 2012.