Hadoop 上で動作する Sawzall サブセット

中田秀基[†] 井上辰彦[††,†] 小川宏高[†]
高野了成[†] 工藤知宏[†]

1. はじめに

容易に並列計算を記述できる実行パラダイムとして MapReduce[1] が注目されている。MapReduce は Hadoop を始めとする処理系の普及によって、科学技術計算のみならず業務データの解析などに広く用いられつつある。われわれは、MapReduce の高速な実装[2] を進めるとともに、MapReduce を汎化した計算機構の検討を行い、さらに汎化計算機構上の計算記述を補助するための言語処理系の開発を行っている。

この言語処理系開発の 1 ステップとして、Google による MapReduce 向け言語である Sawzall のサブセット (以下 SawzallClone) を実装し Hadoop 上で実行できる環境を実装した。SawzallClone は Java を中間言語として用いるコンパイラとして実装した。構文解析に Scala の Parser Combinator を用いることで、処理系の記述量が削減できた。

Hadoop 上では、mapper として Sawzall スクリプトから生成されたコードを、reducer として Scala で実装したアグリゲータを実行する。Hadoop の Java コードで直接記述した場合と比較し、実行速度の面で一定のオーバヘッドがあることを確認した。また、単体実行では Google による Sawzall のオープンソース実装 szl[3] と、ほぼ同等の実行速度であることを確認した。

2. Sawzall

Sawzall[4] は、Google が内部的に利用しているデータ処理用の言語で、Google の MapReduce 実装上で稼働している。Sawzall は、MapReduce モデルの Mapper と Reducer のうち、Reducer を Sawzall が提供する固定的なアグリゲータで提供し、ユーザには Mapper のみを記述させる。ユーザが記述する範囲を限定することによって、非技術者のユーザでもさらに容易に並列プログラミングを行うことを可能にしている。

```
proto "p4stat.proto"
submitsthroughweek: table sum[minute: int] of count: int;

log: P4ChangelistStats = input;
t:      time = log.time; # microseconds
minute: int  = minuteof(t) +
               60*(hourof(t)+24*(dayofweek(t)-1))
emit submitsthroughweek[minute] <- 1;
```

図 1 Sawzall プログラムの例

Sawzall は型付きの言語である。Sawzall コードはマップの入力データ 1 レコードに対して処理を行う。複数のレコードにまたがった処理を記述することはできない。レコード単位で処理するという点は、テキストファイルを 1 行ずつ処理する Awk 言語と類似している。ただし、Awk では行単位の処理を行いながら内部状態を更新することで、文書全体に対する処理を行うことができる。これに対して、Sawzall では言語処理系内部の状態はレコードごとにリセットされる。文書全体に対する総計処理はアグリゲータ部分で行われる。

図 1 に、文献[4] より抜粋した Sawzall によるログ解析プログラムの例を示す。このプログラムは、Protocol Buffers 形式で格納されたログデータの頻度を分単位で集計するものである。

3. 設計と実装

図 2 に、SawzallClone の概要を示す。SawzallClone は、以下の 4 つのモジュールから構成される。すべてのモジュールは Scala で記述されている。

メインプログラム Hadoop のクライアントプログラムとして機能する。コンパイラを起動して、Sawzall スクリプトを Java ソースに変換、さらに Java の class ファイルへコンパイル、jar ファイルにまとめて Hadoop に投入する。

コンパイラ Sawzall スクリプトを Java のソースコードに変換する。Parser Combinator を利用して

† 独立行政法人 産業技術総合研究所
National Institute of Advanced Industrial Science and Technology
†† 株式会社創夢
SOUM Corporation

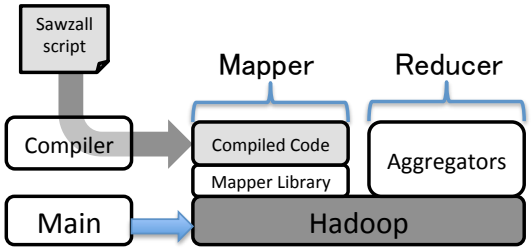

図 2 SawzallClone の実装

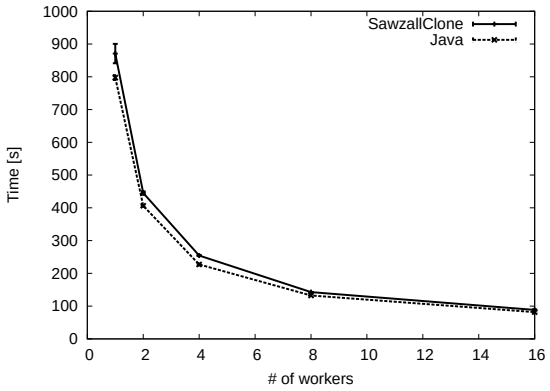

図 3 Java との実行速度比較

いる。

Mapper ライブラリコンパイラの生成したコードをラップして、Hadoop の Mapper として機能する。Hadoop からの入力を変換してコンパイルされたコードに引き渡す。

アグリゲータHadoop の Reducer として機能する。集計演算を受け持つ。

4. 評　　価

4.1 Java との比較

SawzallClone 処理系のオーバヘッドを知るために、Hadoop の Java API を用いたプログラムと、Sawzall で記述したプログラムを比較した。対象プログラムはログ解析を行うプログラムである。ノード数を 1,2,4,8,16 と変化させて比較した。

SawzallClone のほうが 5 %から 10 %程度実行時間が長い。これが SawzallClone によるオーバヘッドであると考えられる。

4.2 szl との比較

Google のオープンソース Sawzall 処理系 szl との比較を図 4 に示す。szl は並列実行をサポートしていないため、単一 CPU での比較とした。対象プログラムはログ解析を行うプログラムで、ログのレコード数を変化させて実行時間を計測した。

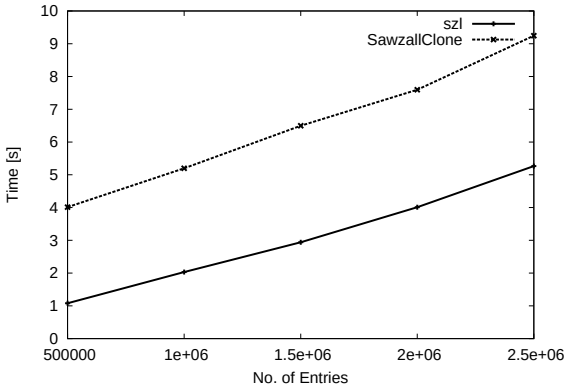

図 4 szl との比較

SawzallClone は szl よりも実行時間が長いが、差異はログレコード数に依存せず定数であることが分かる。このことから実行時間の差は主に Java VM の起動と JIT コンパイルによるものであり、起動後の実行速度はほぼ同等であることがわかる。

5. おわりに

Sawzall サブセットの Hadoop をターゲットとした処理系 SawzallClone について述べ、性能を評価した。

今後の課題としては、1)SSS 処理系への適用、2)Reduce フェイズ記述への拡張、3) 複数段の Mapper および Reducer によるデータフロー記述への拡張、が挙げられる。

参 考 文 献

1) Dean, J. and Ghemawat, S.: MapReduce: Simplified Data Processing on Large Clusters, *OSDI'04: Sixth Symposium on Operating System Design and Implementation* (2004).
2) Ogawa, H., Nakada, H., Takano, R. and Kudoh, T.: SSS: An Implementation of Key-value Store based MapReduce Framework, *Proceedings of 2nd IEEE International Conference on Cloud Computing Technology and Science (Accepted as a paper for First International Workshop on Theory and Practice of MapReduce (MAPRED'2010))*, pp. 754–761 (2010).
3) Szl - A compiler and Runtime for the Sawzall Language, http://code.google.com/p/szl/.
4) Pike, R., Dorward, S., Griesemer, R. and Quinlan, S.: Interpreting the Data: Parallel Analysis with Sawzall, *Scientific Programming Journal, Special Issue on Grids and Worldwide Computing Programming Models and Infrastructure*, Vol. 13, No. 4, pp. 227–298 (2005).