3B-01

Spark/Hadoopを用いたユーザアクセスデータの解析基盤構築

佐藤 哲 †

NHN comico 株式会社 データ研究室 †

1 はじめに

ネット上のサービスのログデータの分析は，サービス企業にとって重要な問題である．ノイズや欠損値が含まれ，それがまた故意の場合も有り得る．目的の一つはマーケティングのためにユーザのプロパティを推測することであり，インターネット業界ではベイズの枠組みでの統計的推論の理論が多く活用されている．またデータ量の増大から，Apache Hadoopが利用されることが一般的となってきている．そこで本発表では，Apache Hadoop 上の YARN リソースマネージャを活用した Apache Spark を用いたデータ分析基盤を構築した事例を紹介する．以下，団体名称である Apache は省略する．

2 ユーザログデータの解析基盤

弊社ではログのフォーマットは RDB 形式または Web ログ形式の 2 種類があり，まず ETL 処理によりフォーマットを統一し，その後，統一的な処理システムで分析することが標準的な処理である．

従来は，開発の高速化及びメンテナンスのために Hadoop Streaming を使うことが多かった．しかし，データ量の増大から，我々は Hadoop の MapReduce 処理から，Spark の RDD（Resilient Distributed Dataset）を中心とした DAG（Directed Acyclic Graph）に移行することにした．RDD は，処理の中断にも堅牢なメモリベース（ディスクに書き出すことも可能）な独自のデータ構造で，メモリベースであることからキャッシュのようなものだと考えても良い．DAG は，MapReduce と比べて複雑なデータフローを実現する処理モデルである．

データを解析する手法は莫大な種類があるが，弊社で重視しているものは大きく分けてクラスタリングによる類似ユーザの抽出，ユーザデータの時系列分析による数値予測の 2 つがある．クラスタリングにはユニバーサル符号化の理論を用いた汎用的な手法を採用しており，様々なログデータに適用可能な他，異常値検出にも使用している．時系列分析には粒子フィルタを採用しており，モデル化にはエンジニア・データアナリストだけでなく経営・営業側の知見も採用できるようになっている．

3 MapReduce 及び Spark によるユーザデータの類似度計算の実装

データ分析としてはデータのクラスタリングを行うが，クラスタリングはデータ間の類似度が計算できればデータ間の距離に応じて分類が可能なため，ここでは類似度計算までを説明する．類似度の計算には NCD(Normarized Compression Distance) を用いた手法を採用している [1]．NCD は，以下の様な式で定義される．

$$NCD(x,y) = \frac{Z(xy) - \min(Z(x), Z(y))}{\max(Z(x), Z(y))}$$

ここで，$Z(x)$ は文字列 x を圧縮した後の長さであり，$Z(xy)$ は文字列 x と y を結合して圧縮した後の長さである．NCD は距離であるので，NCD の値が小さいほど類似度が高くなる．

我々が従来の MapReduce を用いて実装した際は [2]，以下の様なアプローチを用いた．なお，分散処理の複数のプロセスから同一のデータソースにアクセスするため，メモリベース KVS である HBase を用いている．

(1) ログをパースし，ユーザ毎のアクセスパラメータを抽出し，ユーザ ID をキーに HBase にアクセスパラメータを格納する
(2) アクセスパラメータを，ユーザ毎に集約する（ユーザデータがログファイル内に分散されているため，全ログファイルの参照が必要であり，処理速度のネックとなる）
(3) ユーザ毎のアクセスログから，ユーザ毎のログの圧縮サイズを計算し，HBase に格納する．ただし，圧縮サイズが十分に大きくないものは除外処理をする
(4) ユーザ毎の圧縮後ログサイズより NCD を用いてユーザ同士の類似度を計算し，HBase に格納する

Spark での実装では，以下のようなアプローチを用いた．

(1) アクセスログを RDD 化する
(2) ログをパースし，ユーザ毎のアクセスパラメータを抽出し，RDD 化後にユーザ ID をキーに HBase にアクセスパラメータを格納する
(3) アクセスパラメータを，ユーザ毎に集約し RDD 化する
(4) ユーザ毎のアクセスログから，データサイズが十分に大きくないものは除外処理をし結果を RDD 化する
(5) ユーザ毎のアクセスログから，ユーザ毎のログの圧縮サイズを計算し，HBase に格納する
(6) HBase の RDD を作成後，RDD を用いてユーザ毎の圧縮後ログサイズより NCD を用いてユーザ同士の類似度を計算し，HBase に格納する

MapReduce ではステップが終了する度に結果を全てディスクに書きださなければならないため，処理に工夫が必要である．一方 Spark ではディスクに書き出さずに連続的に処理することが可能であるが，メモリ不足が発生し処理が停止することが現実には起こり得る．

Data Analysis Platform for User Log Data using Apache Hadoop and Apache Spark

†Tetsu R. Satoh，NHN comico Corporation

4　Sparkによる粒子フィルタの実装

粒子フィルタは次々と入力されるデータから学習を繰り返し，予測や平滑化等を行う [3]．Sparkにより実現する場合の実装の概要を以下に示す．

(1) 粒子を以下で定義されるシステムモデルよりRDDの形で生成する

$$x_k = F(x_{k-1}, v_k)$$

ここで，x_k は時刻 $t = t_k$ での推定値，v_k はノイズである．

(2) 粒子と入力データから，各粒子の尤度を計算する．粒子の成分値はコレクションの形でRDDで保持されているため，分散処理が可能である．入力データとしては，Spark Streamingを利用することも可能である．ここで粒子の尤度は，定義された観測モデル

$$y_k = H(x_k, \omega_k)$$

において，推定値と入力データがどれくらいの尤もらしさ一致しているかを計算する．ここで，y_k は時刻 $t = t_k$ での観測値，ω_k はノイズであり，尤度は関数 H の逆関数及びノイズの確率密度関数から計算される．

(3) 各粒子の尤度を元に，復元サンプリングを実施する．復元サンプリングは厳密に全粒子からのサンプリングを行うのでは無く，層化抽出と呼ばれる手法を使うため，各ワーカノードにて並列に実行が可能である．

(4) サンプリング後の粒子群を推定値としてRDDの形で保持する．

以上の処理は，MapReduceでも実装は可能である．ただし，全粒子からの再サンプリングの実現方法や処理毎の結果のディスク出力など，Sparkの実装に比べ難しい問題が生じる．

5　データの類似度計算を例にした実行時間比較

簡単な実験として，bzip2圧縮をした状態で約800Mバイトのデータを用意した．データにはユーザを識別するためのIDとアクセス時間，アクセスしたサービスの種類などが記録されており，このログからユーザ間の類似度を計算し，クラスタリングの前処理とすることが目的である．

MapReduceでは，ステップ1及びステップ2のログのパース等の処理に1950秒かかった．ステップ3のデータ圧縮には60秒，ステップ4の類似度計算には32秒であった．一方Sparkでは，ステップ1からステップ5のログのパース及びデータ圧縮の処理には561秒，ステップ6の類似度計算には85秒かかった．合計ではMapReduceが約34分，Sparkが約11分と処理速度に差が出た．その理由の例をデータの流れを例に説明する．まず，ログの行のパースのような単純な逐次処理では，SparkのDAGは図1のようになる．これはMapReduceによるものと全く同じである．一方，類似度計算の場合には2つのキーを用意してそれぞれのキーに対するデータを取得する必要がある．この場合は図2のようになり，2つのキーのペアを作成するための図が現れる（6つの処理のうち，3つ目と4つ目の間）．また，一つの処理が終了する毎に結果をディスクに出力することも必要ないため，6つの処理が連続的に行われていることも分かる．MapReduceでは，キーのデータを走査することで2つのキーのペアを作成し，一つのデータとして一時的に保存する処理が入っているため，データの流れとしてはこのようにシンプルにはならない．

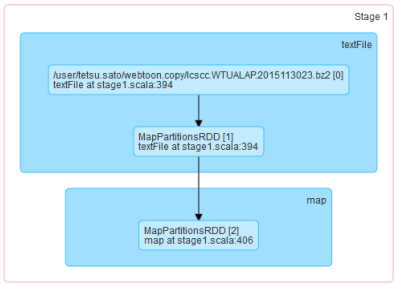

図 1: ファイルを読みだしパースする場合の DAG

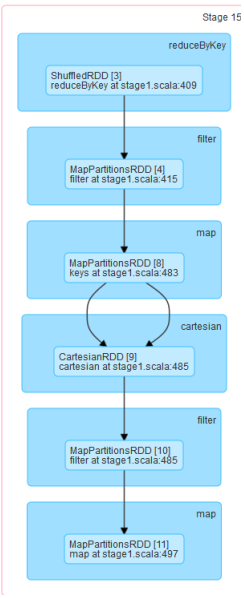

図 2: データの類似度を計算する場合の DAG

ここで用いたSparkクラスタは，データノードが6台の小規模なものであり，OSはCentOS6.5，CPUはXeon E5-2630L（2.00GHz），メモリは各64Gバイト，HDFSの総容量は89Tバイトである．

6　おわりに

Hadoopのリソース管理システム YARN を利用したSparkクラスタを構築し，その上で実行するデータ解析アルゴリズムであるクラスタリングに用いるデータの類似度計算，及び時系列分析である粒子フィルタの実装の概要を述べ，データの類似度計算を例にMapReduceに比べたSparkの優位性を紹介した．紙面の都合により，実装の概要に基づいた実験結果は発表時に紹介する．

参考文献

[1] P. M. B. Vitányi, Compression-Based Similarity, Proc. Int. Conf. Data Compression, Communications and Processing, pp. 111-118, 2011.

[2] 佐藤哲, Hadoopを用いた電子書籍ユーザのアクセスデータ分析, 第14回情報科学技術フォーラム，D-025, 2015.

[3] 北川源四朗, モンテカルロ・フィルタおよび平滑化について , 統計数理 , Vol. 44, No. 1, pp. 3148, 1996.