5M-07

半構造データに対するストリーム処理とバッチ処理の統合フレームワーク

長　裕敏[†]　　王　岩[‡]　　北川　博之[§]　　天笠　俊之[§]

[†]筑波大学情報学群情報科学類　[‡]筑波大学院システム情報工学研究科　[§]筑波大学システム情報系

1. 序論

代表的な大規模データ処理にはストリーム処理やバッチ処理があり，支援ツールとして前者にはStorm[1]や S4[2]，後者には Hadoop[3]等がある．利用者は，通常，対象データや処理要求に応じて両処理を選択的に組み合わせて使用することも多い．この時，利用者はそれぞれのツールに対応した処理プログラムやそれらを連携するプログラムを用意する必要があり，負担が大きい．

本研究では，半構造体データである JSON[4]を対象に，2 つの処理を融合したイベント駆動型処理記述法とその処理システムを提案する．提案フレームワーク中では，設計した処理記述を解析してストリーム処理記述とバッチ処理記述に分け，両処理を既存のツールで連携して処理する．具体的には，ストリーム処理ツールは筑波大学北川データ工学研究室で開発中の JsSpinner を，バッチ処理ツールには OSS の Hadoop を用いる。また，提案フレームワークにおける処理記述は Jaql[5]に基づく．提案するフレームワークを用いることで，利用者は単一の記述法に従った処理記述を用意するのみで両処理を組み合わせたデータ処理が可能となる．

2. 基本事項

2.1 JSON

JSON は JavaScript 言語のオブジェクトの表記方法であり，「変数名：値」のペアの集合，または値の順序付リストで表現される．以下に具体的な記述例を示す．

・{“name”：“Sato Taro”，“age”：25}
・{“language”：[“C++”，“java”，“PHP”]}

2.2 Hadoop

Hadoop は大量のデータを容易に複数マシンに分散して処理できるプラットフォームで，大きな特徴に MapReduce がある．MapReduce は並列可能な問題に対し，複数のマシンを用いて並列処理するフレームワークで，入力データを細かい単位に分割し複数マシンに分配・処理する Map 処理と，Map 処理で得た結果を集約し問題に対する答えを出力する Reduce 処理で構成され，大量のデータを高速に処理できる。

2.3 Jaql

Jaql は JSON 形式のデータを処理するクエリ言語で，Hadoop 上で動作する Jaql 記述処理系が提供されている[6]．処理式をパイプライン方式で記述でき，シンプルで短いコード量で JSON データに対してフィルタや結合，グループ化等の操作が行える．図 1 に Jaql クエリの動作例を示す。ここでは”users”というファイルから”income”が 25000 以上の”ID”と”name”を取り出し”result”というファイルに書き込むクエリを示している．

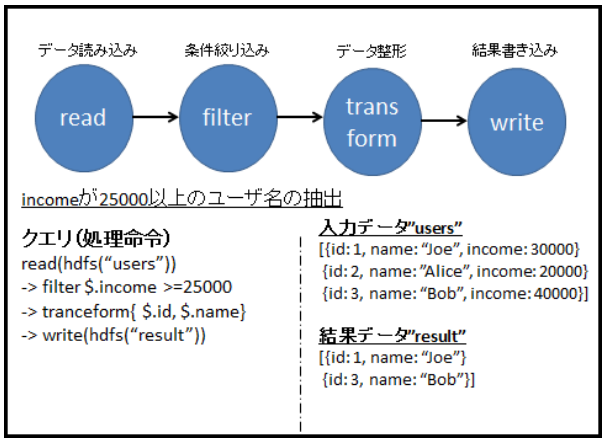

図 1　Jaql クエリの動作例

2.4 JsSpinner

JsSpinner は JSON に対応したストリーム処理ツールであり，現在当研究室で研究開発中である．Jaql クエリを実行可能で，window 演算を用いて無限長のデータを，生存期間を持つ n 項組の有限のデータ集合に変換することで，フィルタや結合等の処理をリアルタイムに行えるように設計されている．

3. 提案フレームワーク

3.1 フレームワーク概要

提案するフレームワークの全体像を図 2 に示す．利用者は，3.2 節に述べる記述法に従い，JSON 形式の静的データやストリームデータに対する処理が混在した処理記述を与える．処理記述解析系ではその内容を解析し，JsSpinner と Hadoop いずれで処理を行うか振り分け，両者の処理を連携させながら要

A framework for integrating stream and batch processing of semi-structured data
Hirotoshi Cho[†], Wang Yan[‡], Hiroyuki Kitagawa[§], Toshiyuki Amagasa[§]
[†]College of Information Science, University of Tsukuba
[‡]Graduate School of Systems and Information Engineering University of Tsukuba
[§]Faculty of Engineering, Information and Systems, University of Tsukuba

求された処理を実行する.

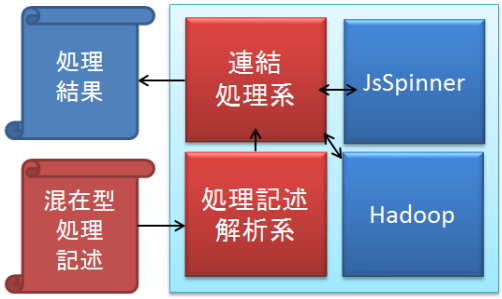

図 2 提案フレームワーク

表 1 扱える演算子群

データ取得演算	データ操作演算	データ出力演算
window, read, read-	filter, transform, expand, sort, top, group, udf, join	istream, rstream, dstream, write, write+

```
registerFunction("SplitAndChoice" , "udf1.SplitAndChoice");
registerFunction("EraceStopWords", "udf2.eraceStopWords");
registerFunction("AppendSplitWords", "udf3.appendSplitWords");
P1 = window("NewsStream", row 1) -> SplitAndChoice() -> write+("store1");
P2 = window("EventStream24h", row 1);
P3 = read-("store1") ;
P4 = join  P2, P3  where true into {P3.*} -> expand unroll $.word ->
      group by w = $.word into {word : w , ct : count($)} ->  EraceStopWords() ->
                                  sort by [$.ct desc] -> top 30  -> write("store2");
P5 = window("TwitterStream", row 1) -> transform {$.text}
                                        ->AppendSplitWords();
P6 = read("store2");
P7 = join P5 , P6 where P5.word == P6.word into {P5.text, P6.word}  -> rstream;
```

図 3 提案処理記述例

3.2 イベント駆動型処理記述法

設計した処理記述法について例を交えて述べる.利用者は表 1 の演算子群を組み合わせて処理を記述する.実際の処理記述の例を図 3 に示す.処理内容は「最近のニュースに関連があると思われるツイートの取得.」,具体的には「ニュースデータを取得し続け,テキストを単語分割して名詞のみファイルに保存し,1日分蓄積したら集約,カウント,ストップワードの除去を行い,高頻出の単語を 30 抽出する.また Twitter からデータを取得し続け,tweet 中にニュースの高頻出の単語が含まれていたならば出力する.」である.処理記述中の registerFunction はユーザ定義関数(udf)の登録であり,それぞれ「テキストの単語分割と名詞抽出」,「ストップワード除去」,「テキスト中に含まれる名詞を追加」を行う.P1 のパイプラインはニュースデータ取得から保存,P2 から P4 のパイプラインは 24 時間毎に蓄積した名詞群に対する各種処理と抽出,P5 から P7 のパイプラインは Twitter データ取得から結果出力に対応する.このように,利用者は静的データとストリームデータに対する処理を混在して記述可能である.

3.3 処理の振り分け法

処理の振り分けについて述べる.処理のパイプラインを後述の 3 通りに分ける.

1)window を含むデータ取得演算で始まり,何らかのデータ出力演算で終了.
2)イベントに対する window を含むデータ取得演算で始まり,何らかのデータ出力演算で終了.
3)read を含むデータ取得演算で始まり,何らかのデータ出力演算で終了.

1)についてはストリーム処理として振り分け,JsSpinner で永続的に問合せを行う.2)についてはストリーム処理とバッチ処理の連携処理に振り分ける.JsSpinner にバッチ処理を起動するオペレータを用意し,イベントが発生したら Hadoop 処理を起動する. 3)についてはバッチ処理に振り分け Hadoop で処理する.ただし 1),2)それぞれのパイプラインとの join がある場合はそれに従う.

図 3 の例では,P1 のパイプラインは window で始まり write+で終わるのでストリーム処理に振り分ける.P2 から P4 は,P2 がイベントに対する window 演算で開始しているので,P2 をストリーム処理として振り分け,イベントが発生した場合,P3 と P4 の処理をバッチ処理として呼び出し実行する.P5 から P7 は window と read で始まるパイプラインで join しているためストリーム処理に振り分ける.

4. 結論

本研究では,ストリーム処理とバッチ処理を統合的に実行可能なフレームワークを提案した.提案フレームワークでは,Jaql に基づく単一の言語で両処理を記述可能で,既存のストリーム処理ツールとバッチ処理ツールを連携させた処理が実現される.

謝辞

本研究の一部は,科研費(#26280037)および文科省「実社会ビックデータ利活用のためのデータ統合・解析技術の研究開発」による.

参考文献

[1] Storm, distributed realtime computation. http://storm-project.net/
[2] S4: Distributed Stream Computing Platform. http://incubator.apache.org/s4/.
[3] Apache Hadoop. http://hadoop.apache.org.
[4] JavaScript Object Notation. http://json.org
[5] Kevin S.Beyer. Jaql: A Scripting Language for Large Scale Semistructured Data Analysis. IBM Research - Almaden San Jose, CA, USA.
[6]http://www-01.ibm.com/software/data/infosphere/hadoop/jaql/