5A-5

大規模表構造データに特化した分散処理システムの開発

中元政一†　羽室行信‡

†ERATO 湊離散構造処理系プロジェクト　‡関西学院大学大学院経営戦略研究科

1 はじめに

「big データ」のキーワードが世間を賑わせている今日、大規模データからの知識発見 (KDD) の重要性が増々高まっており、その基盤技術として大規模データの並列処理技術の開発が活発化している。その先鞭をつけたのが、Google で開発された Map-Reduce[1] であり、Key-Value 構造データを並列処理する汎用的システムである。そして、Map-Reduce のフリーウェア実装としての Hadoop[2] の登場で、多くのユーザが並列処理の恩恵を享受できるようになった。さらには、アクティブユーザが 10 億人を突破した Facebook が利用している Hive[3] は Hadoop 上に構築された SQL ライクな言語で表構造データを扱う仕組みとして注目されている。

一方で、これまでに我々は、表構造テキストデータ (CSV) を効率的に処理するためのソフトウェアライブラリ KGMOD[4] を開発してきた。また KGMOD を応用して作成した UNIX コマンド群 M コマンド (以下 MCMD と呼ぶ) を開発し、また、それらのソフトウェア基盤上に様々な応用ソフトウェアを構築してきた [5]。これらのソフトウェアを利用すれば市販の PC でも数億件 (数ギガバイト) レベルの表構造データの処理が可能となる。しかしながら、近年のデータ量の劇的な増加に伴い、テラバイト単位の処理も珍しくなくなっている。

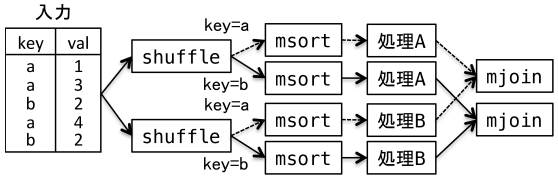
以上を背景に、我々はこれまでに MCMD を Hadoop と同様の原理を用いて並列化してきた。本稿では、その基本的アイデアを論じる共に、その効率性を検証することを目的とする。

2 提案方式

MCMD には、ある項目が整列していることを前提とした処理コマンドが多くある。いわゆるキーブレイク処理と呼ばれるもので、キー項目の値別に集計したり、複数の表を共通のキー項目で結合する処理である。キーブレイク処理を分散処理させることを考えると、同じキー値を持つデータブロックが一つの処理マシン上にあることが保証されるならばそのブロック単位で処理を完結することができる。これは、まさに Map-Reduce における shuffle 過程が行っていることである。

†Masakazu NAKAMOTO　‡Yukinobu HAMURO
†JST ERATO Minato Discrete Structure Manipulation System Project
‡Kwansei Gakuin University

具体例を図 1 に示す。入力データとして、key と val 項目を持つ CSV データが示されており、そのデータに対して処理 A と処理 B を実行して加工されたデータを key 項目で結合する処理例である。shuffle によって key 項目に対する Hash 値が計算され、その値が同じレコードは必ず同じストリームを流れることが保証される。



key 項目の値の Hash 値によってストリームを分割する。ここでは、key=a のレコードは必ず点線のストリームを流れ、key=b のレコードは必ず実線のストリームを流れることを例示している。

図 1: shuffle による MCMD の分散処理例

以上の基本的アイデアを中心として、次に示す特徴を持ったシステム (以下 GGP と呼ぶ) を構築した。1) 分散化されていないプログラムを自動的に分散化されたプログラムに変換する。2) 分散ファイルシステム (NDFS) を独自に導入することで処理効率を高める。3) 集約処理を sorting 前に実行するなどのいくつかの最適化を導入している。4) ノード間のストリームにネットワークパイプを利用している。

Hadoop や Hive に対する GGP の利点は、データ構造と処理内容を予め想定するために、最適化が比較的容易となり、また、MCMD の特徴であったテキスト処理の高速性をそのまま受け継ぐことができ、結果として高い効率性を期待できる。しかし一方で、表構造以外のデータ構造には対応できず、また SQL などの標準的なプログラミング言語を用いていないことが欠点となる。さらに、Hadoop では、ノード間でのストリームにおいて、結果を一時ファイルとして格納するが、GGP ではネットワークパイプを用いたストリーム処理のため、ファイル I/O の削減による高速化が実現できるものの、処理負荷によるノード間のロードバランスや障害に対するシステムの頑強性において Hadoop に劣る。

3 実験

提案方式 GGP の効率性を示すために、4 種の実行環境 (表 1) において 5 種の処理 (表 2) についてのベンチマークテストを実施した。その中で構成比計算につい

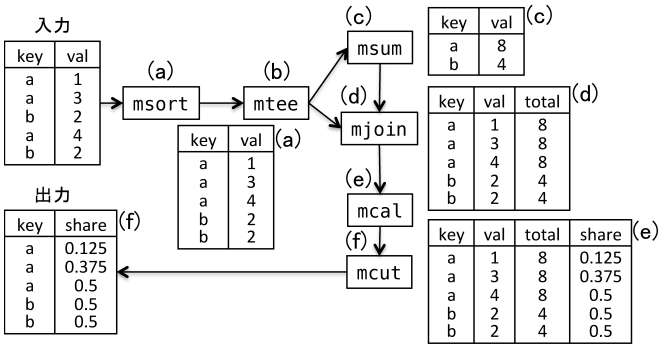
て、分散化されていないストリームの概要を図2に、その処理プログラムを表3上に例示している。また同等の処理を Hive で行う場合の SQL を同表下に示す。

表 1: 4 種の実験構成

分散なし	MCMD を単体のマシン上で実行する実験。
Hive	Hadoop 上で SQL を扱えるようにしたアプリケーション Hive を利用した実験。
Hadoop	Hadoop 上で MCMD をストリームとして実行する実験。
GGP	分散処理に GGP、ファイルシステムとして NDFS を用いた実験。本研究の提案システム。

表 2: 5 種の処理内容

cut	全ての項目を選択する。データのコピーに等しく、CPU 負はほとんどない。
sum	一つの項目を集計キーとして、数値項目を合計する。キー項目は事前に昇順に並べ変わっている。
sort	ある一様ランダムに並んだ項目をキーとして全行並べ替える。
share	構成比を計算する。ソーティングや結合など複数コマンドの組み合せ処理。
rule	2 アイテム相関ルールを計算する。18 個のコマンドが利用され、集計キーが 3 回変更される。



枠線で囲われた各コマンドとそのコマンドが出力するデータが表形式で示されており、また対応関係を記号 (a)〜(f) で示している。(b) はストリームを単純に分岐させる処理で出力データは (a) と同じである。

図 2: mcmd による構成比計算の流れ

実験結果は表4に示すの通りである。提案方式 (GGP) が cut から rule までいずれの処理においても最速の結果となっている。Hive と Hadoop の比較で見ると、sum,share,rule では Hive を若干上回っているものの、その他の処理においては逆に遅くなっている。両システムは分散処理とファイルシステムを共通としているので実行時間の差は、mapper と reducer の処理性能の差と考えられる。Hadoop では mapper と reducer において MCMD を利用しており、その効率性は高いものの、mapper の後で Hadoop が実行する sorting の処理効率に問題があると思われる。sort 処理の実行結果を見る限り、その点を Hive は改善することで効率性を達成していると推察される。次に Hadoop と GGP の比較であるが、分散処理方式を全て独自の実装としているため、ソーティングも独自のコマンドを利用している。そのため、sort 実験において飛躍的な効率化 (約 3 倍速) が

表 3: MCMD と Hive による構成比計算のプログラム

MCMD	(a)	msortf k=key f=val i=idat.csv |
	(b)	mtee o=npipe1 |
	(c)	msum k=key f=val:total o=npipe2
	(d)	mjoin k=key m=npipe2 f=total i=npipe1 |
	(e)	mcal c="${val}/${total}" a=share |
	(f)	mcut f=key,share o=odat.csv
Hive		SELECT key, val / sum FROM data JOIN (SELECT sum(data.val) AS sum FROM data GROUP BY key) sumData ON data.key=sumData.key;

MCMD における、"|" 記号は名前なしパイプによるストリームの接続を示し、ファイル名 npipe1,npipe2 は名前付きパイプによる接続を示す。2 列目に示された記号 (a)〜(f) は図 2 の記号に対応している。

表 4: 実験結果

	cut	sum	sort	share	rule
分散無し	179.3	1553.4	1,435.2	2,797.5	709.8
Hive	434.6	316.0	2,040.4	3,229.2	1,193.1
Hadoop	494.2	270.0	2,818.0	2,866.2	1,184.5
GGP	137.1	69.5	598.9	2,053.1	611.5

実験環境: Mac mini(cpu:Intel core-i5 2.3GHz, memory:16GB, OSX 10.7)。「分散無し」は 1 台のマシンによる mcmd による実行時間 (秒) を示し、その他は 5 台のマシンによる並列処理の結果を示す。入力データは rule 以外は 10GB で 4 億行、rule は 1GB で 4 千万行。

達成できている。しかしながら share や rule のように複雑な処理については、飛躍的な改善はない。これは、各ノードでの処理効率に比べてネットワーク負荷が増しているためである。これら複雑な処理の効率性を改善するために、さらなる最適化の導入が課題となる。

4 まとめ

表構造データに特化した並列処理システム GGP を開発した。その有効性は Hive との比較において 2〜4 倍程度の処理効率を達成している。今後は、さらなる最適化による処理効率の向上を目指すとともに、システムの公開に向けてのシステムの安定性を達成したい。

参考文献

[1] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. *In Proc. of the OSDI'04*, pages 137–150, 2004.

[2] Apache Hadoop. http://wiki.apache.org/hadoop.

[3] A. Thusoo and et al. Hive- a warehousing solution over a map-reduce framework. *In Proc. of VLDB'09*, 2009.

[4] 森田裕之, 羽室行信, 加藤直樹, 中元政一, and 松田康之. 大容量データに対する高速データマイニングシステム-kgmod-. **経営情報学会** *2008* **年度秋季全国大会**, 2008.

[5] 羽室行信, 中元政一, and 森田裕之. 離散構造データのデータマイニングツール開発とその応用. **人工知能学会誌**, 27(3):261–270, 2012.