分散ファイルシステムを用いた大規模データ処理システムの構築

ERATO湊離散構造処理系プロジェクト
2012年度 秋のワークショップ

2012年10月15日(月)
ゆうばりホテルシューパロ

中元政一 JST ERATO 湊離散構造処理系プロジェクト
羽室行信 関西学院大学経営戦略研究科

目的

大規模表構造データの高速分散処理システムGGPの開発。
（表構造データに特化したHadoopの開発）

今回の目的

ファイルシステム（NDFS)の開発と効果の検証
（表構造に特化したHDFS）

関連技術

	Google	Yahoo	本研究
並列処理	Map-Reduce	Hadoop	GGP
ファイルシステム	GFS	HDFS	NDFS
言語	Sawzall	HIVE	KGMOD

前回までの結果

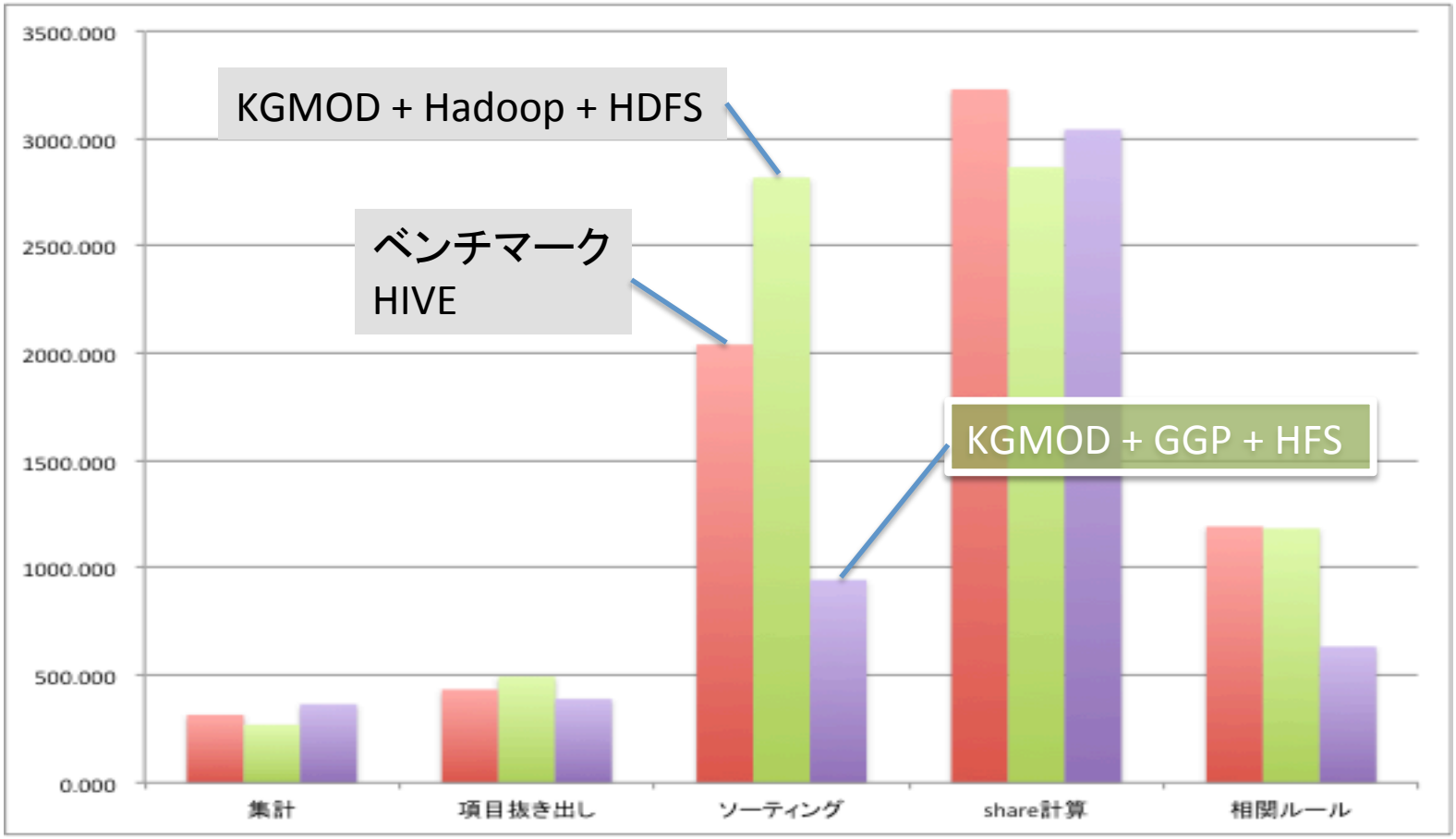
これまでに我々が開発を進めてきた、表構造データの高速処理システム「KGMOD」を分散処理させる基礎技術GGPを開発した。

1. KGMOD on GGP + HFS
   前回までに開発した仕組み
   (ファイルシステムはMacの一般的なもので分散されていない)

2. Hive on Hadoop + HDFS
   大規模表構造データをHadoop上で実現したシステム
   このシステムが対象ベンチマークとなる。

3. KGMOD on Hadoop + HDFS
   処理はKGMODを使い、分散処理はHadoopを利用
   1と比べてHadoopの性能を評価するための実験

前回までの結果



ソーティングと相関ルール以外はHIVEと同等程度
　←通常のファイルシステムを使っていることの限界(Shuffleとmergeが頻発する)
Hadoop+HDFS上での実行では逆に遅くなる。

→ 分散ファイルシステムを独自に実装

GGPサーバ構成

クライアント

スクリプトの
実行要求

命令に応じた結果
エラーメッセージ

NDFS

GGPマスタサーバ

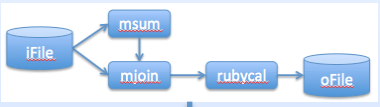
資源管理

GGP処理サーバ別情報

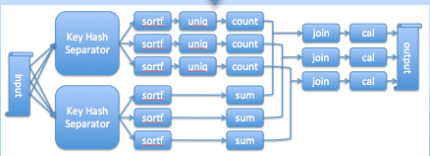
Static情報
1. CPU数、CPU処理性能
2. メモリ量、メモリ性能
3. HDD量、HDD性能
4. ネットワーク性能

Dynamic情報
1. 実行中スクリプト
2. マシンStatus
3. 受信データ量
4. 送信データ量

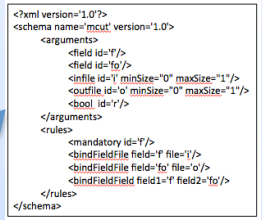
並列化スクリプト変換



1. 処理フロー並列化
2. 入出力ファイル名解決
3. ノード解決(GGP処理サーバ割り当て)
4. エッジ解決(メモリFIFO, NET FIFO)



コマンドスキーマ(XML)



```
<?xml version='1.0'?>
<schema name='mcut' version='1.0'>
    <arguments>
        <field id='f'/>
        <field id='fo'/>
        <infile id='i' minSize="0" maxSize="1"/>
        <outfile id='o' minSize="0" maxSize="1"/>
        <bool id='r'/>
    </arguments>
    <rules>
        <mandatory id='f'/>
        <bindFieldFile field='f' file='i'/>
        <bindFieldFile field='fo' file='o'/>
        <bindFieldField field1='f' field2='fo'/>
    </rules>
</schema>
```

実行ステータス管理

処理ID, 件数, Status
0001, 2000, 処理中
0002, 0, 起動中
0003, 100, エラー
:

サーバ別
ハートビートメッセージ
・Dynamic資源情報

実行要求

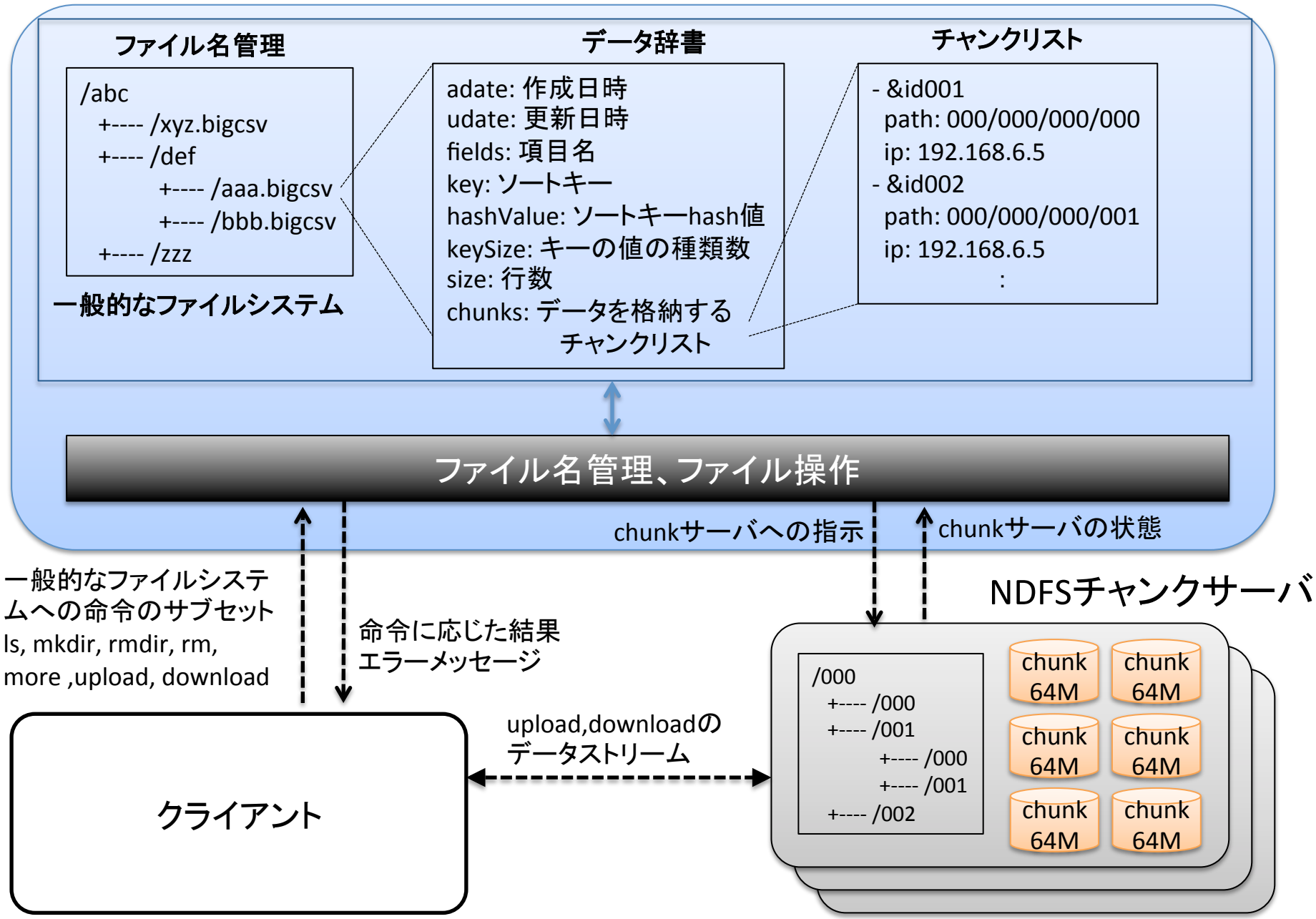
コマンド別
ハートビートメッセージ
・実行ステータス
・処理済件数 など

GGPサーバ

コマンド

コマンド

NDFSサーバ構成


NDFSマスタサーバ
ファイル名管理
/abc
+---- /xyz.bigcsv
+---- /def
+---- /aaa.bigcsv
+---- /bbb.bigcsv
+---- /zzz
一般的なファイルシステム
データ辞書
adate: 作成日時
udate: 更新日時
fields: 項目名
key: ソートキー
hashValue: ソートキーhash値
keySize: キーの値の種類数
size: 行数
chunks: データを格納する
チャンクリスト
チャンクリスト
- &id001
path: 000/000/000/000
ip: 192.168.6.5
- &id002
path: 000/000/000/001
ip: 192.168.6.5
:
ファイル名管理、ファイル操作
chunkサーバへの指示
chunkサーバの状態
一般的なファイルシステムへの命令のサブセット
ls, mkdir, rmdir, rm,
more ,upload, download
命令に応じた結果
エラーメッセージ
NDFSチャンクサーバ
/000
+---- /000
+---- /001
+---- /000
+---- /001
+---- /002
chunk 64M
chunk 64M
chunk 64M
chunk 64M
chunk 64M
chunk 64M
クライアント
upload,downloadの
データストリーム

NDFSの特徴

- マスタースレーブ型アーキテクチャ
- 64Mバイト単位でのデータの分散化
- データ書き込み時の排他制御。
- CSVデータのデータ辞書を格納（項目名、項目情報など）
- 処理ノードにデータがない場合は、ネットワーク越しのストリーム接続
- ファイル冗長化
- データのストリームは、マスタサーバを経由せず、チャンクノード間で行う。

実験環境

- マシンスペック
  - mac mini 1台
  - OS X 10.5
  - CPU: Intel core2 duo 2.26GHz
  - メモリ: 4Gバイト

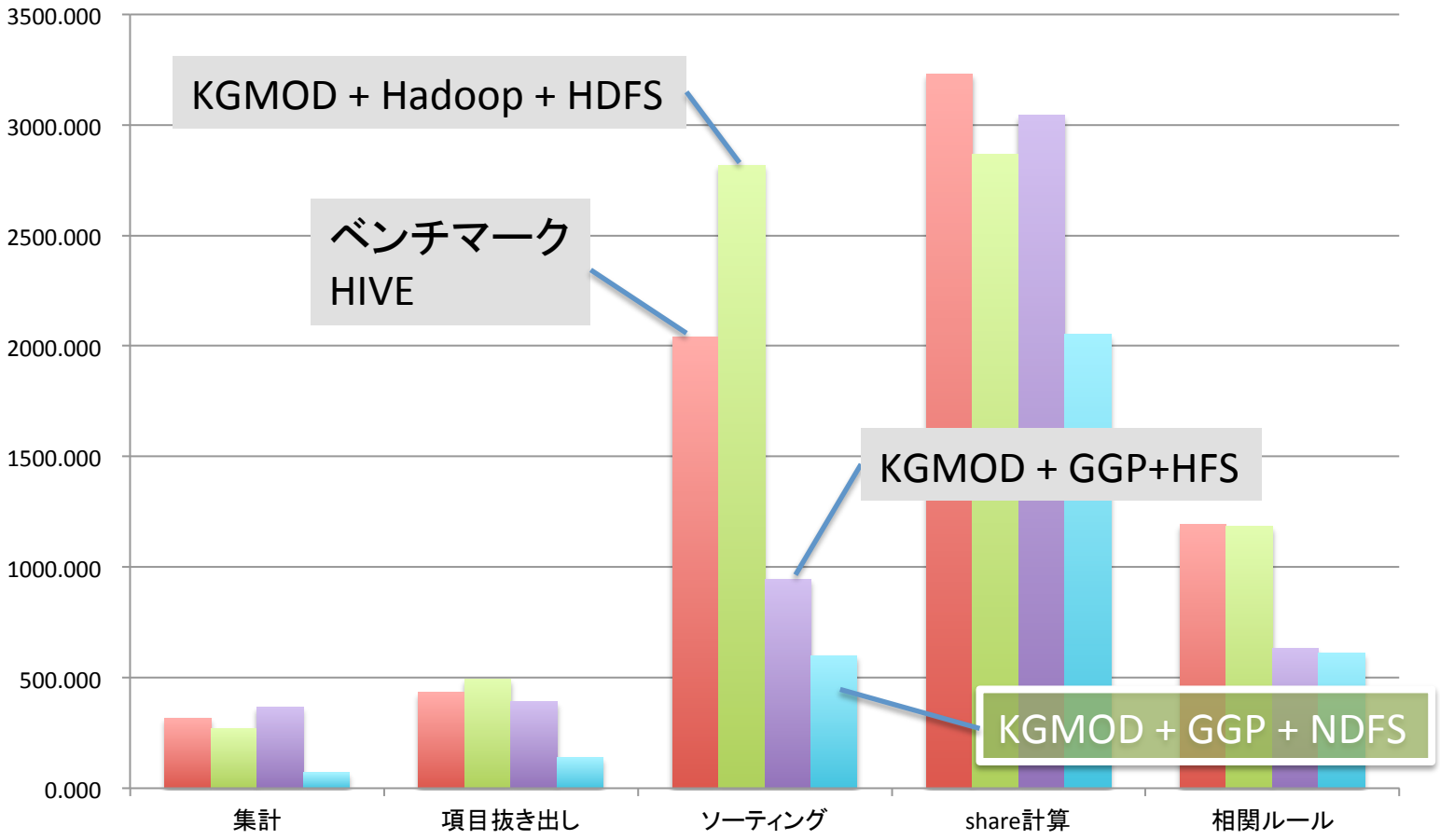
  - mac mini 5台
  - OS X 10.7
  - CPU: Intel core-i5 2.3GHz
  - メモリ: 16Gバイト

データスペック

	実サイズ	key数	行数(億行)
1G	1,136,517,084	1000	0.4
10G	11,365,160,701	1000	4
100G	113,651,606,894	1000	40
500G	568,258,034,470	1000	200
1T	1,136,258,153,680	1000	400

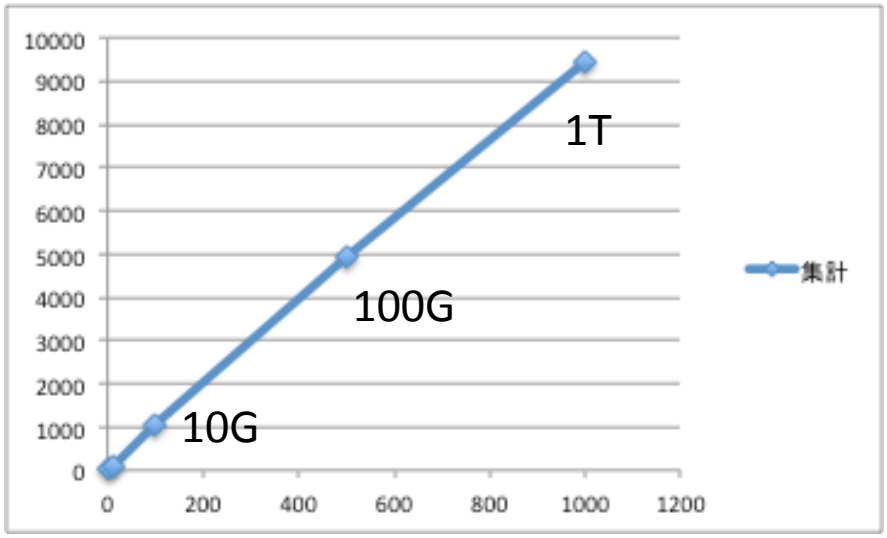
相関ルール計算用データ
データサイズ：約1Gバイト
行数： 約1億
key数： 約15百万
アイテム数： 270

実験結果

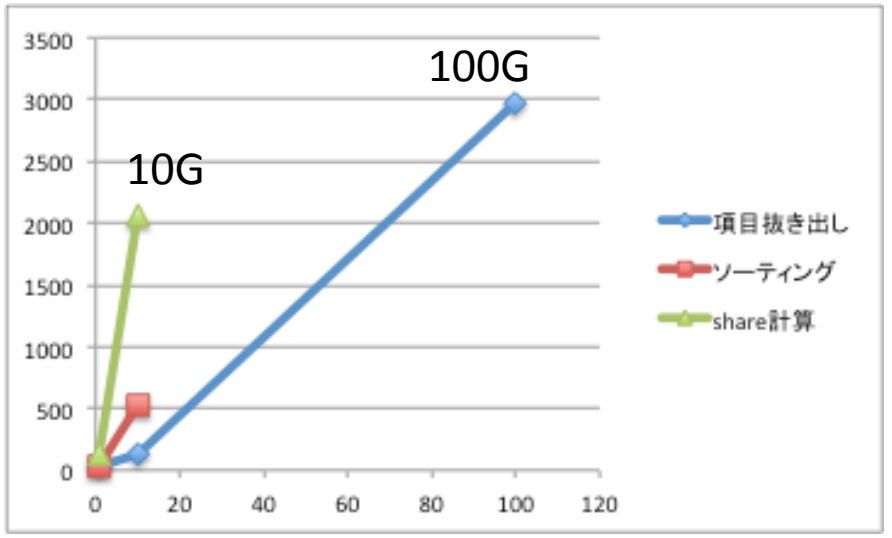


前回の実験に比べて、いずれの処理についても改善が見られた。

データ量と処理効率の関係



集計処理を1G～1Tまでスケールした場合の処理速度。



その他の処理を1G～100Gまでスケールした場合の処理速度。

なぜHIVEより速いのか？

- Hadoopは途中経過をファイル出力し、GGPはネットワークパイプで接続する
  - Hadoopは失敗したときは容易にresumeできる。
  - GGPではネットワークパイプにおいて、通過したデータサイズを常に監視するなど、より複雑な技術が必要。
- GGPは試験実装のため、厳格なエラーハンドリングや詳細な監視モニタ(ハートビートメッセージ等)の実装はしていない。それらのオーバーヘッドがどの程度の負荷になるかは不明。

今後の開発予定

- ネットワークパイプをプロセスで起動しているので、大規模並列(例えば、1000台並列)では最大プロセス数制限に抵触する。
  - →thread化 or サーバー化
  - GGPにおけるボトルネックは、ワークファイルが書きだされることによるディスクI/Oの頻発
    - 処理負荷のアンバランスによるFIFOキューのファイル書き出し
    - ソーティングにおける分割ソートに伴うファイル書き出し
- スクリプトの並列化変換の最適化。
- タスク管理モニタの高度化
- エラーハンドリング
- ストレステスト
- Linux環境への移植
- Amazon Web Service等でのペタ級データでの超大規模実験