UNISYS

PostgreSQLによるHadoopを利用した大規模データ分析機能 PG Inter-Analyticsの紹介

中山　陽太郎
yotaro.nakayama @ unisys.co.jp
日本ユニシス株式会社
総合技術研究所

All Rights Reserved, Copyright © 2013 Nihon Unisys, Ltd.

アジェンダ

UNISYS

- No-NoSQLによるアプローチ
  - Hadoop
  - ビッグデータ活用の課題
- PostgreSQL Inter-Analyticsとは
  - なぜPostgreSQL + Hadoopか
- Webログ分析の例
- まとめ
  - 課題と次のステップ

All Rights Reserved, Copyright © 2013 Nihon Unisys, Ltd.

No-NoSQLによるアプローチ

UNISYS

- No-NoSQLとは、Not only NoSQL
- Hadoop, NoSQLだけによらないBig Dataへのアプローチ
- No-NoSQLのいくつかの動向
  - ✓ HadoopとRDB・DWH連携
    - HadoopからDWHへのデータロード最適化
    - Hadoop対応ETL
  - ✓ SQL-Hadoop融合
    - SQLとMapReduceを融合 (Hadaptなど)
    - 外部表による連携 (CitusDBなど)
  - ✓ SQL in Hadoopの展開
    - Hiveの高速化Stinger (Howtonworks)
    - Shark (AMPLab)
- PG Inter-Analyticsも、Hadoop, NoSQLだけではないNo-NoSQLによるアプローチの一つ

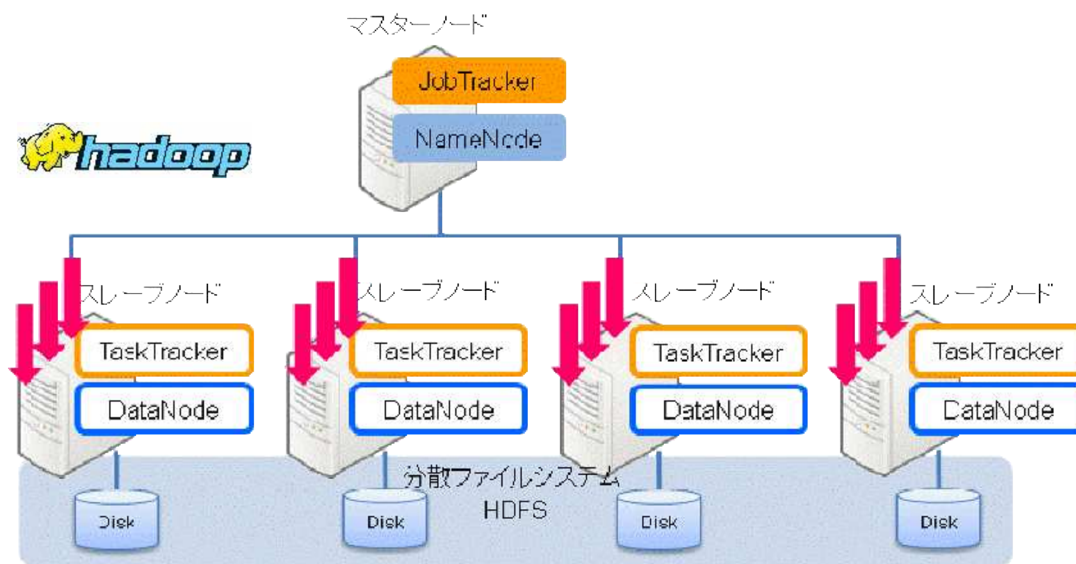
All Rights Reserved, Copyright © 2013 Nihon Unisys, Ltd.

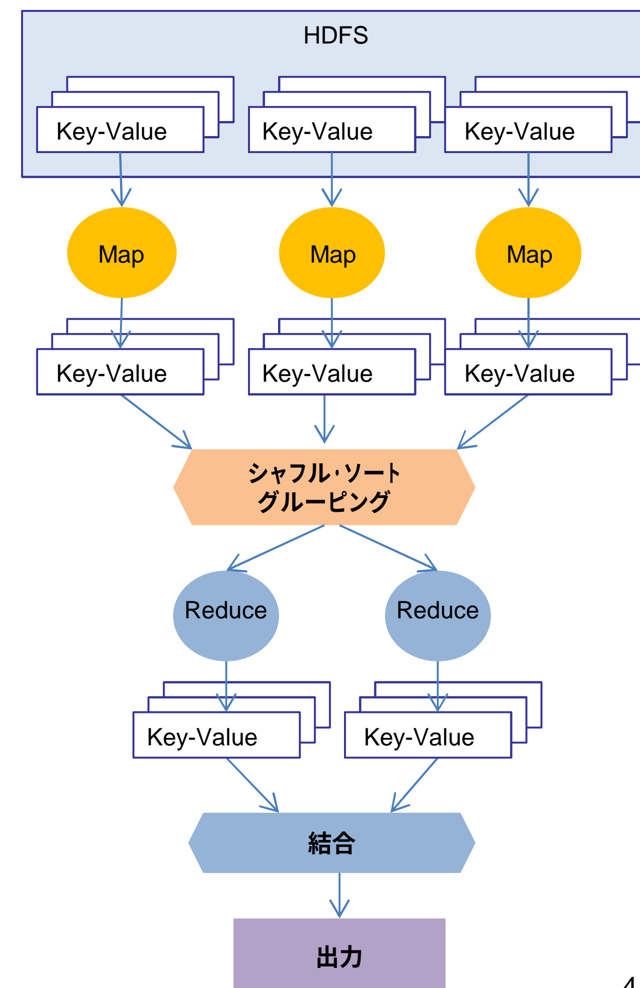
Hadoop

UNISYS

- Googleによる Google File System(2003), MapReduce(2004)の論文をベースに、D.Cutting氏らにより、検索エンジンのために開発される
  - Yahoo!, Facebook等の超大規模データ処理基盤
- 2つのコア機能
  - 並列処理フレームワーク：MapReduce
  - Hadoop分散ファイルシステム：HDFS
  - MapReduceのKey, Valueによる入出力

Map: input ➔ (key, value)
Reduce: (key, [values]) ➔ output





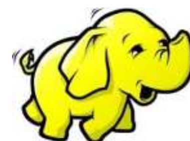
All Rights Reserved, Copyright © 2013 Nihon Unisys, Ltd.

ビッグデータ活用の課題

UNISYS

- RDBMSの課題
  - 大規模データに対するスケーラビリティ
    - 一台のサーバで処理可能なデータベース容量の限界
    - スケールアップによる対応はコストが増大
  - 大量データ転送･大量データロードの負荷増大
    - データソースの転送、成形、データロード処理
  - 非構造化データの対応のハードル
    - 正規化の負担、半構造データの扱い
    - 大規模テキストデータ
- Hadoopの課題
  - データ処理のためのMapReduceプログラミングが必要
  - Key-Valueデータ処理を意識した設計と実装
  - 少量データ分析におけるオーバーヘッドの増加によるパフォーマンス低下
  - 構造化データにおける集合演算、結合による複雑な問い合わせは難しい

PostgreSQL



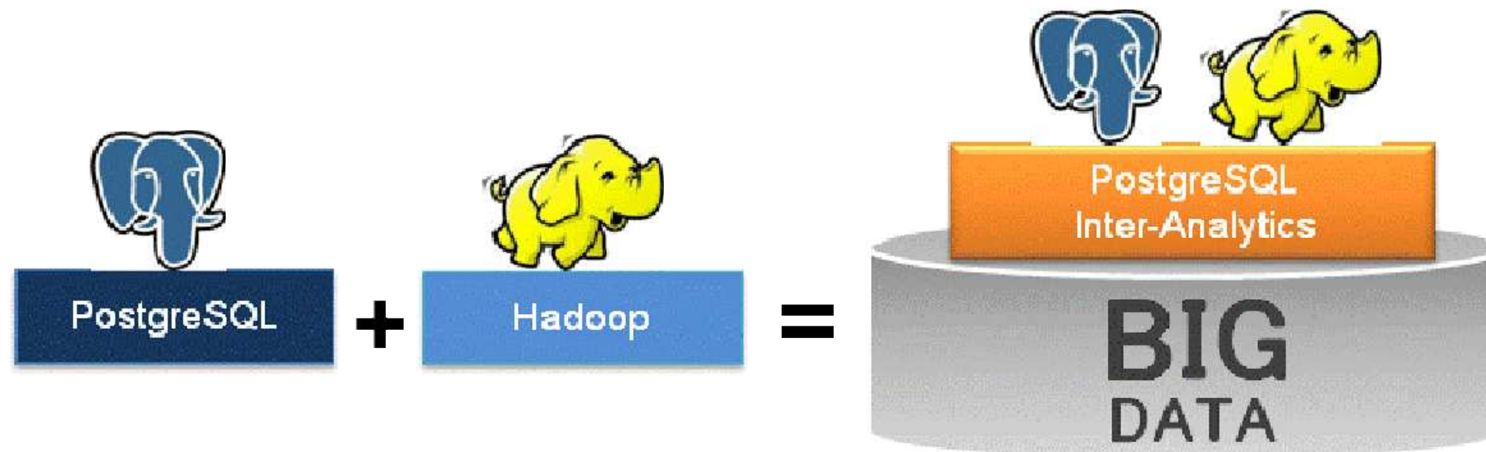
Hadoop

All Rights Reserved, Copyright © 2013 Nihon Unisys, Ltd.

PG Inter-Analyticsとは

UNISYS

- PostgreSQLとHadoop双方の利点を活かし、ビッグデータに対応
- PostgreSQL・Hadoop相互連携
  - 分析対象のデータをHadoop上に格納
  - PostgreSQLからHadoop上のデータを処理
- 巨大な分析・処理対象データは、Hadoop上で管理
  - 大規模データに対するスケーラビリティ
- データ移動・データロードの負荷を排除
  - プラットフォーム切り替えの煩雑さ、データ移動のコストを排除
- 非構造化データ、テキストデータに対応
  - Hadoopにより非構造化データ処理が効率的に行える

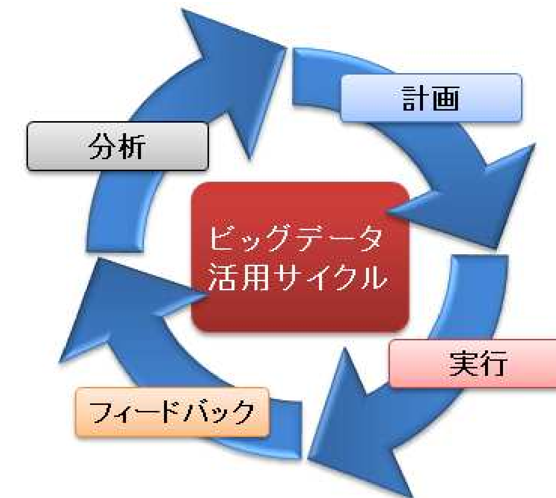


All Rights Reserved, Copyright © 2013 Nihon Unisys, Ltd.

PG Inter-Analyticsとは (続き)

UNISYS

- 一元的なデータアクセス
  - PostgreSQLからSQLによりHadoop上のデータ処理を実施
- In-Situ (その場) によるデータ処理
  - Hadoop上のデータ処理はHadoop上で実行
  - 全データを対象
  - データ移動の排除



- インターラクティブなデータ操作
  - 分析は、実行結果をフィードバックした繰り返しが重要
    実行結果のフィードバックを繰り返し反映することで、
    効率的な分析サイクルを実現
- マルチインターラクティブなデータストア
  - Hadoop側からも独立してデータを利用可能
  - データソース、結果はHadoopに保持することで、バッチ処理や
    定型処理をHadoop上で直接実行することも可能

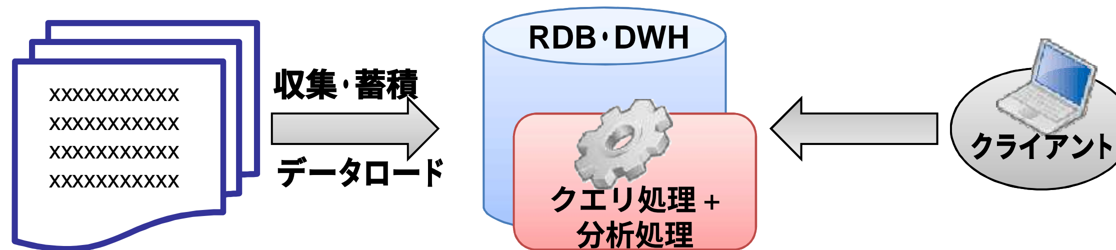
All Rights Reserved, Copyright © 2013 Nihon Unisys, Ltd.

RDBとHadoopの相互連携によるデータ分析

UNISYS

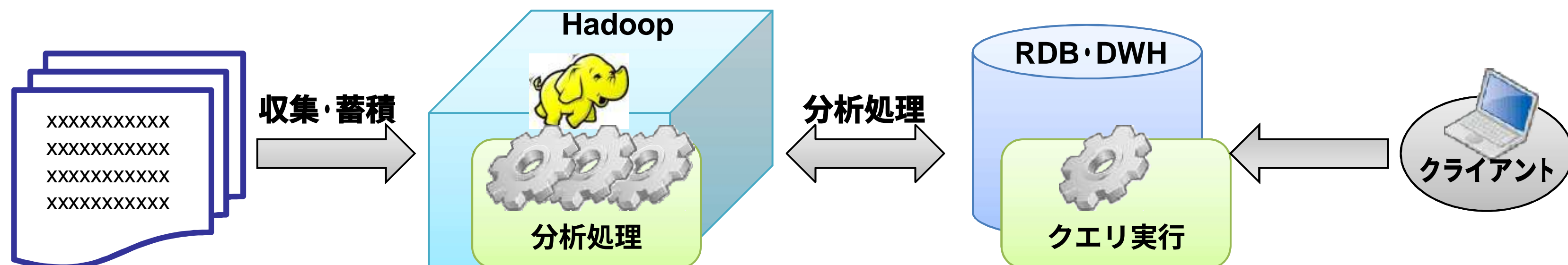
データベース内分析によるログ分析処理

- 収集蓄積サーバからデータ転送
- クレンジング･データロード
- データソース保持の限界
- DBサーバの処理リソースの枯渇



相互データベース内分析によるログ分析処理

- Hadoop上でのデータソース蓄積
- Hadoopクラスタの処理リソース
- 並列化処理による効率化

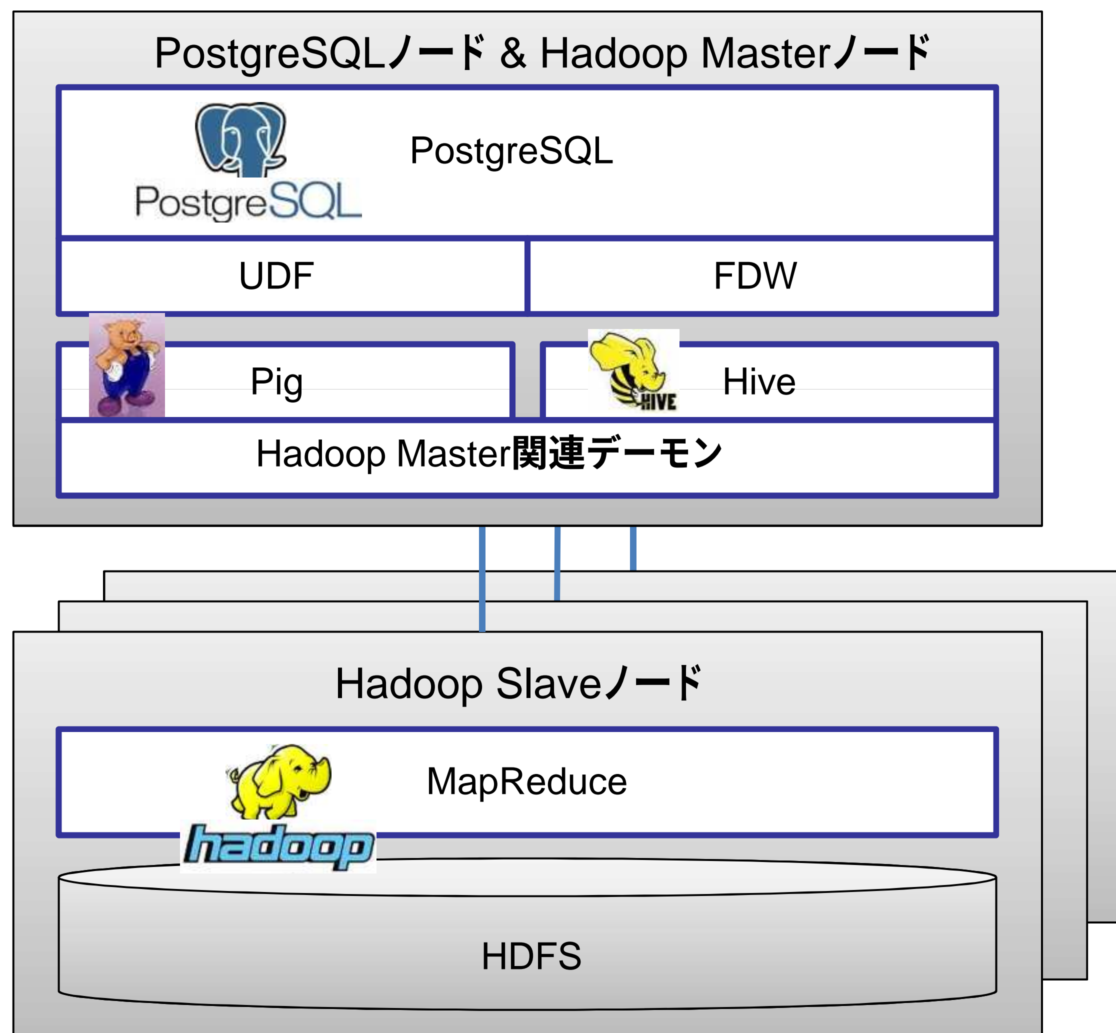


All Rights Reserved, Copyright © 2013 Nihon Unisys, Ltd.

全体像

UNISYS

- PostgreSQLストアド関数(利用者定義関数)からHadoopを利用

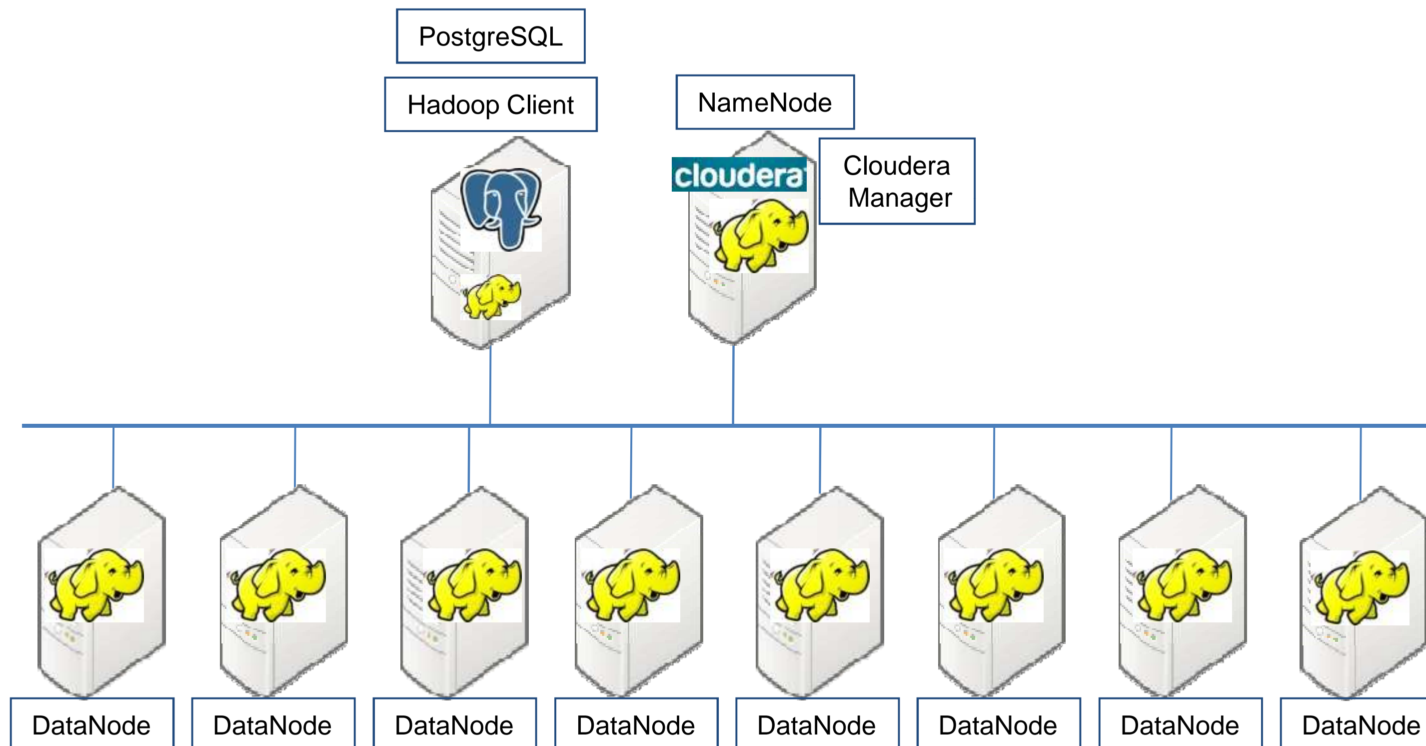


All Rights Reserved, Copyright © 2013 Nihon Unisys, Ltd.

システム構成概要

UNISYS

- PostgreSQL 9.3
- Hadoop - Cloudera CDH4.3
- PostgreSQLをHadoopマスターノードに導入
  - ✓ Hadoop HDFSへのアクセス権限
- PGストアド言語として、PL/Python使用

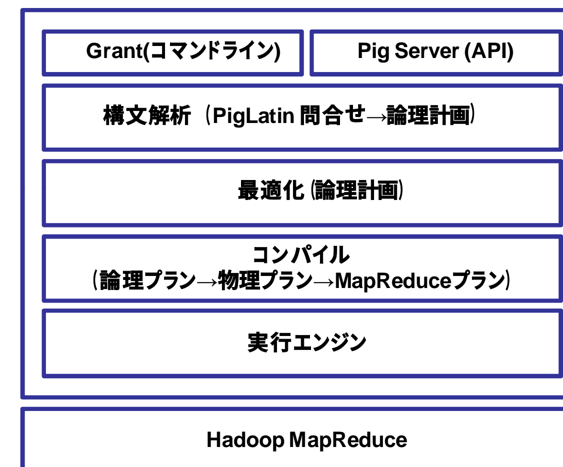


All Rights Reserved, Copyright © 2013 Nihon Unisys, Ltd.

Apache Pig

UNISYS

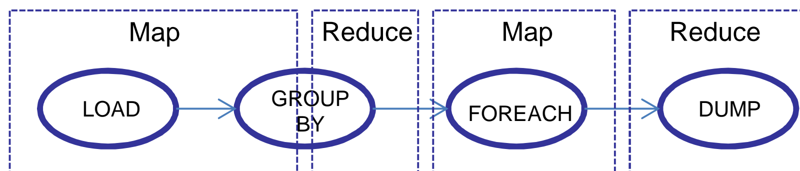
- Yahoo!により開発、Hadoopファミリプロジェクト。
  10月に最新版0.12がリリース
- 大規模な半構造化データセットに対して問合せを実行するための手続き型高級言語
- HDFS上のファイルを入力テーブルとして、Hadoop MapReduceフレームワークで動作 (ローカルでも実行可能)
- SQLとほぼ同等のPig Latin (ピッグラテン) 独自言語
- 関係演算、算術演算で対応できない処理は、UDF (User Defined Function) で対応
- 実行方法:インタプリタ、バッチ、埋め込み言語 (Python, Java Script) に対応



処理例

```
grunt> A = LOAD 'data' USING PigStorage() AS (f1:int, f2:int, f3:int);
grunt> B = GROUP A BY f1;
grunt> C = FOREACH B GENERATE COUNT ($0);
grunt> DUMP C;
```

MapReduce実行イメージ



All Rights Reserved, Copyright © 2013 Nihon Unisys, Ltd.

ストアド関数によるPig実行

UNISYS

- ストアド言語 PL/Pythonを使用
  - Pythonの開発生産性、応用性の高さ
  - Pythonが埋め込みPig対応言語 (今回利用せず)
- PythonからPigの実行
  - PythonからPigスクリプトを実行
  - Pigによる手続き的なデータ処理が可能
    - SQLでは苦手なデータ処理にも対応
- Hadoop上の入力データソースの参照
  - データソースの項目をフィールドの順番で参照
    - プロトタイプにて採用
  - データソースの項目を定義した名前で参照
    - PostgreSQLの外部表により、Pigの入力、出力データを定義
    - CREATE FOREIGN TABLEで定義
    - Hadoop上にも同じテーブルを定義する関数を提供

All Rights Reserved, Copyright © 2013 Nihon Unisys, Ltd.

ストアド関数によるPig実行 (2)

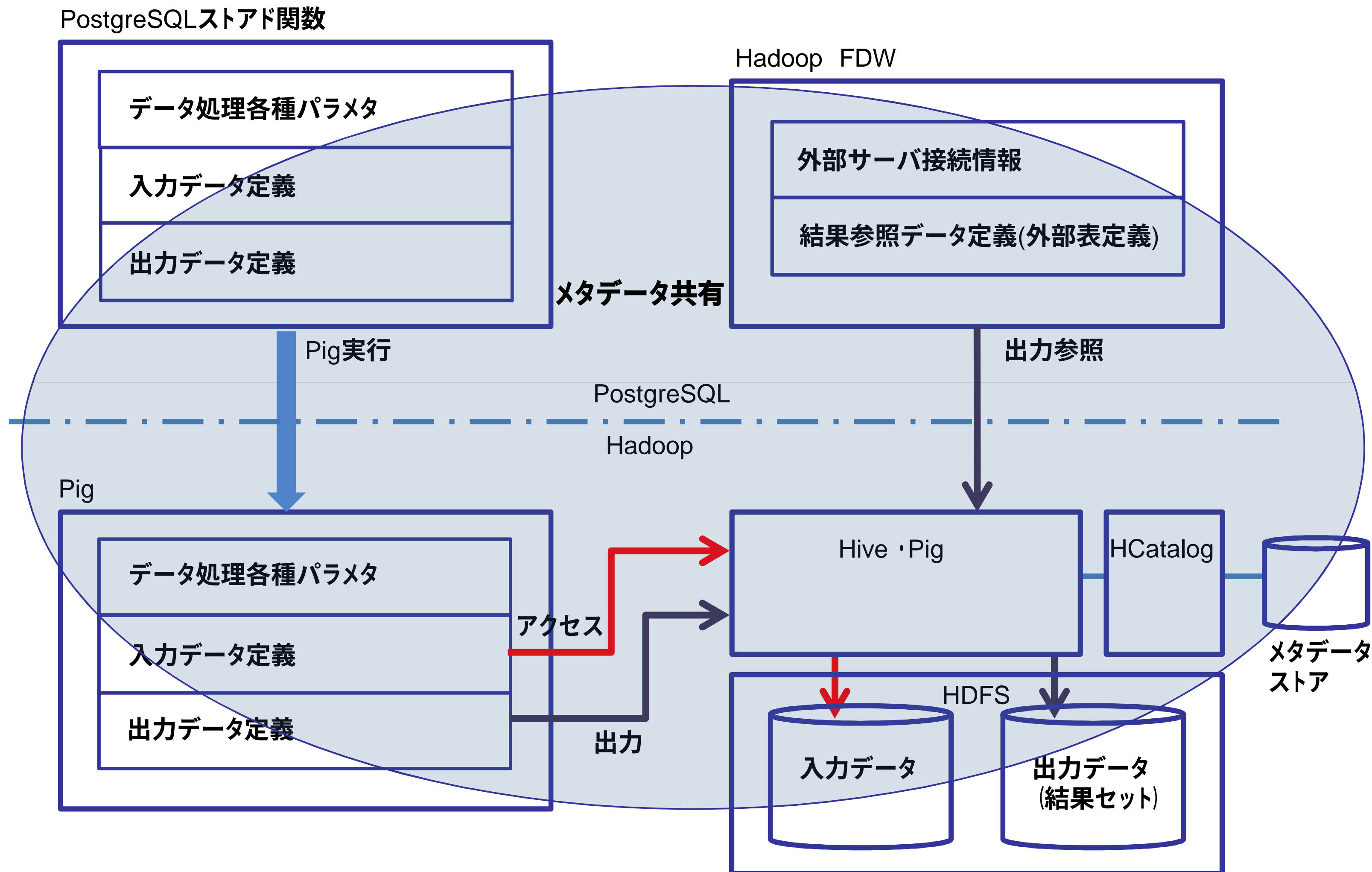
UNISYS

- 実行ステータスと結果セット
  - 呼び出し側に返すステータス
    - Pigスクリプトの実行ステータス (成功・失敗)
      - 状況に応じたステータスは未対応
        Hadoop側のステータス(ログファイル確認など)
  - 結果セットの出力
    - 結果セットは、Hadoop HDFSに出力
    - 事前定義済みのテーブルに出力
  - 結果セットの参照
    - PostgreSQLからの参照のため、Hadoop上の外部表を使用
      - OpenSCGのBigSQLで提供されるHadoop FDW
        http://www.bigsql.org/se/hadoopfdw/
      - 結果セットは、PostgreSQLで事前に定義
    - PostgreSQL、Hadoop Pig, Hiveでスキーマ定義を共有
      - PostgreSQLで定義した外部テーブル定義から、Pig, Hiveから参照可能なテーブルを自動で定義する関数 pg2hive_create_table
      - (注) HiveのテーブルとR義からPostgreSQLの外部テーブルを生成する
        hadoop_create_tableは、BigSQLで提供されている。

All Rights Reserved, Copyright © 2013 Nihon Unisys, Ltd.

データ定義･データ参照の関係

UNISYS

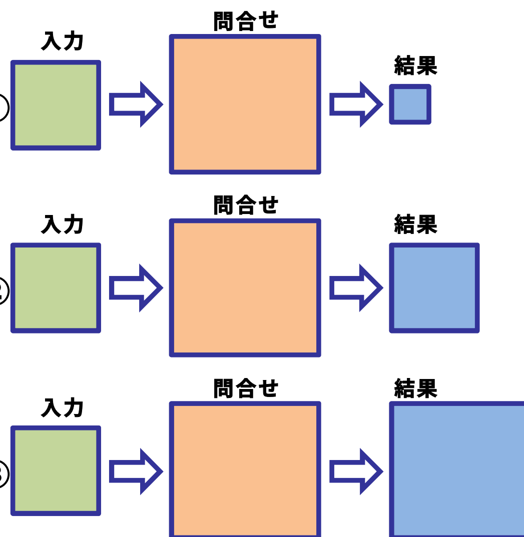


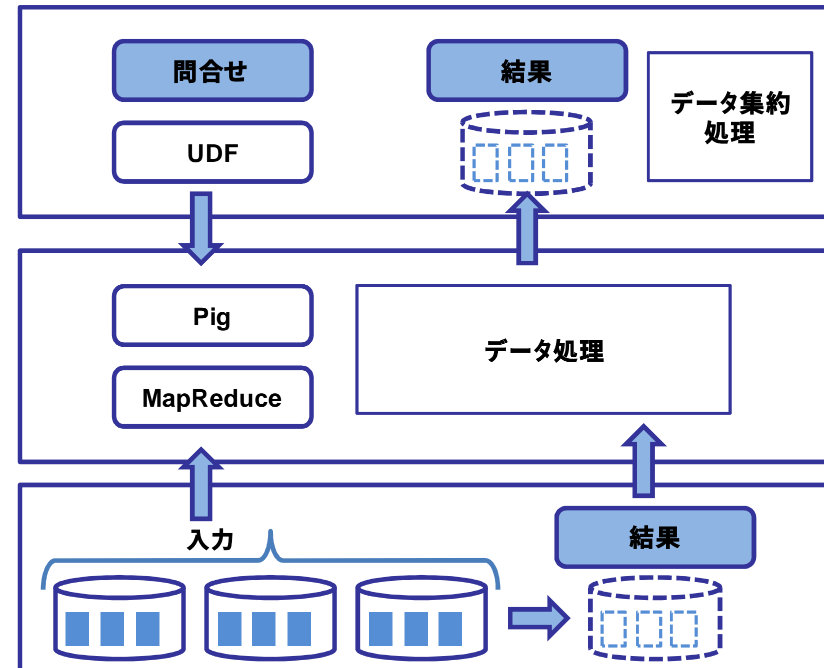
All Rights Reserved, Copyright © 2013 Nihon Unisys, Ltd.

問合せ結果の考慮

UNISYS

- 結果データをどうするべきか？
  - データの移動をできる限り排除したい
  - 結果セットの全体サイズが非常に大きい場合、データ転送に時間がかかる
  - 結果セットを転送後、PostgreSQLのテーブルに格納することも可能
    ⇒ しかし、結果が多量の場合、処理時間が増大
- 対応
  - 実行結果は、結果セットとしてHadoop上に出力
  - PostgreSQLから外部表として参照



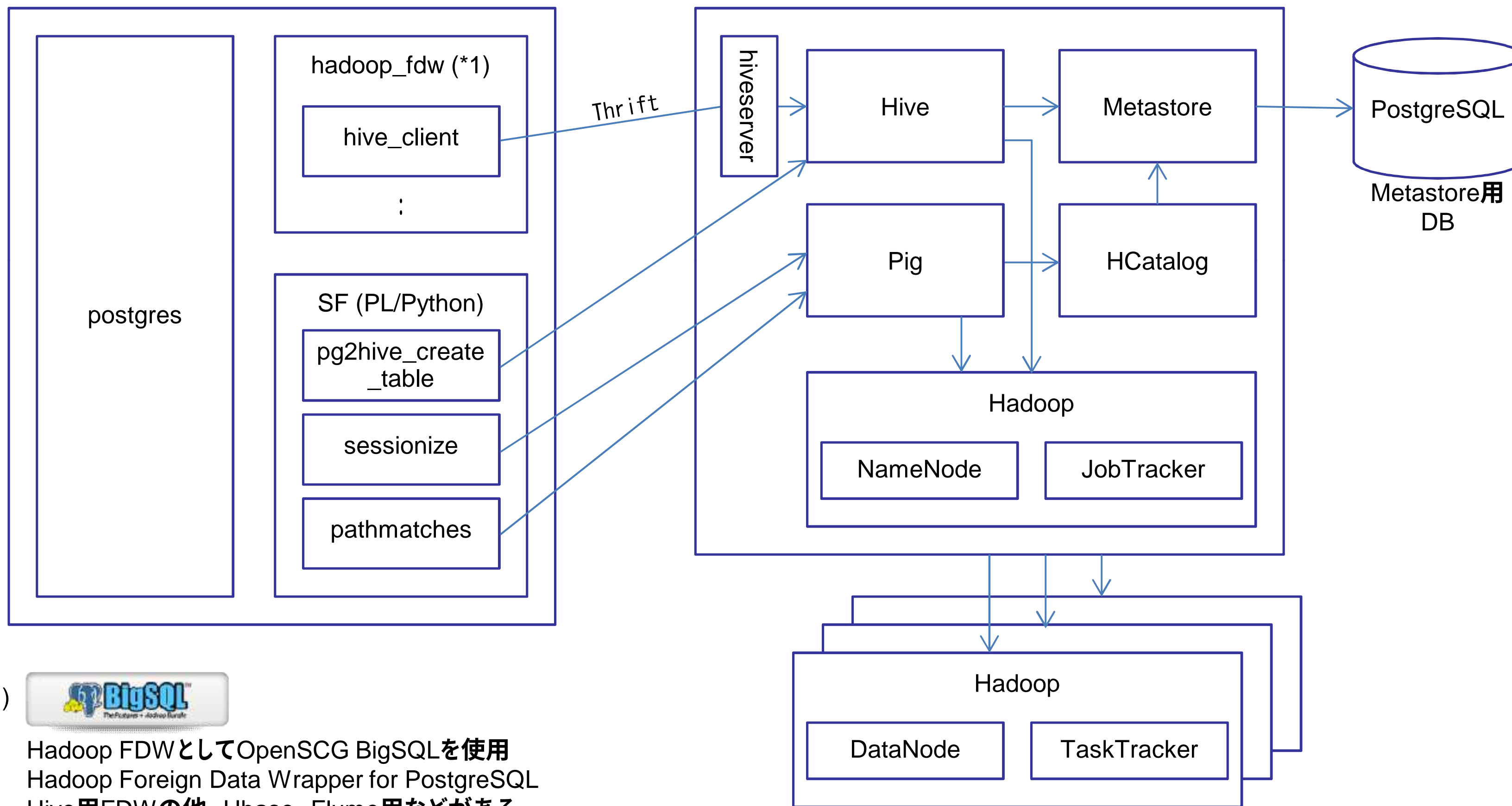


All Rights Reserved, Copyright © 2013 Nihon Unisys, Ltd.

結果セットの参照

UNISYS

- 結果セットの参照のため、BigSQLのHadoop FDWを使用
- HCatalogにより、PigとHiveでデータの共有が可能
  - HiveとPigとでスキーマ定義を共有し、データを共有



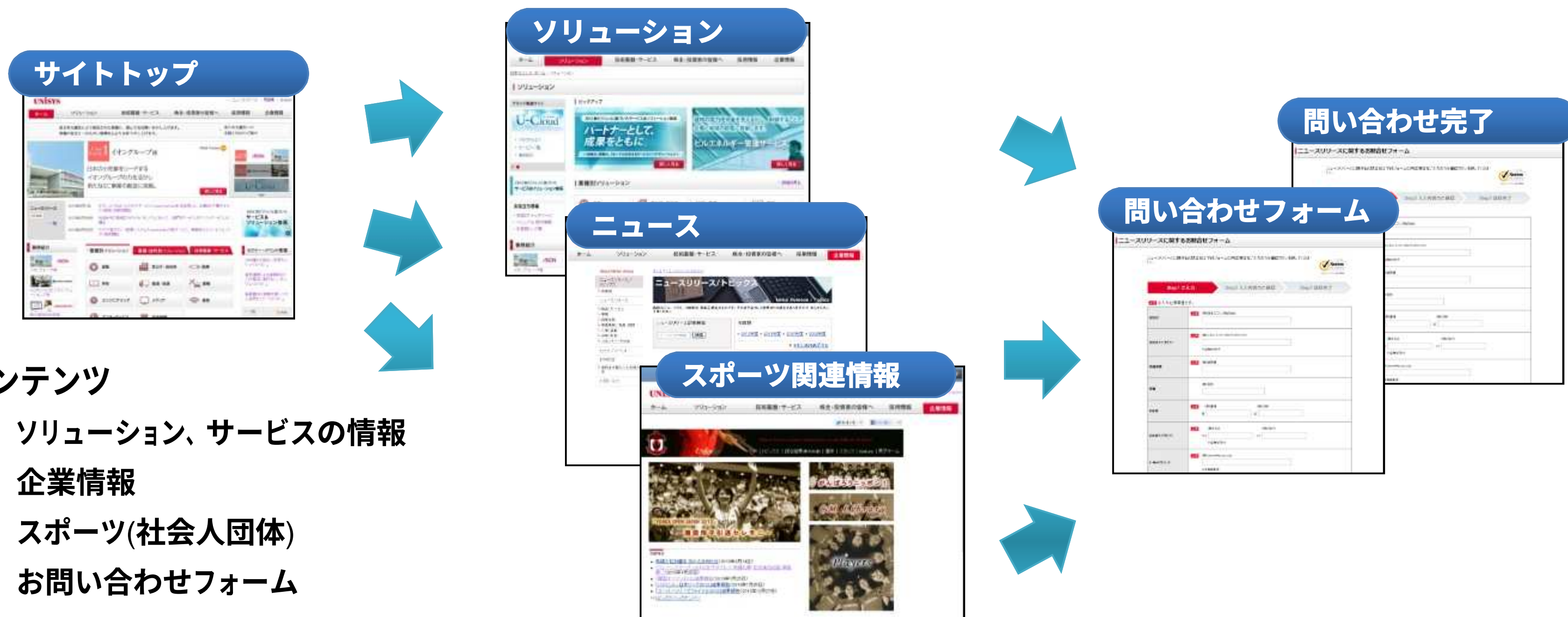
Hadoop FDWとしてOpenSCG BigSQLを使用
Hadoop Foreign Data Wrapper for PostgreSQL
Hive用FDWの他、Hbase、Flume用などがある

All Rights Reserved, Copyright © 2013 Nihon Unisys, Ltd.

Webサイトのアクセス経路の解析例

UNISYS

- Webサイト内をどのような経路でアクセスして「問い合わせフォーム」に到達したかを、Webログから解析
  - Webサイトのアクセシビリティ改善
  - 製品・サービスの注目度の把握
- 解析の第一歩として、どのページからの問合せが多いかを解析
  - 問合せに至る直前のページを集計

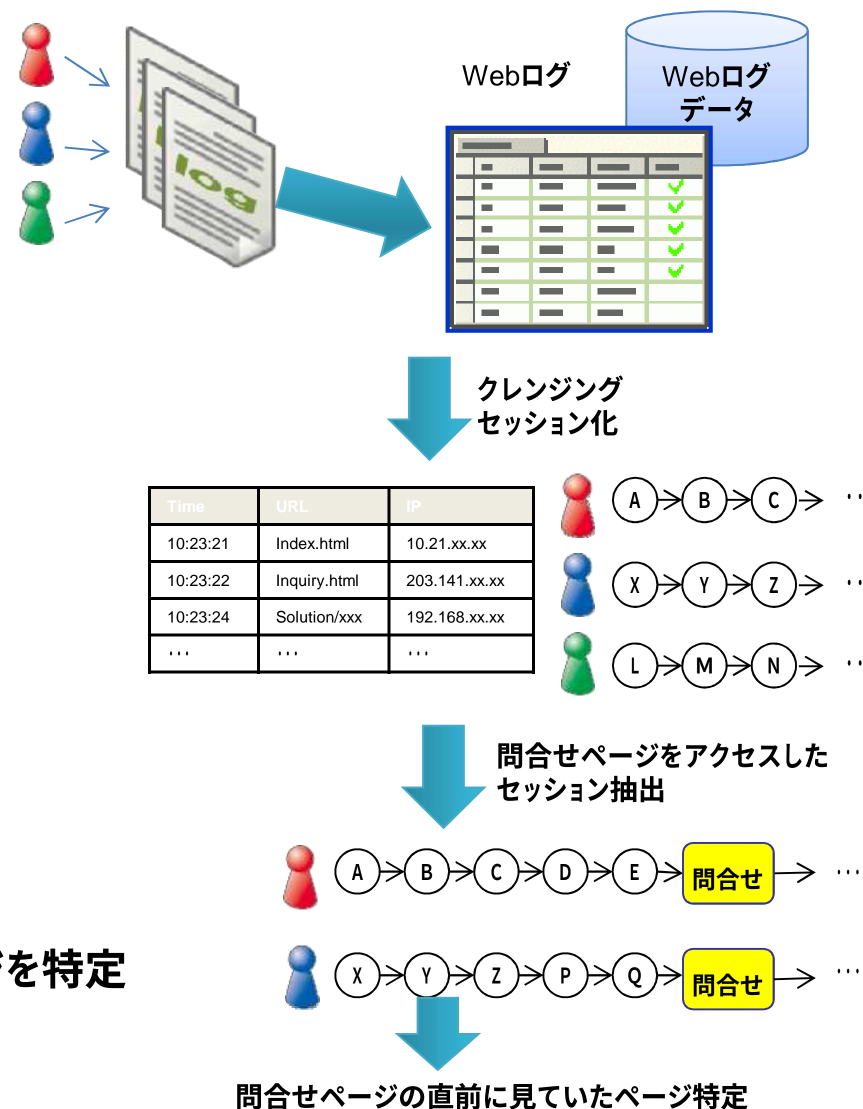


- コンテンツ
  - ソリューション、サービスの情報
  - 企業情報
  - スポーツ(社会人団体)
  - お問い合わせフォーム
    お客様からのお問い合わせ
  - サイト内の全ページからリンク

All Rights Reserved, Copyright © 2013 Nihon Unisys, Ltd.

Webログ分析処理の流れ

- ログデータの格納
  - ログファイルをHDFSへ配置
- クレンジング
  分析に不要なデータの除去
  - 拡張子:.css/.js/.gif/.png/.jpg
  - Robot.txtへのアクセス、etc
- セッション化
  - ストアド関数「Sessionize」を使用
  - 同一IPアドレスからのアクセス
  - アクセス間隔が10分以内(設定は任意)
    ⇒ 同一セッションと判断
- 動線分析
  - 問い合わせページの直前に見ていたページを特定
  - ストアド関数「Pathmatches」を使用

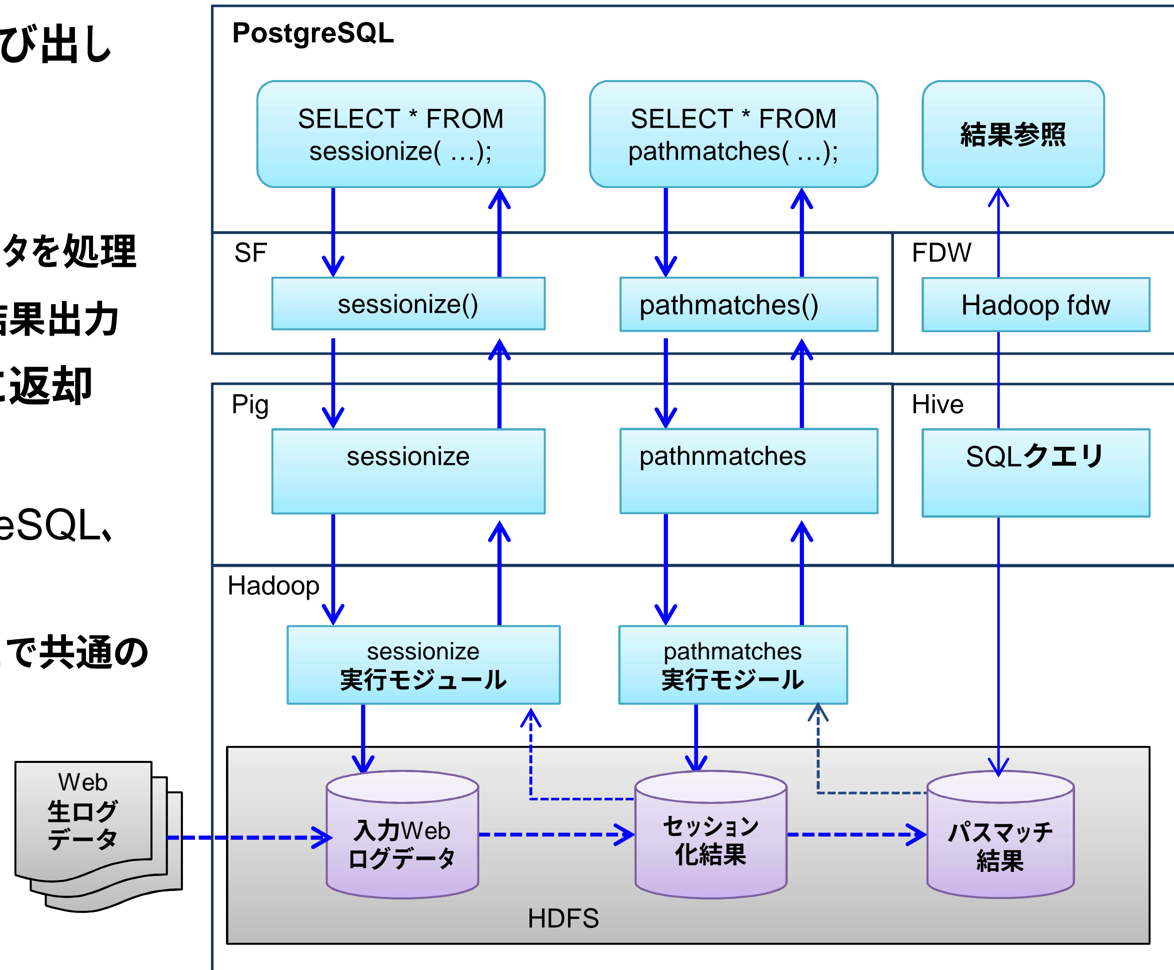


All Rights Reserved, Copyright © 2013 Nihon Unisys, Ltd.

ストアド関数からMapReduce実行まで

UNISYS

- PostgreSQLからストアド関数呼び出し
- PythonからPigスクリプト実行
- PigによるMapReduceの実行
  - ✓ MapによりHDFS上の入力データを処理
  - ✓ Reduce処理により、HDFSに結果出力
- 実行ステータスをPostgreSQLに返却
- 結果データは、HDFS上に出力
- メタデータの共有により、PostgreSQL、Hadoop間でデータを共有
  - ✓ 外部テーブル定義とHcatalogとで共通の定義



All Rights Reserved, Copyright © 2013 Nihon Unisys, Ltd.

ログのセッション化とは

UNISYS

- IPアドレスと時間間隔から、同一ユーザのアクセスログをグループ化
- セッションタイムアウト値より離れたログは別セッションと判断

入力データ (時系列ログ)

タイムスタンプ	参照先	IPアドレス	･･･
3/7 15:00:00	･･･	xxx.xxx.xxx.110	
3/7 15:00:12	･･･	xxx.xxx.xxx.220	
3/7 15:01:11	･･･	xxx.xxx.xxx.110	
3/7 15:01:33	･･･	xxx.xxx.xxx.220	
3/7 15:01:52	･･･	xxx.xxx.xxx.330	
3/7 16:36:12	･･･	xxx.xxx.xxx.110	
･･･	･･･	･･･	

※セッションタイムアウト値を10分とした場合

3/7 15:00:00	･･･	xxx.xxx.xxx.110

タイムスタンプが10分以内なので、同一セッションと判断

3/7 15:01:11	･･･	xxx.xxx.xxx.110

タイムスタンプが10分以上離れているので、別セッションと判断

3/7 16:36:12	･･･	xxx.xxx.xxx.110

セッション化

出力 (セッション化済みログ)

タイムスタンプ	参照先	IPアドレス	session番号
3/7 15:00:00	･･･	xxx.xxx.xxx.110	0
3/7 15:01:11	･･･	xxx.xxx.xxx.110	0
3/7 16:36:12	･･･	xxx.xxx.xxx.110	1
･･･	･･･	･･･	･･･

All Rights Reserved, Copyright © 2013 Nihon Unisys, Ltd.

セッショナイズ関数

UNISYS

- Sessionizeストアド関数

```
Sessionize (入力パス, 出力パス[, タイムアウト間隔[, デリミタ]])
```

- 入力パラメタ
  - 入力パス･出力パス： pigからのパス
    入力パスに入力データを配置
  - 出力パス　hdfs://<アドレス>:8020/user/postgres/
  - タイムアウト間隔．デフォルト ‘30m‘
  - デリミタ．デフォルト　タブ文字
- 入出力データ形式
  - 入力データ形式：1番目のカラムを日付時刻、2番目のカラムをセッション化のキー項目
  - 出力データ形式：入力データの最後にセッションIDの項目を追加したデータセット
- 返り値
  - 正常に出力ファイルが作成された場合、返り値0

PostgreSQLからのストアド関数呼び出しの例

```
SELECT Sessionize(‘input.data’, ‘output/XXX’, ‘10m’, ‘¥¥t’);
```

- セッショナイズ処理: Pigスクリプトにおける実装
  - DataFuのSessionize関数を利用

(注) DataFuは、Linkedinから公開されたPigのデータ分析用のUDFライブラリ
他にPageRank, Quantiles (median), varianceなどがある。

All Rights Reserved, Copyright © 2013 Nihon Unisys, Ltd.

セッショナイズ関数 (2)

UNISYS

ストアド関数Sessionize

```
CREATE FUNCTION sessionize
 (input text, output text, time_window text, delimiter text)
 RETURNS integer
 AS $$
  import commands
  esc_delimiter = delimiter.replace('¥¥', '¥¥¥¥¥¥¥¥')
  cmdstr = 'pig ' ¥
         + '-param INPUT_PATH=' + input + ' ' ¥
         + '-param OUTPUT_PATH=' + output + ' ' ¥
         + '-param TIME_WINDOW=' + time_window + ' ' ¥
                   + '-param DELIMITER="' + delimiter + '" ' ¥
         + '/home/postgres/pig/sessionize2.pig'
  r = commands.getstatusoutput(cmdstr)
  return r[0]
 $$ LANGUAGE plpythonu;
```

パラメタを動的にpigの引数として指定

Pigスクリプト実行

Pigスクリプトコード

```
%default DATAFU_JAR_PATH '/usr/lib/pig/datafu-0.0.4-cdh4.3.0.jar'
%default INPUT_PATH 'input.data'
%default OUTPUT_PATH 'output.data'
%default TIME_WINDOW '30m'
%default DELIMITER ' '
%default TIMESTAMP_COLUMN_NO 0
%default GROUPING_KEY_COLUMN_NO 1
%default DATA_COLUMN_NO 2
```

DataFu UDF登録

```
-- Register datafu.jar
register $DATAFU_JAR_PATH
-- Define Sessionize and set timeout param.
define Sessionize datafu.pig.sessions.Sessionize('$TIME_WINDOW');
```

データロード

```
-- Load input data
views = LOAD '$INPUT_PATH' USING PigStorage('$DELIMITER');
views = FOREACH views GENERATE
  (chararray)$0 as time, (chararray)$1 as uid, TOTUPLE(*) as v;

-- Group by uid
views_grouped= GROUP views BY uid;
```

セッション化

```
-- Sessionize
sessions = FOREACH views_grouped {
  views = ORDER views BY time;
  GENERATE FLATTEN(Sessionize(views));
}
```

結果セット出力

```
-- Create projection for store data
sessions = FOREACH sessions GENERATE FLATTEN($2), $3;
-- Store output data
STORE sessions INTO ‘$OUTPUT_PATH’ USING
PigStorage(‘$DELIMITER’);
```

All Rights Reserved, Copyright © 2013 Nihon Unisys, Ltd.

セッショナイズ処理の結果

UNISYS

■ セッションIDの付加

【入力】

```
2013-01-01T01:00:00Z,1,10
2013-01-01T01:15:00Z,1,20
2013-01-01T01:00:00Z,2,10
2013-01-01T01:31:00Z,2,20
```

入力： 不要なデータを除去したWebログ
ファイルの先頭カラムを日付時刻
二番目のカラムをセッション化のキー項目

セッション化

【出力】

出力： 入力データの最後にセッションID (UUID) のフィールドを追加

```
2013-01-01T01:00:00Z,1,10,aa3f14c7-5d04-4613-a1e0-9d122bfa6efa
2013-01-01T01:15:00Z,1,20,aa3f14c7-5d04-4613-a1e0-9d122bfa6efa
2013-01-01T01:00:00Z,2,10,423087a1-8631-4455-8708-5c0aa36f2532
2013-01-01T01:31:00Z,2,20,b3d48d6c-6016-46c5-bc0a-58b5644e01e2
```

■ 参考：SQLによるセッショナイズの例

```
CREATE TABLE webclicks (click_timestamp timestamp, ip varchar(30), url_keyword varchar(5000), original_data
varchar(5000));
WITH dt (IP, Click_Timestamp, samesession) AS (
SELECT IP, Click_Timestamp,
CASE WHEN EXTRACT (EPOCH FROM (Click_Timestamp - (max(Click_Timestamp) OVER (partition by IP order by
Click_Timestamp
ROWS BETWEEN 1 PRECEDING AND 1 PRECEDING) ) ) ) < 1800
THEN 0 ELSE 1 END FROM webclicks)
SELECT SUM(samesession) OVER (partition by IP order by  Click_Timestamp rows unbounded preceding) AS Session_No,
IP, Click_Timestamp FROM dt;
```

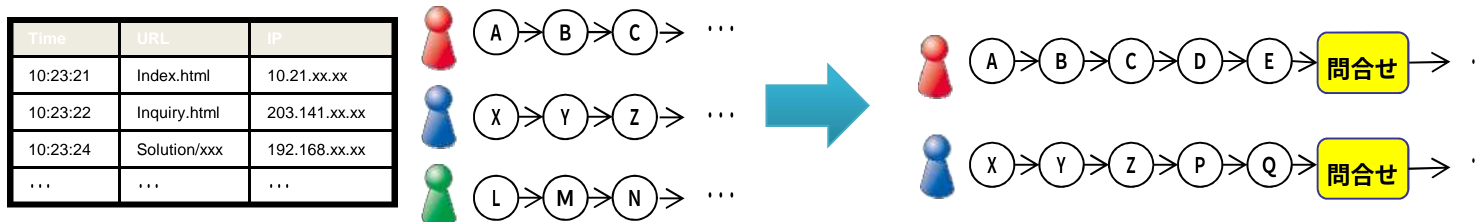
参考：http://developer.teradata.com/extensibility/articles/sessionization-map-reduce-support-in-teradata
(注) 検証のため一部修正

All Rights Reserved, Copyright © 2013 Nihon Unisys, Ltd.

アクセス経路分析

- Webサイト内をどのような経路でアクセスして「問い合わせフォーム」に到達したかを解析
  - Webサイトのアクセシビリティ改善
  - 製品・サービスの注目度の把握
- どのページからの問合せが多いか解析
  - 解析の前提
  - 問合せに至る直前のページを集計

- ストアド関数Pathmatchesによる動線分析
  - どのようなページを経由し、問い合わせページに到達したか
  - 問合せページの直前に見ていたページを特定



Time	URL	IP
10:23:21	Index.html	10.21.xx.xx
10:23:22	Inquiry.html	203.141.xx.xx
10:23:24	Solution/xxx	192.168.xx.xx
…	…	…

All Rights Reserved, Copyright © 2013 Nihon Unisys, Ltd.

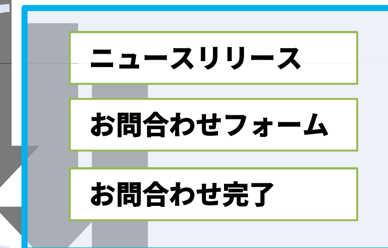
パスマッチとは

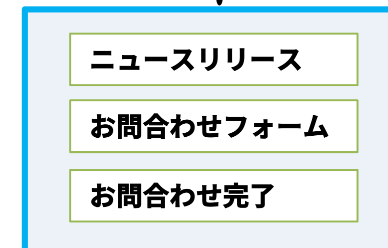
UNISYS

- 複数レコードから特定のパターンを検索する関数
  - 「ニュースリリース」→「お問合せフォーム」→「お問合せ完了」の順にアクセスしたユーザを抽出
- パスマッチのようなパターンマッチ処理は、SQLでは困難

アクセスログ

時刻	ユーザ	参照先
10:01	User1	ニュースリリース
10:02	User1	製品一覧
10:03	User1	ニュースリリース
10:04	User1	お問合わせフォーム
10:08	User1	お問合わせ完了
10:10	User1	ホーム
・・・		
10:02	User2	ニュースリリース
10:03	User2	お問合せフォーム
10:05	User2	ニュースリリース
10:06	User2	お問合せフォーム
10:08	User2	製品情報
・・・	・・・	・・・





(注) ニュースリリース、お問合わせフォームの間に、他のページが含まれている場合もある。

All Rights Reserved, Copyright © 2013 Nihon Unisys, Ltd.

パスマッチ関数

UNISYS

- Pathmatchesストアド関数
  「ニュースリリース」→「お問合せフォーム」→「お問合せ完了」の順にアクセスしたユーザを抽出
  - ✓ 「問い合わせフォーム以外のページ」→「問い合わせフォーム」→「問い合わせ完了」
  - ✓ 「問い合わせフォーム」に遷移する直前のページ
    ⇒ 「問い合わせフォーム以外のページ」の中で、最後にアクセスしたページ

```
pathmatches
 (input_path text, output_path text,  colno_group integer, colno_sort integer, colno_scan integer,
  regex text, get_group text [, delimiter text]);
```

- 入力パラメタ
  - input_path: 入力データのパス
  - output_path: 出力データのパス
  - colno_group: グループ化のキーとなる列の番号 (セッションID)
  - colno_sort: ソートのキーとなる列の番号 (日付・時刻)
  - colno_scan: マッチの対象となる列の番号 (URL文字列)
  - regex: マッチング及び返り値を定義する為に使用する文字列 (正規表現)
  - get_group: 返り値として取得したいグループの番号
    グループは括弧で囲われた文字列 (注) Pigのデータ表現。番号は先頭括弧の出現順
  - delimiter: 入力データのデリミタ(区切り文字)
- 入出力データ形式
  - 入力データ形式：sessionize関数の出力
  - 出力データ形式：セッションID、マッチング対象記号列(グループ) のうち、指定したグループ番号の文字列
- 返り値
  - 正常に出力ファイルが作成された場合、関数の返り値0を返却

All Rights Reserved, Copyright © 2013 Nihon Unisys, Ltd.

パスマッチ関数の動作

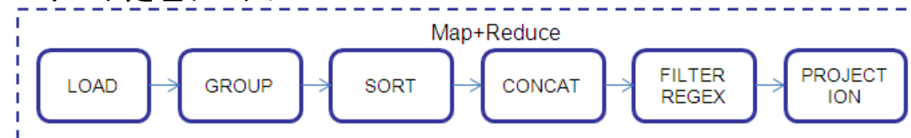
UNISYS

```
PatchMatchesストアド関数
       :
esc_regex = regex.replace('¥¥', '¥¥¥¥¥¥¥¥');
  esc_delimiter = delimiter.replace('¥¥', '¥¥¥¥¥¥¥¥');
  cmdstr = 'pig ' ¥
         + '-param INPUT_PATH="' + input + '" ' ¥
         + '-param OUTPUT_PATH="' + output + '" ' ¥
         + '-param GROUP_KEY=' + str(colno_group) + ' ' ¥
         + '-param SORT_KEY=' + str(colno_sort) + ' ' ¥
         + '-param SCAN_COL=' + str(colno_scan) + ' ' ¥
         + '-param REGEX="' + esc_regex + '" ' ¥
         + '-param GET_GROUP="' + str(get_group) + '" ' ¥
                  + '-param DELIMITER="' + esc_delimiter + '" ' ¥
```

(PathMatchesストアド関数一部)

- ストアド関数の引数を、pigの実行パラメタとして指定
- Pigで参照するデータ項目をストアド関数のパラメタで指定
- 入力データフィールドの参照は項目先頭からの番号

データ処理ワークフロー



- パスマッチ関数のデータ処理

データロード

[1] 入力データをグループ化キーでグルーピング (GROUP BY)
[処理前]
1, 10:01, "BBB"
1, 10:00, "AAA"
1, 10:02, "CCC"
2, 10:05, "bbb"
2, 10:00, "aaa"
[処理後]
1, ((1, 10:01, "BBB),(1, 10:00, "AAA"),(1, 10:02, "CCC"))
2, ((2, 10:05, "bbb),(2, 10:00, "aaa"))

グルーピング・ソート

[2] グループ化されたバッグ内のデータを指定したソートキーでソート (SORT)
[処理前]
1, ((1, 10:01, "BBB),(1, 10:00, "AAA"),(1, 10:02, "CCC"))
2, ((2, 10:05, "bbb),(2, 10:00, "aaa"))
[処理後]
1, ((1, 10:00, "AAA),(1, 10:01, "BBB"),(1, 10:02, "CCC"))
2, ((2, 10:00, "aaa),(2, 10:05, "bbb"))

URLフィールドを結合

[3] マッチ対象の列のデータを文字列連結
[処理前]
1, ((1, 10:00, "AAA),(1, 10:01, "BBB"),(1, 10:02, "CCC"))
2, ((2, 10:00, "aaa),(2, 10:05, "bbb"))
[処理後]
1, "AAA,BBB,CCC"
2, "aaa,bbb"

正規表現によるマッチング

[4] 連結されたマッチ対象列を正規表現にてフィルタリング
[処理前]
1, "AAA,BBB,CCC"
2, "aaa,bbb"
[処理後]
1, "AAA,BBB,CCC"

結果出力

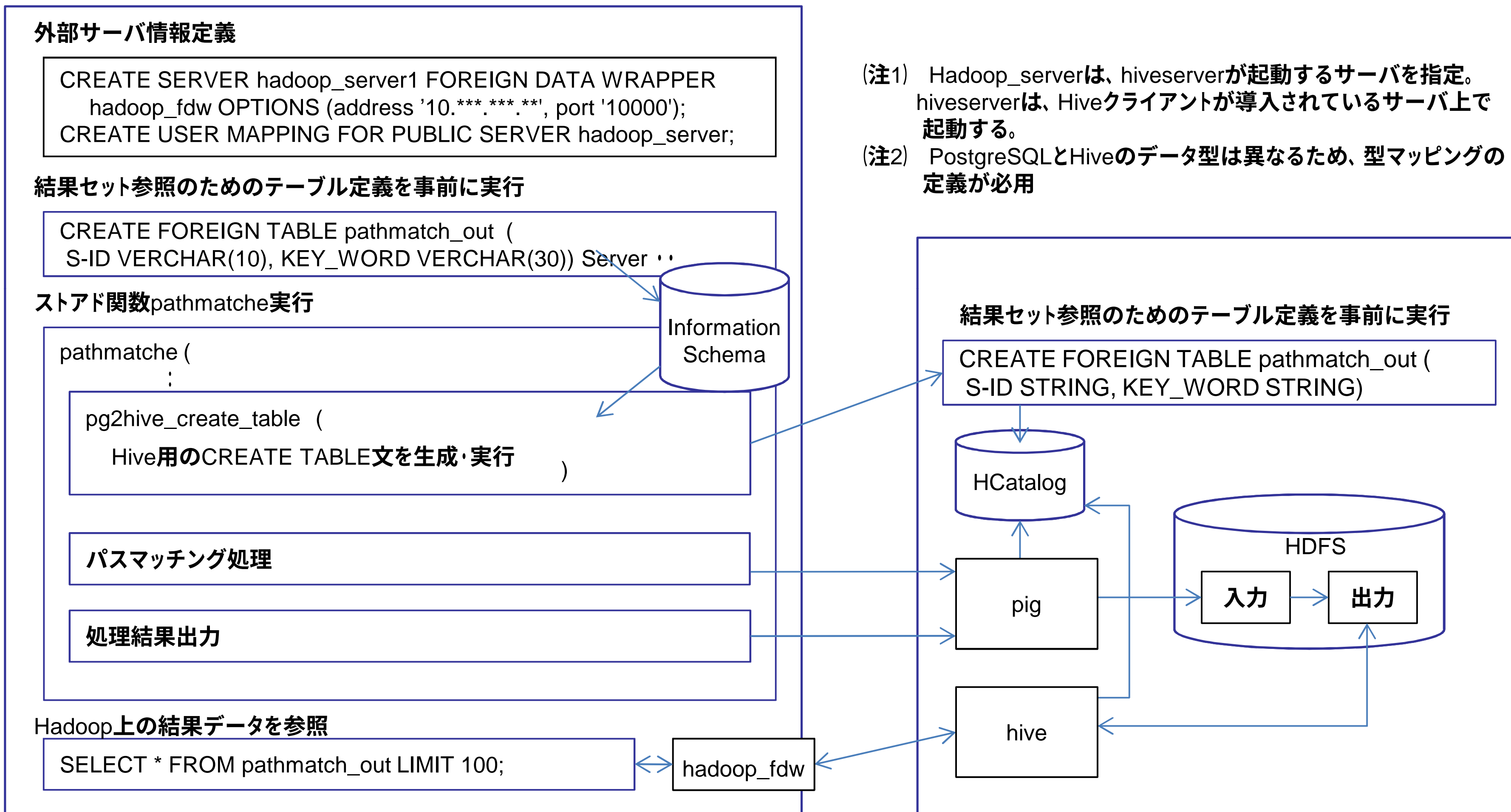
[5] 出力対象文字列グループを抽出
[処理前]
1, "AAA,BBB,CCC"
[処理後]
1, "AAA"

All Rights Reserved, Copyright © 2013 Nihon Unisys, Ltd.

Hadoop FDWによるHDFS上のデータ参照

UNISYS

- PigによりHDFS上に出力された結果データを、PostgreSQLからHive経由で参照



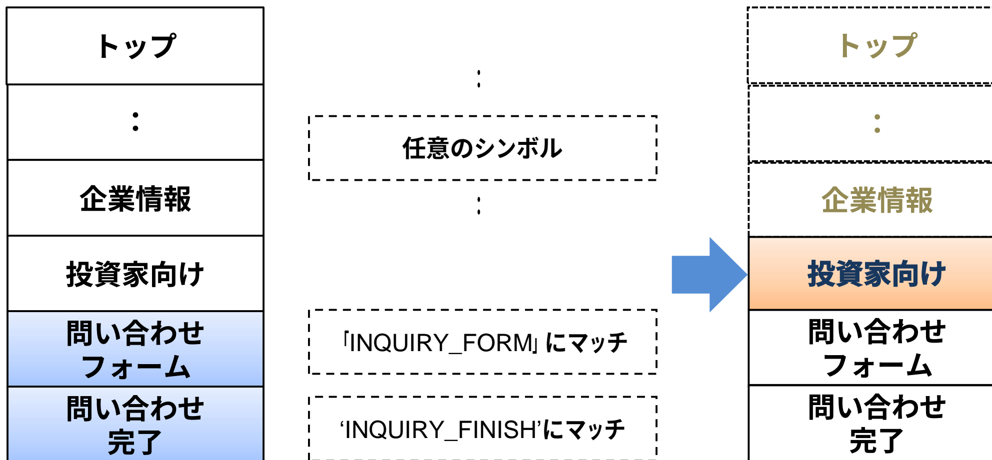
All Rights Reserved, Copyright © 2013 Nihon Unisys, Ltd.

パスマッチ関数によるパターンマッチ

UNISYS

- 「問合わせフォーム」の直前に参照したページを抽出
- PostgreSQLからのクエリ

```
SELECT PathMatches (
 'data/weblog/sessionized/XXX', 'data/weblog/pathmatched/XXX', 4, 0, 2,
 '([^,]+)(,INQUIRY_FORM)++,INQUIRY_FINISH', 1, '¥¥t');
```

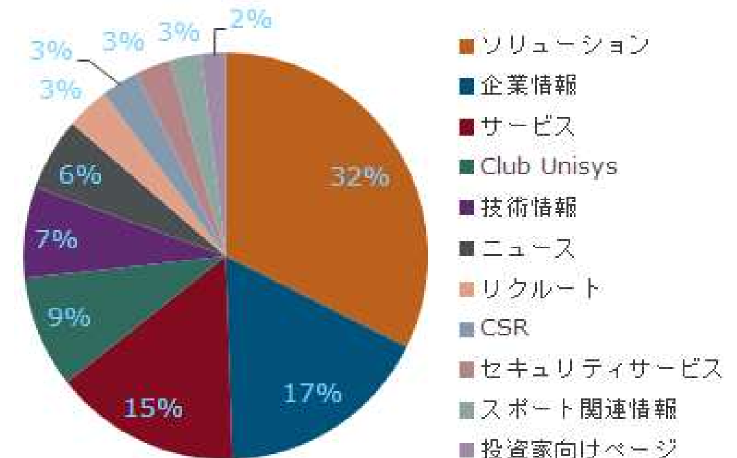


URLは、クレンジング処理で予めシンボル化
（例） INQUIRY_FORM：問い合わせフォーム
INQUIRY_FINISH：問い合わせ完了ページ
上記以外のシンボルを問い合わせ以外のページと判断

- 解析結果

問い合わせ直前に参照していたページ

ランク	ページカテゴリ
1	ソリューション
2	企業情報
3	サービス
4	Club Unisys
5	技術情報
6	ニュース
7	リクルート
8	CSR
9	セキュリティサービス
10	スポーツ関連情報

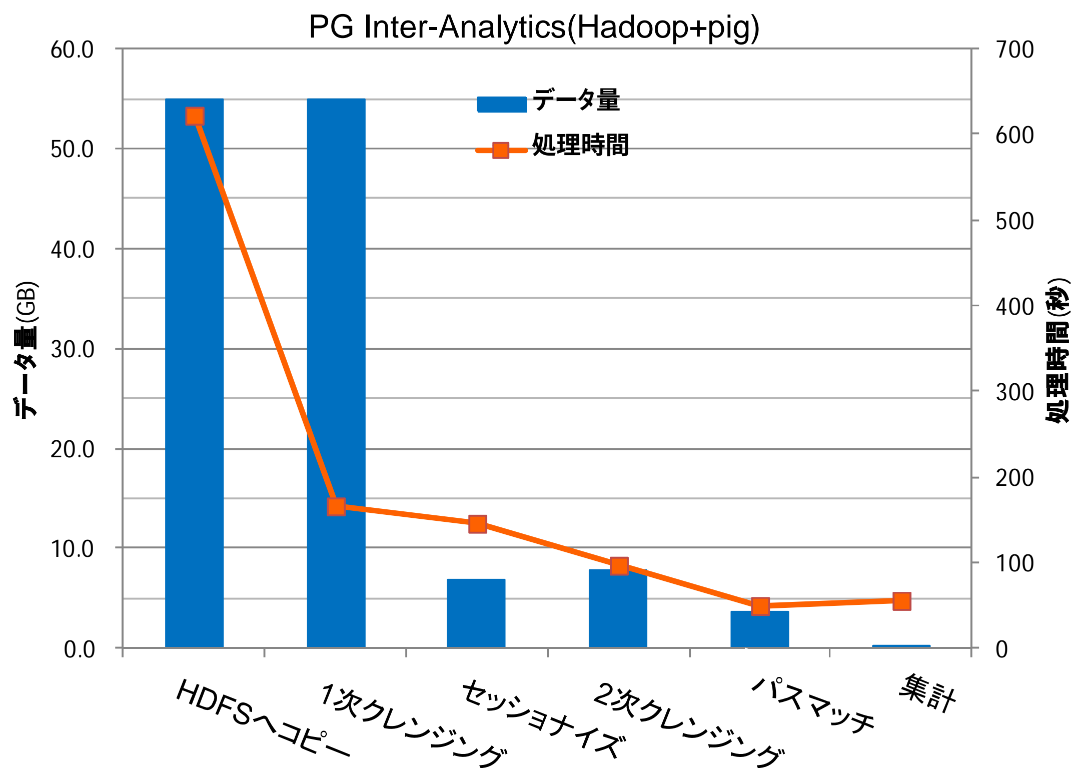


All Rights Reserved, Copyright © 2013 Nihon Unisys, Ltd.

効率的なログデータ処理とは

UNISYS

- 効率よいデータ処理のためには、データのクレンジングが重要
- 今回は、セッショナイズの段階でデータサイズを削減
- 大規模データ処理、複雑なデータ処理は、Hadoopが向く
  - 大規模データのクレンジング
  - 行、列にまたがるSQLでは表現が難しいパスマッチ処理
- パスマッチの結果や集計結果をPostgreSQLに取り込んで利用



All Rights Reserved, Copyright © 2013 Nihon Unisys, Ltd.

まとめ

UNISYS

- PG Inter-Analyticsとは、
  - PostgreSQLからHadoopへ透過的に連携
  - 両者をうまく用いて、さらに利用価値が高められる
    - SQLでは不得手なデータ処理 (大規模・非構造化データ)
    - Hadoopでは不得手なインターラクティブなデータ分析
- 次のステップに向けて
  - ログのクレンジング、成形処理も、関数として汎用化を検討
  - PostgreSQLとHadoopのデータ定義の共有
    - 汎用性のある入力データの参照方式
    - 出力結果の柔軟なデータ定義に対応
  - より動的にHadoop処理ロジックを制御
  - 多くの分析関数への対応
    - 確率統計、機械学習 etc.

All Rights Reserved, Copyright © 2013 Nihon Unisys, Ltd.

U&U

Users & Unisys

UNISYS