

instytut informatyki

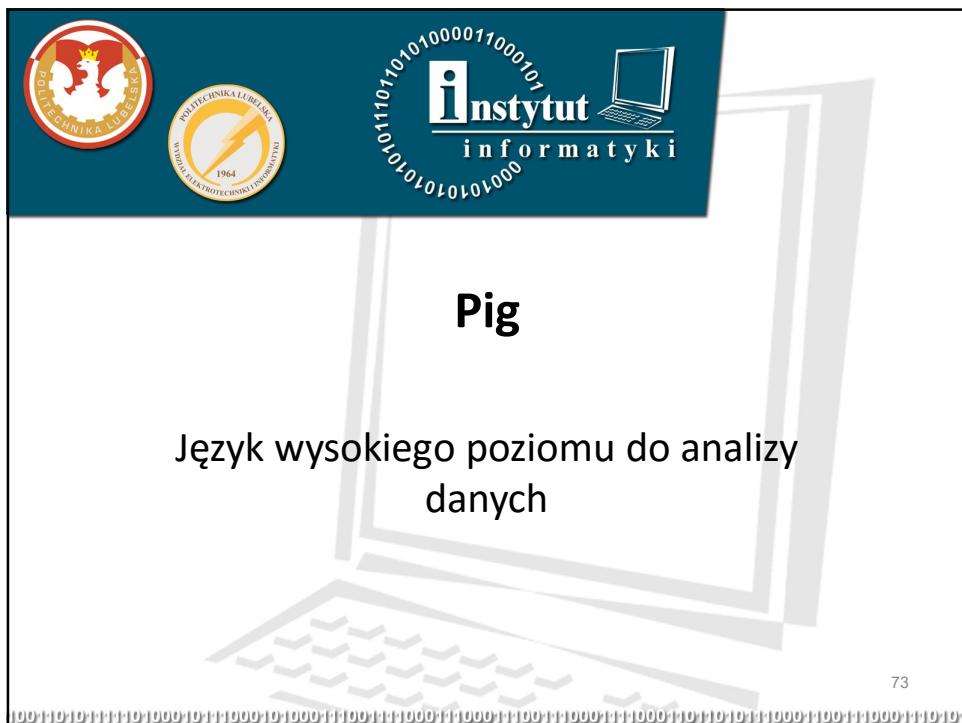
Podprojekty Hadoop

71

Środowiska wywodzące się z Hadoop

- Pig
 - Język wysokiego poziomu do analizy danych
- HBase
 - Składowanie w tabelach danych semistrukuralnych
- Zookeeper
 - Koordynacja aplikacji rozproszonych
- Hive
 - Język zapytań typu SQL oraz Metastore
- Mahout
 - Uczenie maszynowe

72

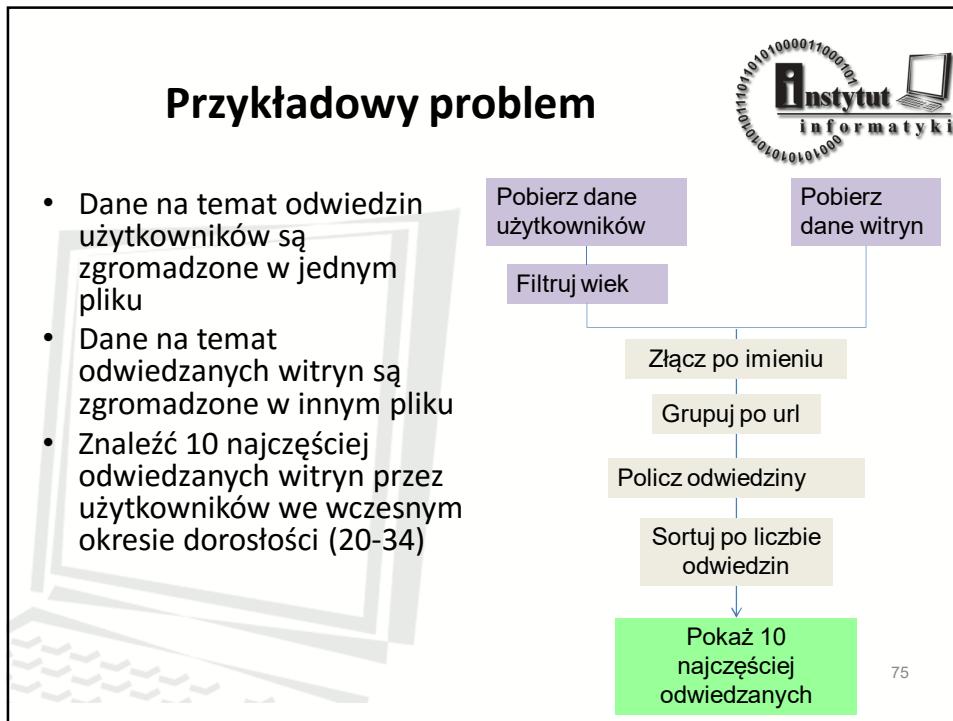


Pig

- Początki Yahoo! Research 2006
 - Około 30% zadań uruchamianych w Yahoo! to Pig Latin
 - Własności
 - Wyraża sekwencje w zadaniach MapReduce
 - Model danych: zagnieżdżone „worki” elementów
 - Dostarcza operatorów relacyjnych (SQL) takich jak (JOIN, GROUP BY, UNION, EXCEPT, ipt)
 - Łatwy do dołączenia w funkcjach Java

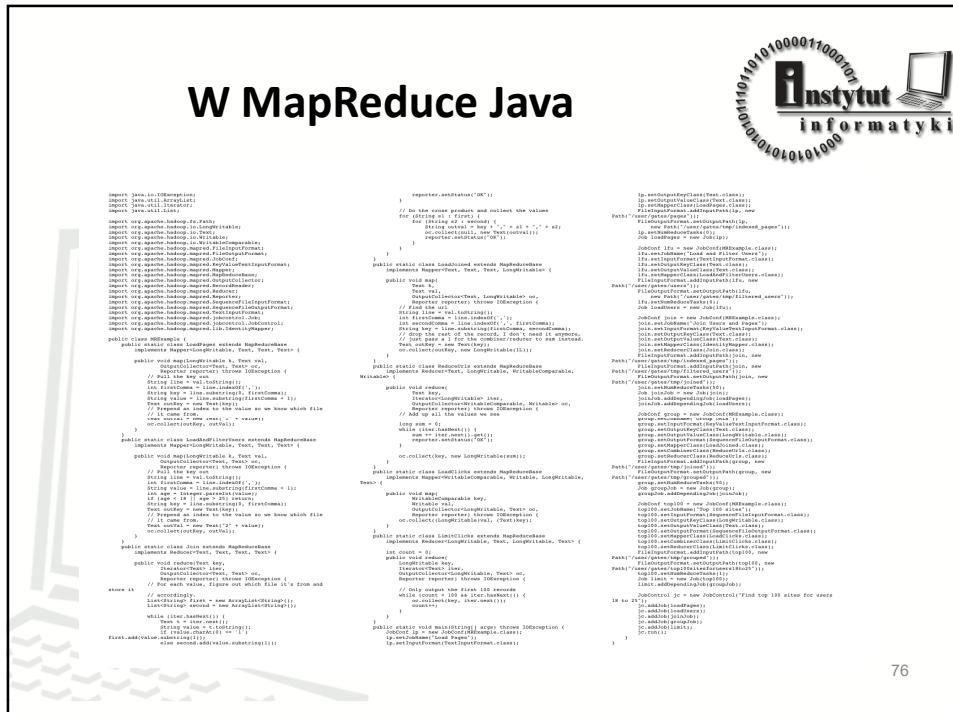


Przykładowy problem



75

W MapReduce Java



76

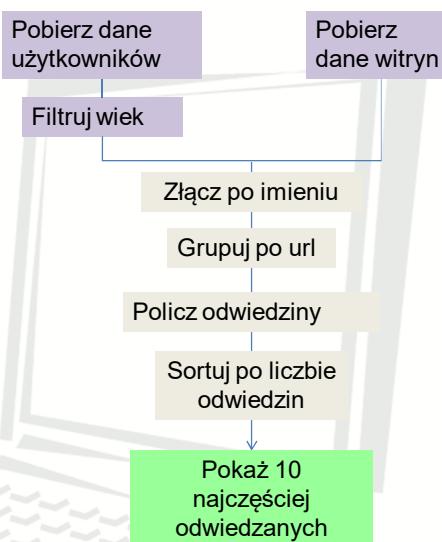
W Pig Latin



```
Uzytkownicy = load 'users' as (imie, wiek);
Filruj = filter Users by wiek >= 20 and wiek <=
    34;
Witryny = load 'pages' as (uzyt, url);
Zlacz = join Filruj by imie, Witryny by uzyt;
Grupuj = group Zlacz by url;
Sumuj = foreach Grupuj generate group,
        count(Zlacz) as odwiedziny;
Sortuj = order Sumuj by odwiedziny desc;
Top10 = limit Sortuj 10;
store Top10 into 'top10witryn';
```

77

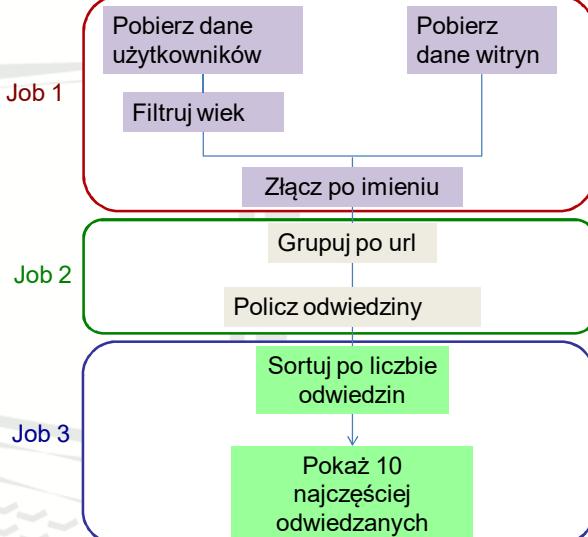
Łatwość translacji



```
Uzytkownicy = load ...;
Filruj = filter ...;
Witryny = load ...;
Zlacz = join ...;
Grupuj = group ...;
Sumuj = count...;
Sortuj = order...;
Top10 = limit Sortuj 10;
```

78

Łatwość translacji



79

Apache Pig



- Apache Pig jest platformą do analizy dużych zbiorów danych.
- Pig składa się z języka wysokiego poziomu do wyrażania programów analizy danych, w połączeniu z infrastrukturą do uruchamiania tych programów.

80

Apache Pig



- Warstwa infrastruktury Pig składa się z:
 - kompilatora, który generuje sekwencje programów Map-Reduce
 - warstwa języka Pig składa się obecnie z języka tekstu zwanego Pig Latin, który ma następujące główne właściwości:
 - Łatwość programowania.
 - Łatwo osiągnąć równoległe wykonywanie prostych zadań przetwarzających dane
 - Kompleksowe zadania składające się z wielu powiązanych ze sobą transformacji danych są wyraźnie zakodowane jako sekwencje przepływu danych, dzięki czemu są łatwe do napisania, zrozumienia i utrzymania.

81

Pig Latin



- Możliwości optymalizacji:
 - Sposób, w który są wpisane zadania umożliwia systemowi automatyczną optymalizację ich wykonania, co pozwala użytkownikowi na skupienie się na semantyce zamiast wydajności.
 - Rozszerzalność . Użytkownicy mogą tworzyć własne funkcje aby opracować przetwarzanie specjalnego przeznaczenia.

82

Uruchomienie Pig



- Aby wpisać polecenia Pig Latin należy:

– używając wiersza poleceń grunt

```
$ pig ... - Connecting to ...
```

```
grunt> dane = load 'data';
```

– w trybie lokalnym lub MapReduce

```
$ pig zadanie.pig
```

Przetwarzanie wsadowe w trybie lokalnym

```
$ pig -x local zadanie.pig
```

– zarówno interaktywnie jak i wsadowo

83

Organizacja przepływu programu



- Polecenie LOAD czyta dane z systemu plików
 - Szereg poleceń przekształcających przetwarza dane
 - Polecenie STORE zapisuje dane do systemu plików
 - Polecenie DUMP wyświetla wynik na ekranie

84

Interpretacja



- Ogólnie, Pig przetwarza wypowiedzi Pig Latin następująco:
 - Po pierwsze, Pig sprawdza składnię i semantykę wszystkich poleceń
 - Następnie, jeśli Pig napotka w kodzie DUMP lub STOR, Pig wykona wszystkie polecenia

85

Interpretacja



```
A = LOAD 'student' USING PigStorage() AS  
    (imie:chararray, wiek:int, srednia:float);
```

```
B = FOREACH A GENERATE imie
```

```
DUMP B;
```

```
(Jan)
```

```
(Maria)
```

```
(Krzysztof)
```

- Operator STORE zapisuje dane do pliku

86

Przykłady



```
punkty1 = LOAD 'input' AS (x, y, z);  
punkty2 = FILTER punkty1 BY x > 6;  
DUMP punkty2;  
punkty3 = FOREACH punkty2 GENERATE y, z;  
STORE punkty3 INTO 'output';  
  
punkty1 = LOAD 'input' AS (x, y, z);  
punkty2 = FILTER punkty1 BY x > 6;  
STORE punkty2 INTO 'output1';  
punkty3 = FOREACH punkty2 GENERATE y, z;  
STORE punkty3 INTO 'output2'
```

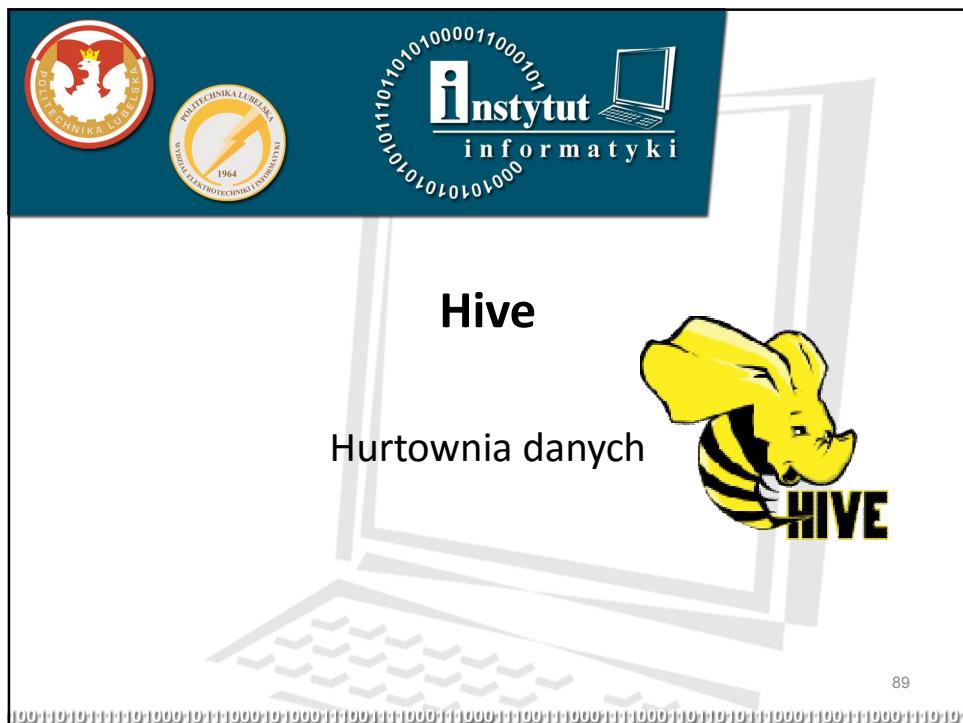
87

Więcej przykładów Pig Latin



- <http://www.cloudera.com/wp-content/uploads/2010/01/IntroToPig.pdf>

88



Środowisko hurtowni danych Hive

- Początkowo rozwijane przez Facebook
 - Używany do większości prac na Facebooku
 - Obecnie Apache Hive używany i rozwijany przez inne przedsiębiorstwa np. Netflix.
 - Amazon utrzymuje zmienioną gałąź (fork) oprogramowania w swoich produktach Elastic MapReduce i Amazon Web Services

Środowisko hurtowni danych Hive



- „Relacyjna” baza danych zbudowana na bazie Hadoop
 - Utrzymuje listę schematów tabeli
 - Język zapytań HiveQL bazuje na SQL
 - Można wywoływać skrypty Hadoop Streaming z HiveQL
 - Obsługuje partycjonowanie tabeli, klastry, złożone typy danych, niektóre optymalizacje

91

Hive – tworzenie tabeli



```
CREATE TABLE odwiedziny(czas0gl INT, userid BIGINT,
    data DATE, url_stony STRING, url_skoku STRING,
    adresip STRING COMMENT 'Adres IP użytk.')
COMMENT 'To jest tablica odwiedzin'
PARTITIONED BY(data STRING, panstwo STRING)
STORED AS SEQUENCEFILE;
```

- Partycjonowanie dzieli tabele na oddzielne pliki dla każdej pary (data, panstwo)
- Np: /hive/odwiedziny/data=2015-05-01,panstwo=PL
/hive/odwiedziny/data=2015-05-01,panstwo=GB

92

Zapytanie



- Znajdź wszystkie witryny pochodzące z domeny abc.pl do końca czerwca

```
SELECT odwiedziny.*  
FROM odwiedziny  
WHERE odwiedziny.data >= '2015-06-01'  
AND odwiedziny.data <= '2015-06-30'  
AND odwiedziny.url_skoku like '%abc.pl';
```

- Hive czyta tylko partycję 2105-06-01,* zamiast całą tabelę

93

Aggregation and Joins



- Policz użytkowników ,którzy wizytowali stronę ze względu na płeć:

```
SELECT odw.url_strony, uz.plec, COUNT(DISTINCT uz.id)  
FROM odwiedziny odw JOIN user uz ON (odw.userid = uz.id)  
GROUP BY odw.page_url, uz.plec  
WHERE odw.date = '2015-03-01';
```

- Przykładowe wyjście:

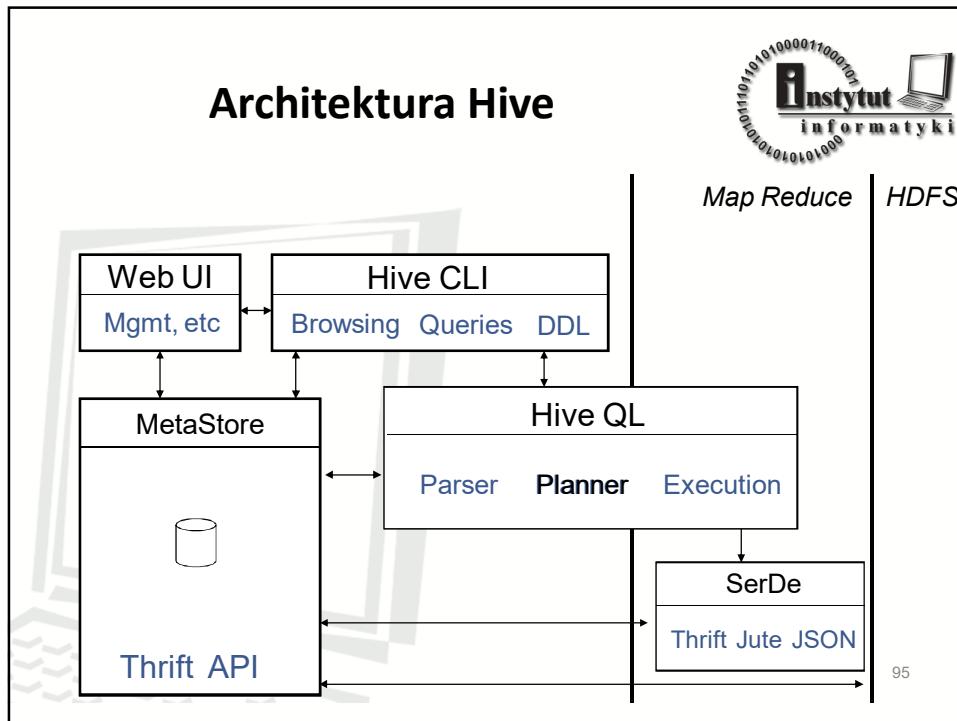
url_strony	plec	Count(uz.id)
index.php	M	13,140,320
index.php	F	15,315,450
galeria.php	M	30,344,482

94

Architektura Hive



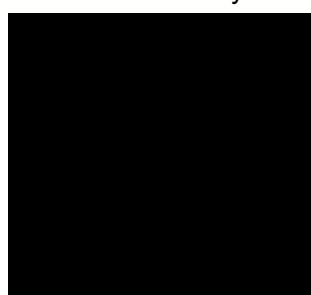
Map Reduce HDFS



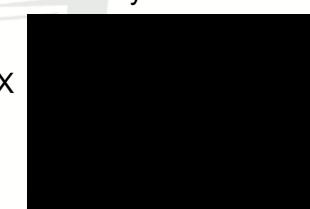
Hive QL – złączenia



odwiedziny



uzytkownik



odw_uzyt



X =

- **Hive SQL:**

```
INSERT INTO TABLE odw_uzyt
SELECT odw.idstr, uz.wiek
FROM odwiedziny odw JOIN uzytkownik uz ON
(odw.iduz = uz.iduz);
```

96

Hive QL – złączenie w Map Reduce



odwiedziny

użytkownik

Map

Shuffle
Sort

Reduc

97

Hive QL – klauzula GROUP BY



odw_uzyt

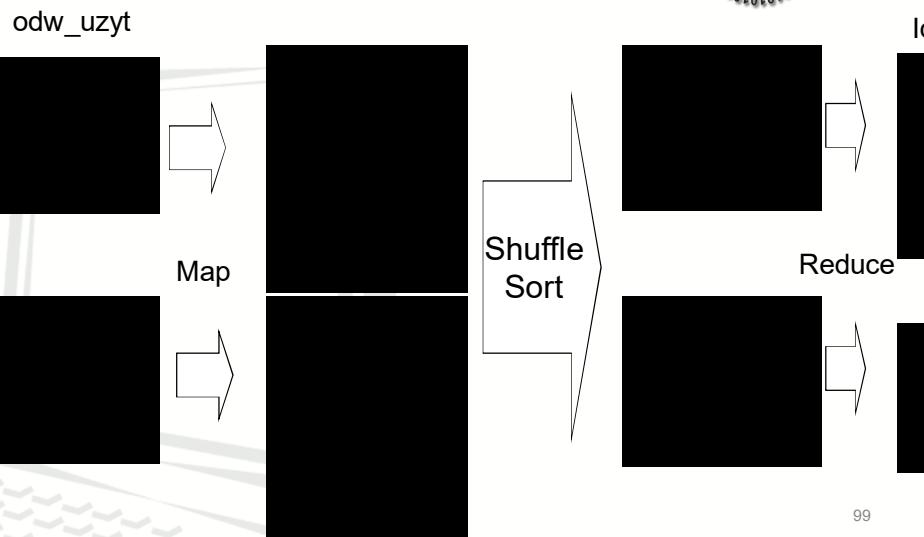
idstr_wiek_suma

- Hive SQL:

```
INSERT INTO TABLE idstr_wiek_suma
SELECT idstr, wiek, count(1)
FROM odw_uzyt
GROUP BY idstr, wiek;
```

98

Hive QL – klauzula GROUP BY w Map Reduce



Hive QL – wyrażenie GROUP BY z DISTINCT



odwiedziny

rezultat

- **Hive SQL**

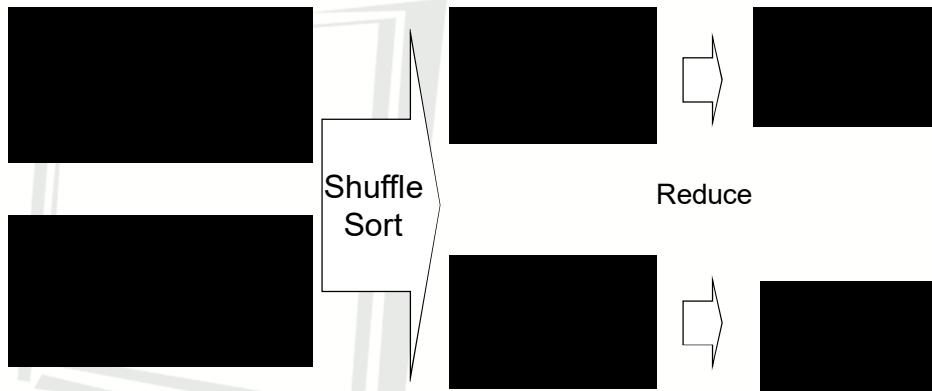
- `SELECT idstr, COUNT(DISTINCT iduz)`
- `FROM odwiedziny GROUP BY idstr`

100

Hive QL – wyrażenie GROUP BY z DISTINCT w Map Reduce



odwiedziny



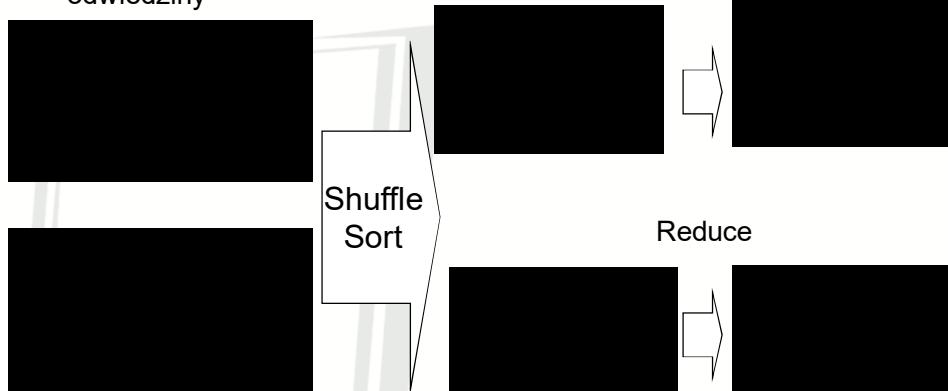
Klucz tasowania jest prefiksem klucza sortowania.

101

Hive QL – klauzula ORDER BY



odwiedziny



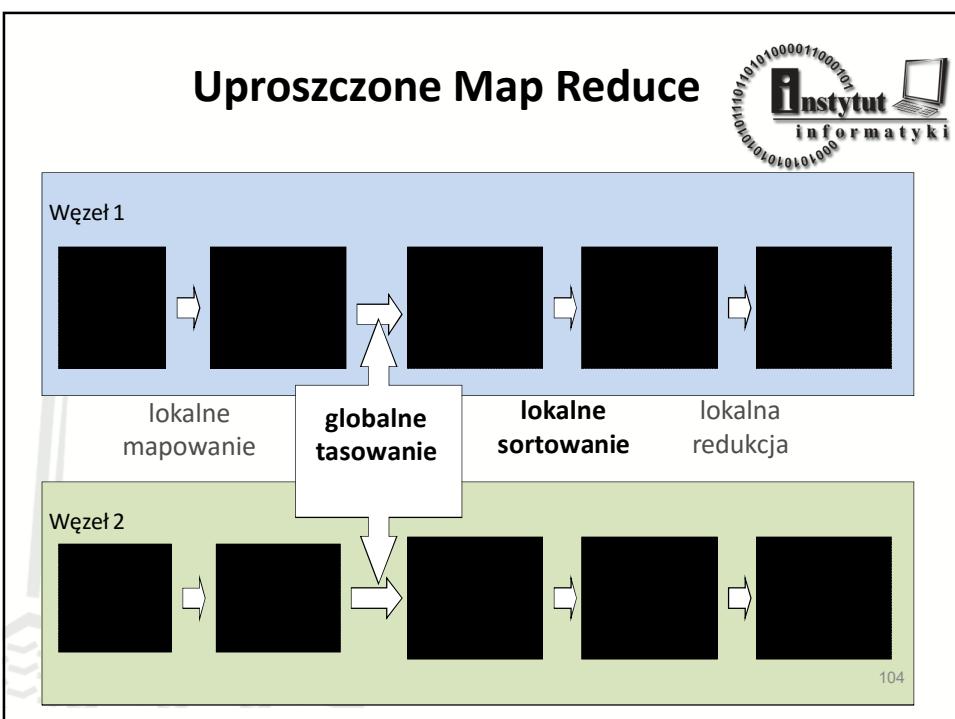
Szuflowanie losowo

102

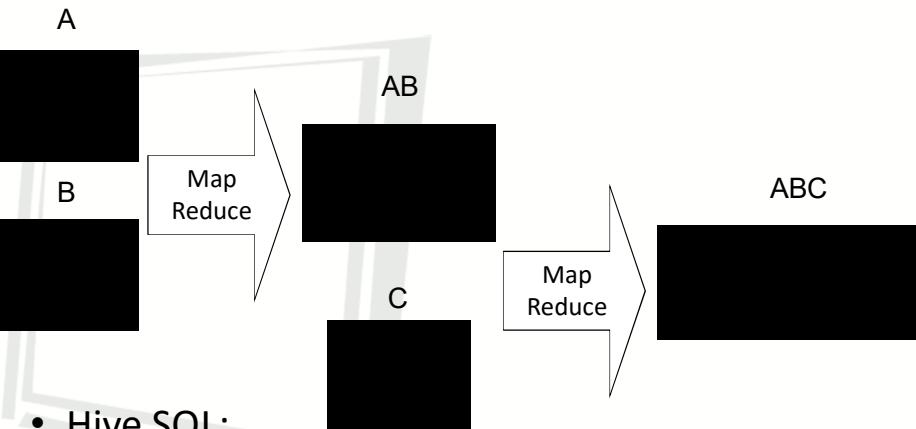
instytut informatyki

Wydajne wykonywanie zapytań SQL
z wykorzystaniem Map Reduce

103



Scalane sekwencyjne zadań Map Reduce



- Hive SQL:

```
- FROM (A join B on A.key = B.key)  
join C on A.key = C.key SELECT ...
```

105

Wspólne operacje odczytu



- Rozszerzony SQL

```
FROM odw_uzytk  
INSERT INTO TABLE  
odw_idstr_suma  
SELECT idstr,  
count(1)  
GROUP BY idstr  
INSERT INTO TABLE  
odw_wiek_suma  
SELECT wiek,  
count(1)  
GROUP BY wiek;
```

106

Problem lokalnego zbalansowania



odw_uzyt

Map
Reduce

idstridostekwieknaumesc

107

Agregacja map



Węzeł 1

lokalne
mapowanie

globalne
sortowanie

lokalne
sortowanie

lokalna
redukcja

Węzeł 2

108

Przepisywanie zapytań



- predykat push-down (stosu)
 - select * from (select * from tabela) where col1 = '2015';
- Przycinanie kolumn
 - select col1, col3 from (select * from tabela);

109

Przykłady



```
CREATE EXTERNAL TABLE blokowani_uzyt(
    iduzyt INT,
    blokowany INT,
    bloker INT,
    utworzono BIGINT)
STORED BY
    'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
WITH SERDEPROPERTIES ("hbase.columns.mapping" =
":key,f:blokowany,f:bloker,f:utworzono")
TBLPROPERTIES("hbase.table.name" = "m2h_repl-
userdb.stumble.blokowani_uzyt");
```

- HBase w tym przypadku posiada unikalny klucz wiersza mapy o nazwie :key
- nie wszystkie kolumny w tabeli wymagają mapowania

110

Przykłady



```
CREATE EXTERNAL TABLE ratings_hbase(
    userid INT, created BIGINT, urlid INT,
    rating INT, topic INT, modified BIGINT)
STORED BY
    'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
WITH SERDEPROPERTIES ("hbase.columns.mapping" =
    ":key#b@0,:key#b@1,:key#b@2,default:rating#b,default:topic#b,default:modified#b")
TBLPROPERTIES("hbase.table.name" =
    "ratings_by_userid");
```

- #b oznacza binarny, @ oznacza pozycję w kluczu złożonym

111



Różnice Hive/SQL - zapytania



Function	MySQL	HiveQL
Retrieving information	SELECT from_columns FROM table WHERE conditions;	SELECT from_columns FROM table WHERE conditions;
All values	SELECT * FROM table;	SELECT * FROM table;
Some values	SELECT * FROM table WHERE rec_name = "value";	SELECT * FROM table WHERE rec_name = "value";
Multiple criteria	SELECT * FROM table WHERE rec1="value1" AND rec2="value2";	SELECT * FROM TABLE WHERE rec1 = "value1" AND rec2 = "value2";
Selecting specific columns	SELECT column_name FROM table;	SELECT column_name FROM table;
Retrieving unique output records	SELECT DISTINCT column_name FROM table;	SELECT DISTINCT column_name FROM table;
Sorting	SELECT col1, col2 FROM table ORDER BY col2;	SELECT col1, col2 FROM table ORDER BY col2;
Sorting backward	SELECT col1, col2 FROM table ORDER BY col2 DESC;	SELECT col1, col2 FROM table ORDER BY col2 DESC;
Counting rows	SELECT COUNT(*) FROM table;	SELECT COUNT(*) FROM table;
Grouping with counting	SELECT owner, COUNT(*) FROM table GROUP BY owner;	SELECT owner, COUNT(*) FROM table GROUP BY owner;
Maximum value	SELECT MAX(col_name) AS label FROM table;	SELECT MAX(col_name) AS label FROM table;
Selecting from multiple tables (Join same table using alias w/"AS")	SELECT pet.name, comment FROM pet, event WHERE pet.name = event.name;	SELECT pet.name, comment FROM pet JOIN event ON (pet.name = event.name);



Różnice Hive/SQL - metadane



Function	MySQL	HiveQL
Selecting a database	USE database;	USE database;
Listing databases	SHOW DATABASES;	SHOW DATABASES;
Listing tables in a database	SHOW TABLES;	SHOW TABLES;
Describing the format of a table	DESCRIBE table;	DESCRIBE (FORMATTED EXTENDED) table;
Creating a database	CREATE DATABASE db_name;	CREATE DATABASE db_name;
Dropping a database	DROP DATABASE db_name;	DROP DATABASE db_name (CASCADE);

113



Hive wiersz poleceń



Function	Hive
Run query	hive -e 'select a.col from tab1 a'
Run query silent mode	hive -S -e 'select a.col from tab1 a'
Set hive config variables	hive -e 'select a.col from tab1 a' -hiveconf hive.root.logger=DEBUG,console
Use initialization script	hive -i initialize.sql
Run non-interactive script	hive -f script.sql

114

Hive shell



Function	Hive
Run script inside shell	source file_name
Run ls (dfs) commands	dfs -ls /user
Run ls (bash command) from shell	!ls
Set configuration variables	set mapred.reduce.tasks=32
TAB auto completion	set hive.<TAB>
Show all variables starting with hive	set
Revert all variables	reset
Add jar to distributed cache	add jar jar_path
Show all jars in distributed cache	list jars
Delete jar from distributed cache	delete jar jar_name

115

Hive - typy danych



- Typy kolumn
- Literały
- Wartości NULL
- Typy złożone

116

Typy kolumnowe



- Liczby całkowite

Type	Postfix	Example
TINYINT	Y	10Y
SMALLINT	S	10S
INT	-	10
BIGINT	L	10L

117

Typy kolumnowe



- Łącuchy
 - mogą być w (‘’) lub (“ ”)

Data Type	Length
VARCHAR	1 to 65355
CHAR	255

118

Typy kolumnowe



- **Czas**
- Wsparcie dla tradycyjnego znacznika czasu UNIX z opcjonalną precyzją nanosekundową.
- Wsparcie dla formatu:
 - java.sql.Timestamp
“YYYY-MM-DD HH:MM:SS.fffffffff”
 - oraz
 - “yyyy-mm-dd hh:mm:ss.fffffffff”.

119

Typy kolumnowe



- Daty
 - wartości DATE są w formie year/month/day {{YYYY—MM—DD}}.
- Wartości rzeczywiste
- Typ DECIMAL w systemie Hive jest taki sam jak Big Decimal format dla j. Java.
- Jest używany do reprezentowania niezmiennej dowolnej precyzji:
 - Składnia:
 - DECIMAL(precision, scale)
 - np. decimal(10,0)

120

Typy UNION



- Unia jest zbiorem różnorodnych typów danych.
- Można utworzyć instancję za pomocą **create union**.
- Przykładowa składnia:
- UNIONTYPE<int, double, array<string>, struct<a:int,b:string>>

121

Typy UNION



- {0:1} //int
- {1:2.2} //double
- {2:["trzy", "cztery"]} //array<string>
- {3:{"a":5,"b":„pięć”}} //struct<a:int,b:string>
- Zmiana wartości
- {2:["sześć","siedem"]}
- {3:{"a":8,"b":"osiem"}}

122

Literały



- W Hive używane są następujące literały:
- Typy Floating Point
- Typy zmienno-przecinkowe są skomponowane z typu DOUBLE.
- Typy Decimal
- Typ DECIMAL jest wartością zmiennoprzecinkową większą niż DOUBLE.
Zakres ten wynosi około -10^{-308} do 10^{308} .

123

Wartości NULL



- Wartości brakujące uzupełniane są poprzez specjalną wartość NULL

124

Typy złożone



- Typami złożonymi w Hive są:
- **Arrays** – tablice, używane w analogiczny sposób jak w j. Java
 - Składnia: ARRAY<data_type>
- **Maps** – mapy są podobne do map Java
 - Składnia: MAP<primitive_type, data_type>
- **Structs** – struktury w Hive są podobne do danych złożonych posiadających komentarz
 - Składnia: STRUCT<col_name : data_type [COMMENT col_comment], ...>

125

Hadoop dodatkowe źródła



- Przewodnik Pig Latin
 - <http://hortonworks.com/hadoop-tutorial/how-to-use-basic-pig-commands/>
- Przewodnik Hive
 - <http://www.tutorialspoint.com/hive/>

126