Złączenia w modelu Map/Reduce

Maciej Penar

Autor

Maciej Penar

- Typ Szalonego Naukowca - „Udowodnię Ci że to zadziała (nie powinno)”
- Aktualnie związany z firmą Vulcan S.p. z o.o.
- Prywatnie pesymista i sceptyk

Agenda

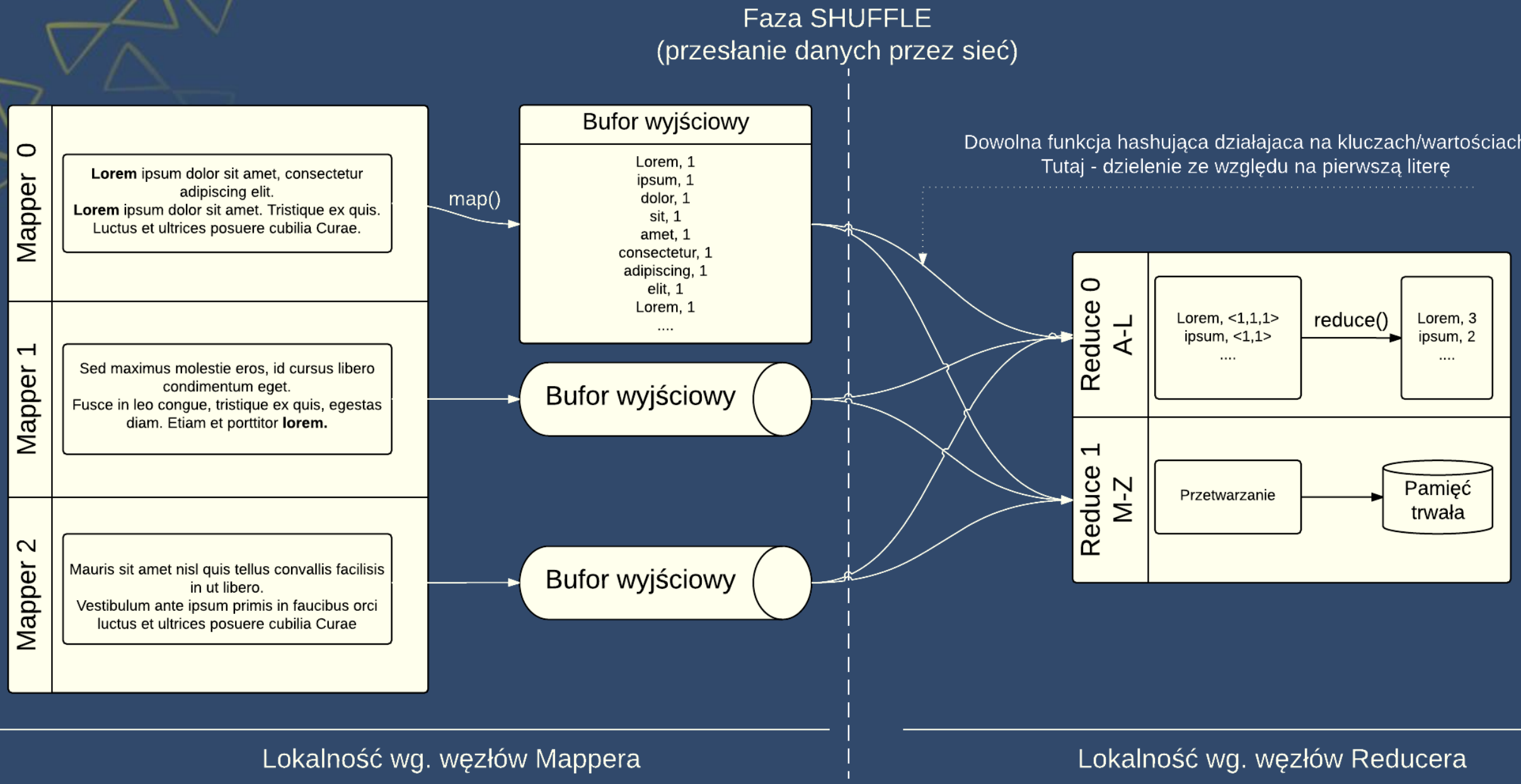
- Model programistyczny
  - Idea – 15 minut
- Złączenia
  - Podejście klasyczne – 20 minut
  - Podejście z równoważeniem obciążenia – 12 minut
  - Wydajność – 3 minuty
- Podsumowanie

+ Hive

Map-Reduce

Idea

- Systemy MapReduce
- Model projektowania programów o dużym stopniu zrównoleglenia:
  - Pierwsza faza Map (Grupowanie)
  - Druga faza Reduce (Agregacja)
- Obiekty wykonujące (niezależne)
  - Mapper
  - Reducer
- Cykl życia dla każdego obiektu
- Java (Hadoop)


Faza SHUFFLE
(przesłanie danych przez sieć)
Mapper 0
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Lorem ipsum dolor sit amet. Tristique ex quis.
Luctus et ultrices posuere cubilia Curae.
Mapper 1
Sed maximus molestie eros, id cursus libero condimentum eget.
Fusce in leo congue, tristique ex quis, egestas diam. Etiam et porttitor lorem.
Mapper 2
Mauris sit amet nisl quis tellus convallis facilisis in ut libero.
Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae
map()
Bufor wyjściowy
Lorem, 1
ipsum, 1
dolor, 1
sit, 1
amet, 1
consectetur, 1
adipiscing, 1
elit, 1
Lorem, 1
....
Bufor wyjściowy
Bufor wyjściowy
Dowolna funkcja hashująca działajaca na kluczach/wartościach
Tutaj - dzielenie ze względu na pierwszą literę
Reduce 0
A-L
Lorem, <1,1,1>
ipsum, <1,1>
....
reduce()
Lorem, 3
ipsum, 2
....
Reduce 1
M-Z
Przetwarzanie
Pamięć trwała
Lokalność wg. węzłów Mappera
Lokalność wg. węzłów Reducera

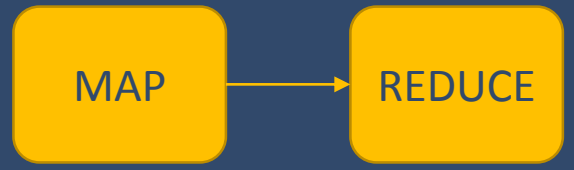
Map-Reduce

Idea

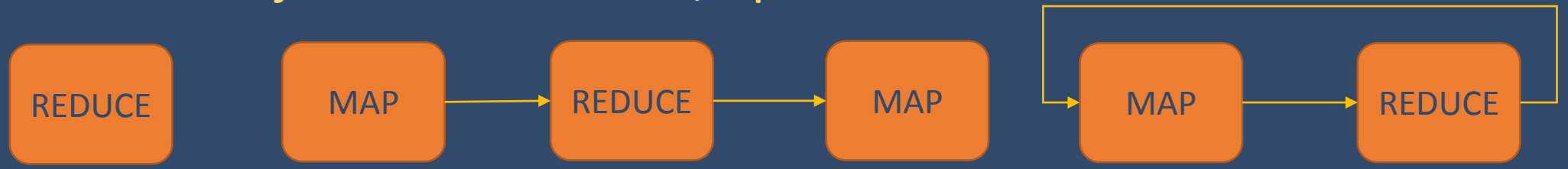
Dozwolone kombinacje programów MapReduce:

- Map-only

- Map-Reduce



Każda inna jest niedozwolona, np.:



Model programistyczny

Idea

Czy to wszystko?

… w rozważaniach teoretycznych przyjmuje się następujący model:

map(k1,v1) -> list(k2,v2)
reduce(k2, list(v2)) -> list(v3)

Często krytykowana prostota:
MapReduce: A major step backwards; DeWitt, Stonebreaker

Wracając do pytania: nie

Złączenia

Problem

Motywacja:

- Dane w postaci znormalizowanej
- Istnienie naturalnych zależności pomiędzy danymi

Problem:

- Złączenie N relacji (n-ta relacja oznaczana R_n) rozumiemy jako podzbiór iloczynu kartezjańskiego pomiędzy tymi źródłami, gdzie każda krotka tego podzbioru spełniają pewien warunek θ
- Za pomocą algebry relacji: $R_1 \bowtie_{\theta_1} \dots \bowtie_{\theta_{n-1}} R_n \equiv \sigma_\theta(R_1 \times \cdots \times R_n)$
- Dla nierozproszonych BD możemy wyróżnić trzy klasy algorytmów:
  - Oparte na pętli zagnieżdżonej
  - Oparte o sortowanie (Sort-Merge join)
  - Oparte o hashowanie (Hash join)

Złączenia

Problem

SELECT

*

FROM

R_1

INNER JOIN R_2 ON θ_1

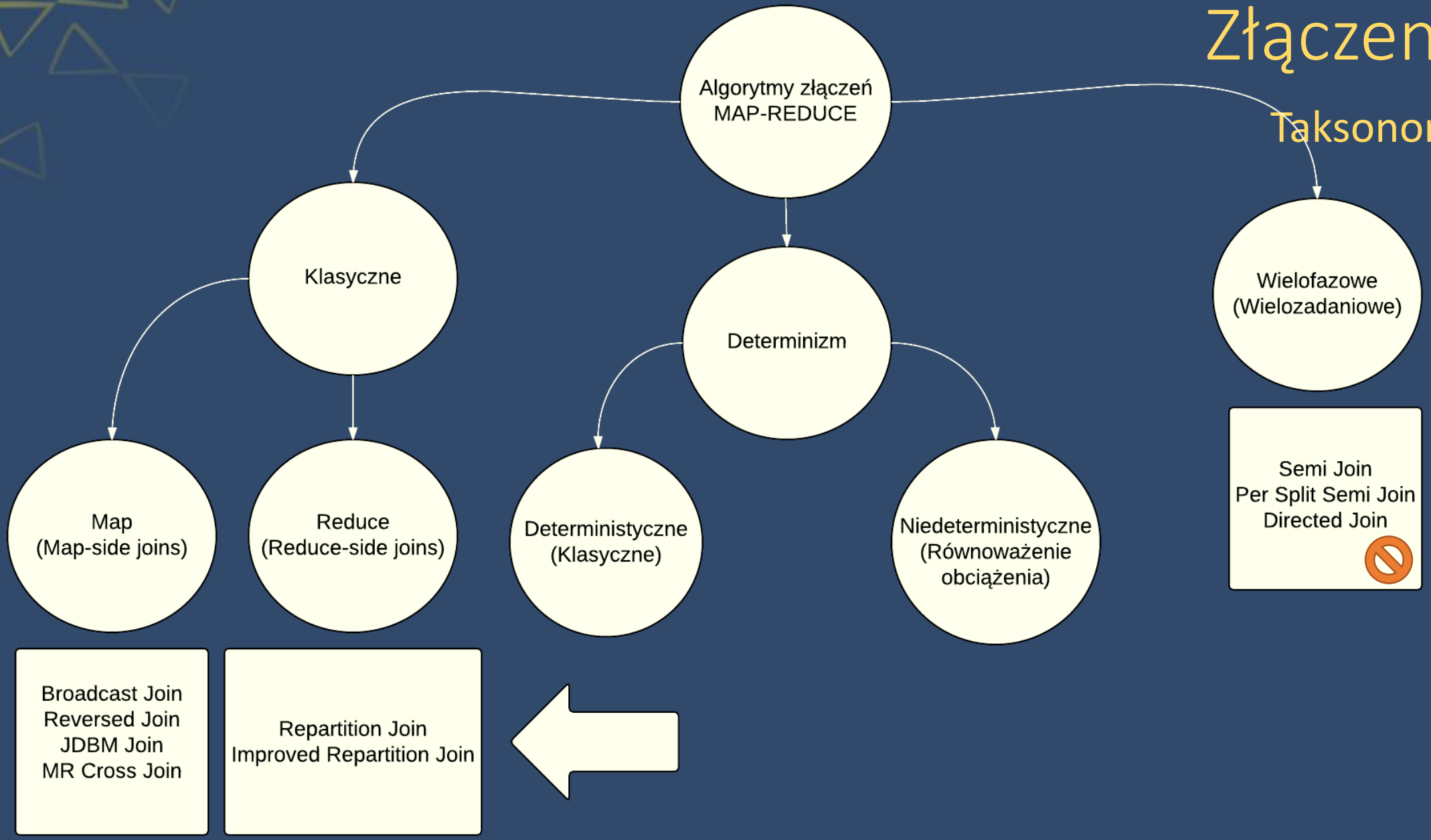
INNER JOIN R_3 ON θ_2

….

INNER JOIN R_N ON θ_{N-1}

Złączenia

Taksonomia



Złączenia

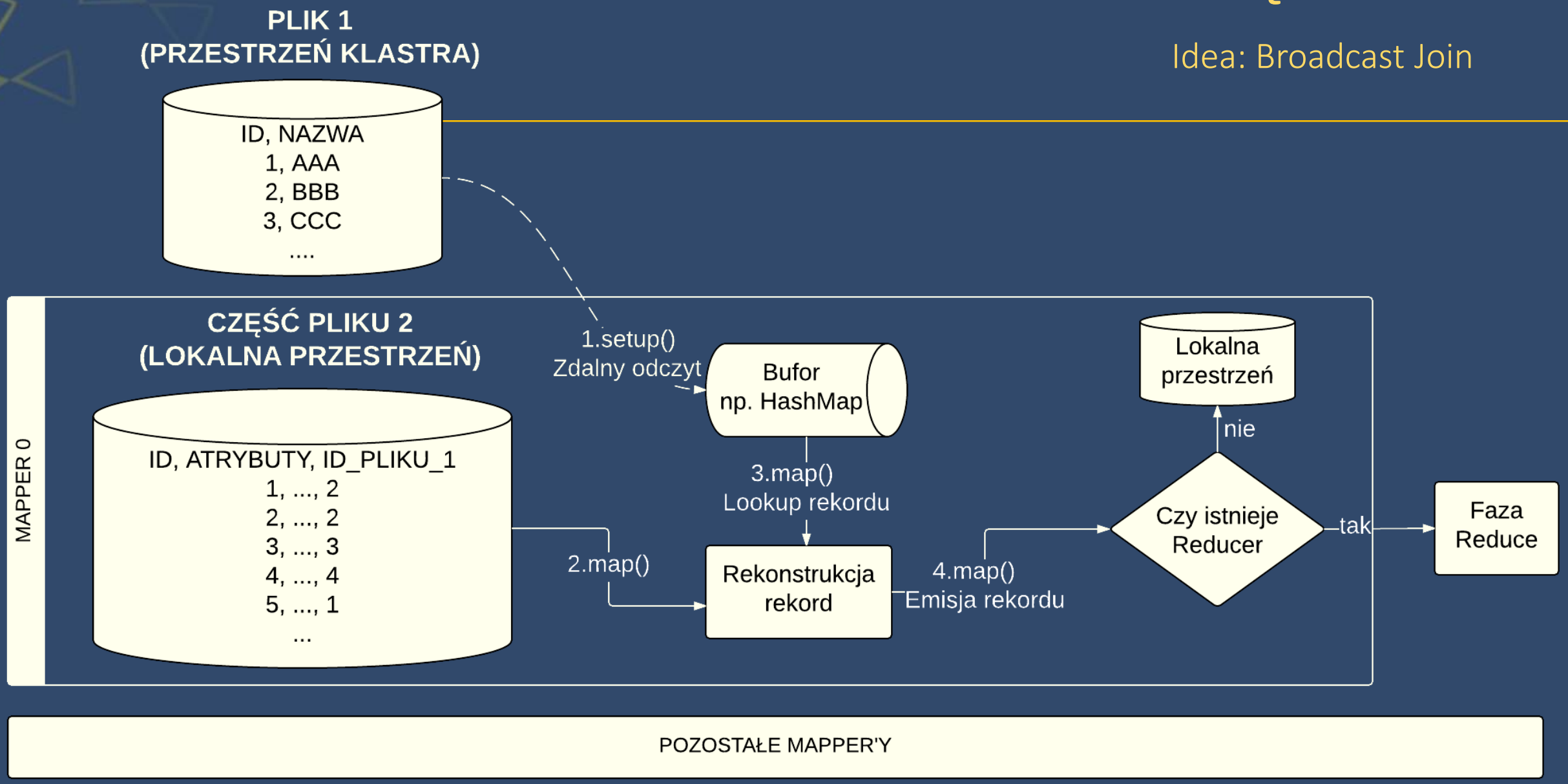
Klasyczne -> Map-Side Joins -> Broadcast Join

Koncepcja:

- Jeden (lub kilka) plików rozesłać (rozgłaszać) do wszystkich Mapperów za pomocą mechanizmu *Distributed Cache* (rozgłoszone pliki nie są traktowane jako wejściowe)
- Zbudować lokalny indeks na kluczu obcym
- Wczytywać rekordy i łączyć za pomocą indeksu

Złączenia

Idea: Broadcast Join



Złączenia

Klasyczne -> Map-Side Joins -> Broadcast Join

Zalety:

- Obsługa theta - złączeń
- Cache’owany plik zostaje na węźle (przez jakiś czas)
- Brak fazy shuffle
- Wielozłączenia

Wady:

- Wszystkie węzły muszą pobrać plik
- OutOfMemoryException dla dużych plików
- Brak fazy Reduce: brak sortowania, liczba plików wejściowych równa liczbie użytych Mapperów

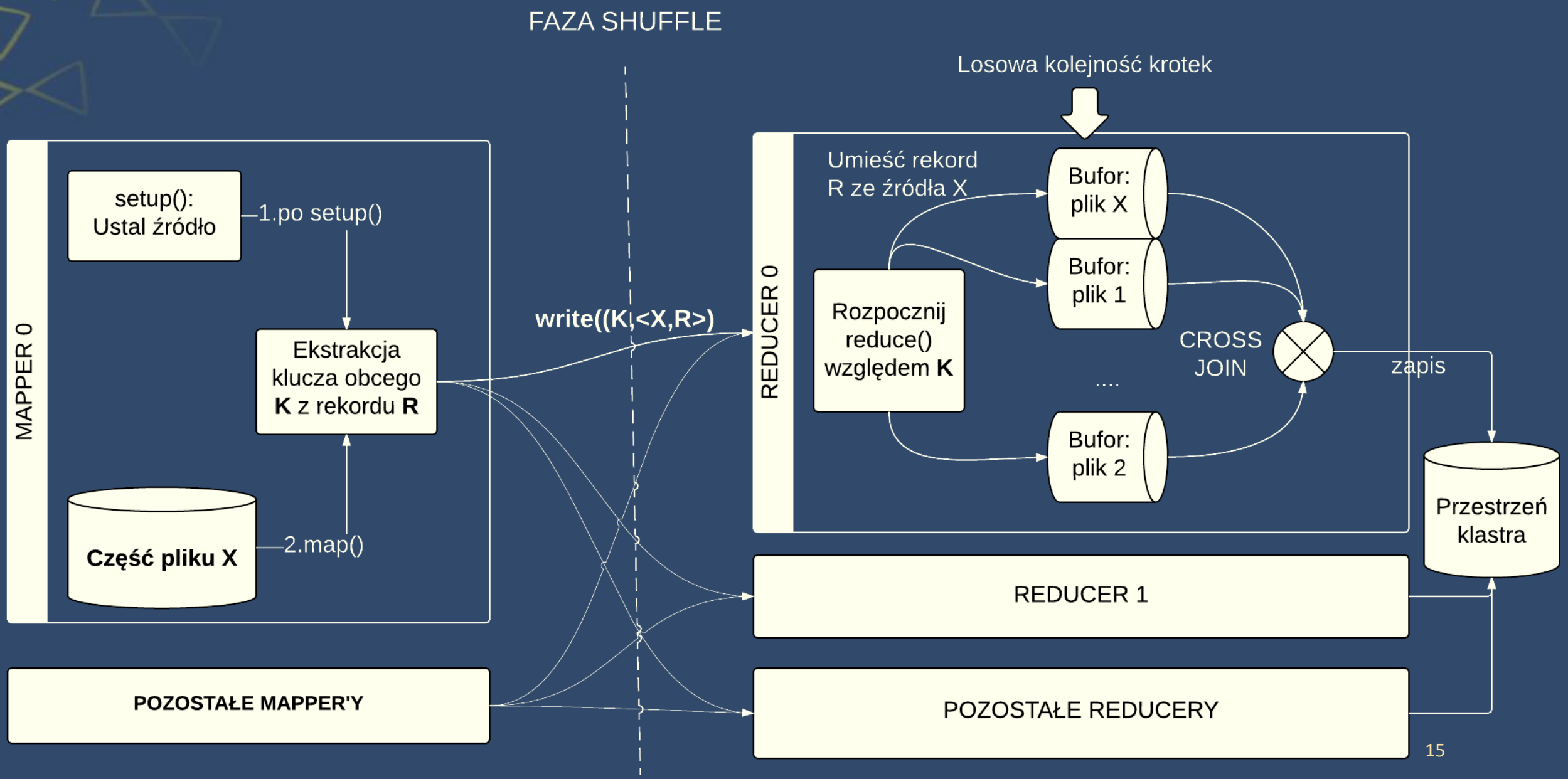
Złączenia

Klasyczne -> Reduce-Side Joins -> Repartition Join

Koncepcja:

- Każdy plik wejściowy (relacje) jest obsługiwany przez osobną implementację Mappera – konfiguracja za pomocą klasy *MultipleInputs*
- Mapper: emituje parę <klucz obcy, <źródło, rekord>>
- Reducer:
  - Buforuje rekordy ze względu na źródło
  - Liczy produkt kartezjański pomiędzy buforami (np. po wywołaniu *hasNext()*)

Idea: Repartition Join

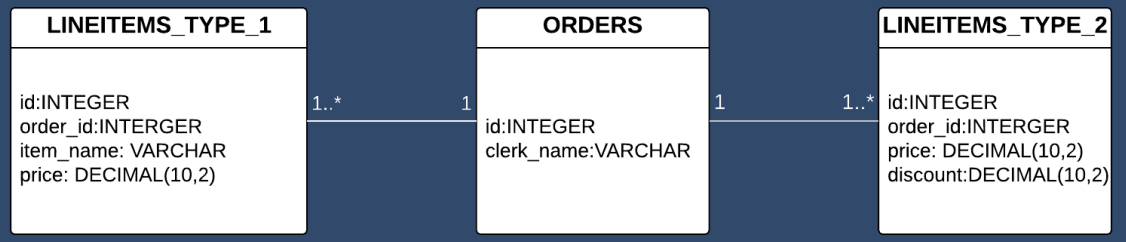


Złączenia

Klasyczne -> Reduce-Side Joins -> Repartition Join

Zalety:

- Wpasowany w model programistyczny
- Wielozłącze dla schematu odwrotnej gwiazdy



Wady:

- Tylko równozłączenie
- Złączenie bez filtrowania wymaga przesłania wszystkich danych
- OutOfMemoryException
- Wielozłącze za pomocą sekwencyjnego łączenia Job’ów

Złączenia

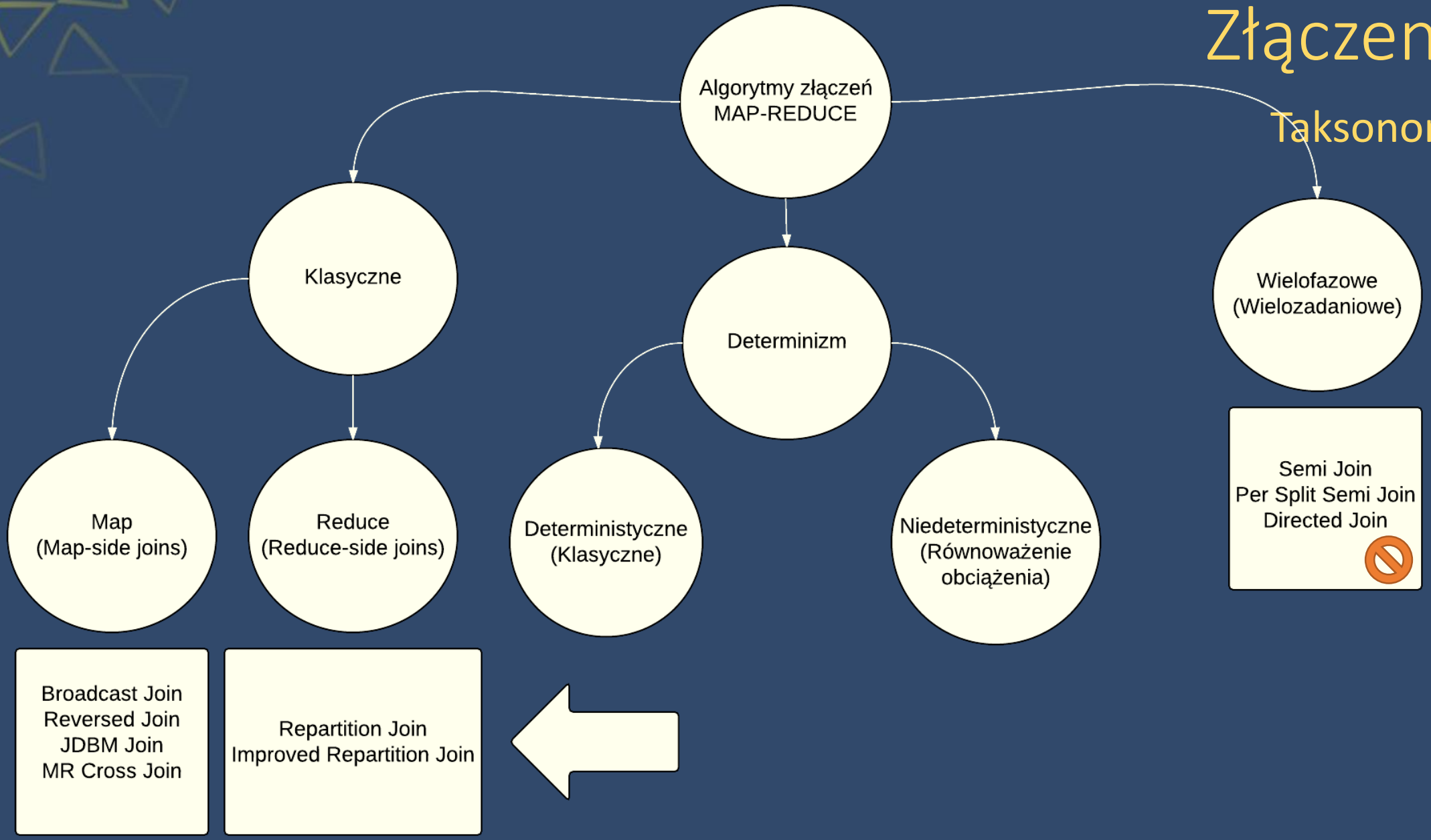
Dystrybucje klucza

Istnieje szereg metod przeciwdziałających nierównym dystrybucjom kluczy obcych. IMHO nieprzydatnych ze względu:

- Map Reduce wykorzystujemy do analizy danych, a nie żeby policzyć złączenie – miejsce dla lokalnej Redukcji
- Filtrowanie rekordów

Złączenia

Taksonomia



Złączenia

Load Balancing Approach: idea

Załóżmy że mamy dwa pliki CSV – „plik_1” i „plik_2”… wypiszmy pierwszy z nich rekord po rekordzie w kolumnie.

Plik 1

1,aaa,2.2
2,bbb,4.3
3,ccc,5.1
4,ddd,1.1
5,aaa,4.2
6,ccc,2.1

Złączenia

Load Balancing Approach: idea

Teraz wypiszmy drugi z nich identycznie.

Plik 1

1,aaa,2.2
2,bbb,4.3
3,ccc,5.1
4,ddd,1.1
5,aaa,4.2
6,ccc,2.1

Plik 2

1,aaa,bbbb,…
1,bbb,aaaa,…
2,ccc,bbbb,…
3,ddd,bbbb,…
3,fff,aaa,…
4,ggg,bbbb,…
4,yyy,bbbb,…
4,aaa,bbbb,…

Złączenia

Load Balancing Approach: idea

Obróćmy – zauważmy że powstaje prostokątna przestrzeń

Plik 2

1,aaa,bbbb,...
1,bbb,aaaa,...
2,ccc,bbbb,...
3,ddd,bbbb,...
3,fff,aaa,...
4,ggg,bbbb,...
4,yyy,bbbb,...
4,aaa,bbbb,...
5,aaa,ccc,...
6,aaa,aaa,...
6,aaa,ccc,...
7,aaa,ccc,...
7,aaa,fff,...
7,aaa,ccc,...

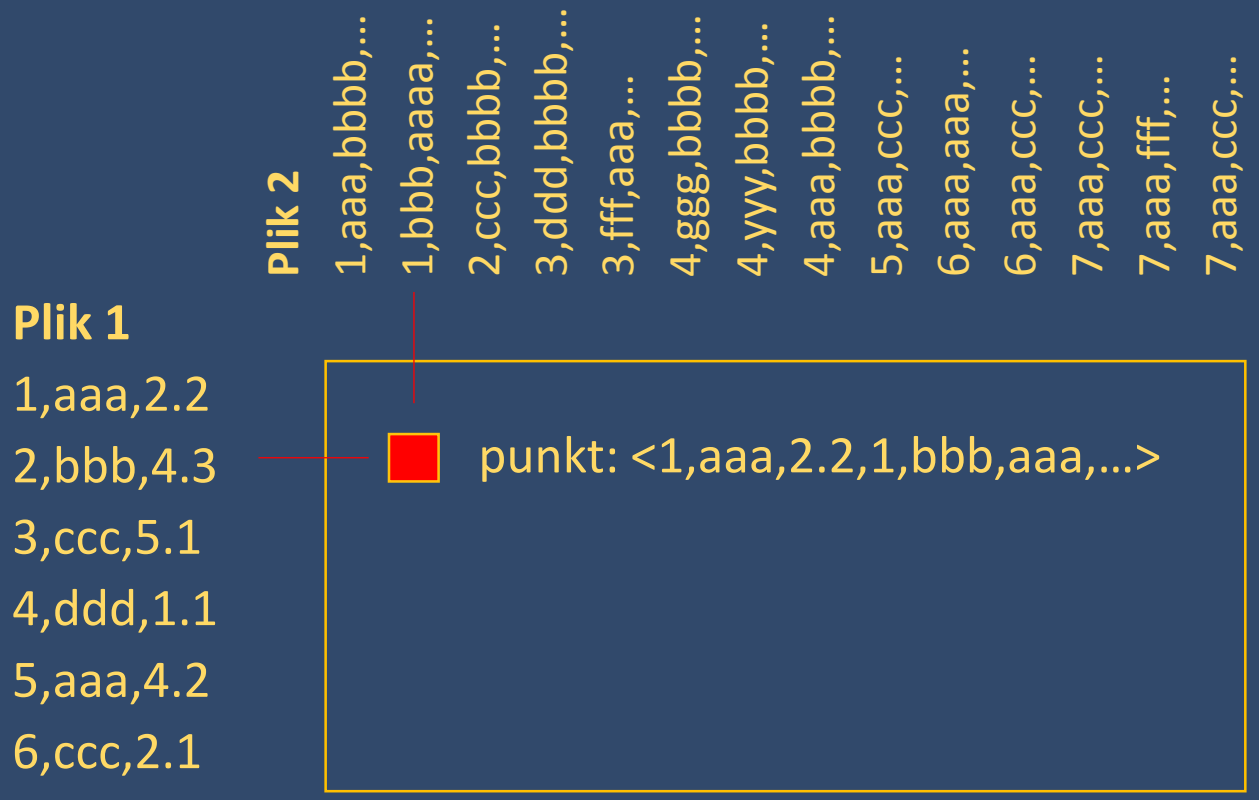
Plik 1

1,aaa,2.2
2,bbb,4.3
3,ccc,5.1
4,ddd,1.1
5,aaa,4.2
6,ccc,2.1

Złączenia

Load Balancing Approach: idea

Obróćmy – zauważmy że powstaje prostokątna przestrzeń



Złączenia

Load Balancing Approach: idea

Wybierzmy pewną liczbę np. 6. Podzielmy przestrzeń na taką liczbę prostokątów i ponumerujmy powstałe prostokąty (od 0)

Plik 2

1,aaa,bbbb,...
1,bbb,aaaa,...
2,ccc,bbbb,...
3,ddd,bbbb,...
3,fff,aaa,...
4,ggg,bbbb,...
4,yyy,bbbb,...
4,aaa,bbbb,...
5,aaa,ccc,...
6,aaa,aaa,...
6,aaa,ccc,...
7,aaa,ccc,...
7,aaa,fff,...
7,aaa,ccc,...

Plik 1

1,aaa,2.2
2,bbb,4.3
3,ccc,5.1
4,ddd,1.1
5,aaa,4.2
6,ccc,2.1

0	1	2
3	4	5

Złączenia

Load Balancing Approach: idea

Niech prostokąty oznaczają dane które „wpadają” do Reduktora o podanym identyfikatorze. Zbiór wszystkich Reducerów umożliwia policzenie Cross Join’a. Przy takim podziale każdy Reduktor wykonuje podobną liczbę iteracji/przetwarza podobną liczbę krotek. Pytanie jak ustalić miejsce podziału.

Plik 2

1,aaa,bbbb,...
1,bbb,aaaa,...
2,ccc,bbbb,...
3,ddd,bbbb,...
3,fff,aaa,...
4,ggg,bbbb,...
4,yyy,bbbb,...
4,aaa,bbbb,...
5,aaa,ccc,...
6,aaa,aaa,...
6,aaa,ccc,...
7,aaa,ccc,...
7,aaa,fff,...
7,aaa,ccc,...

Plik 1

1,aaa,2.2
2,bbb,4.3
3,ccc,5.1
4,ddd,1.1
5,aaa,4.2
6,ccc,2.1

0	1	2
3	4	5

Złączenia

Load Balancing Approach: idea

W tym celu usuńmy z naszego diagramu abstrakcję plików.

0	1	2
3	4	5

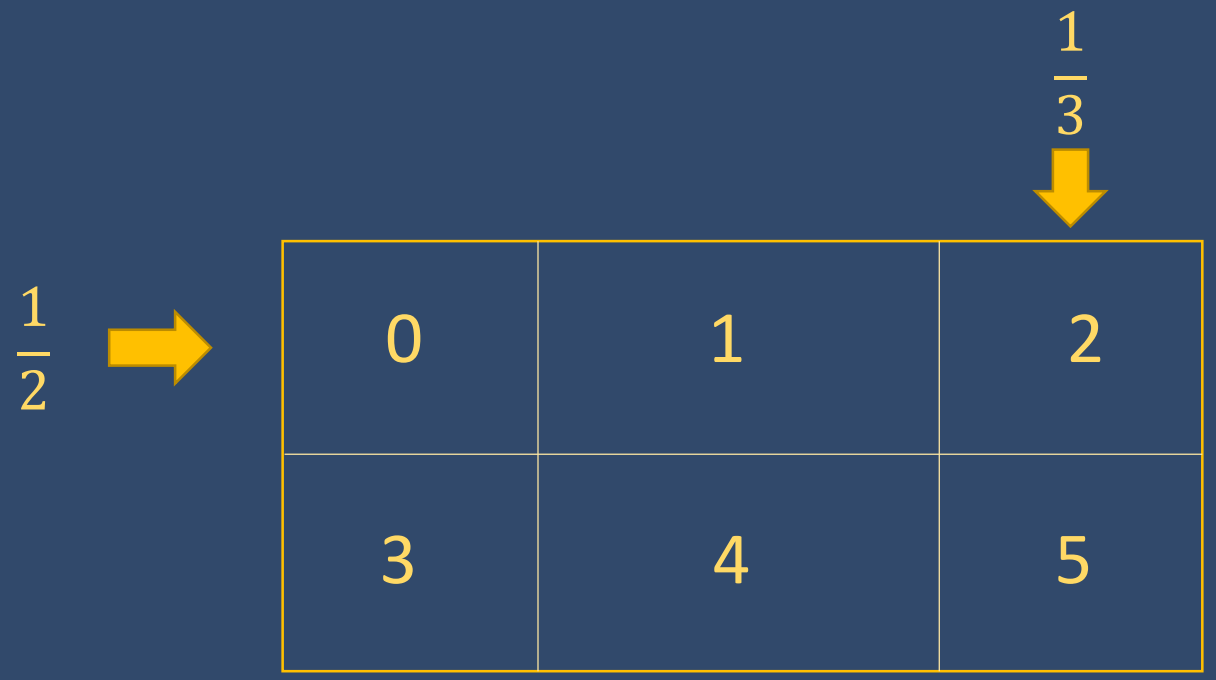
Złączenia

Load Balancing Approach: idea

Przyporządkujmy równomiernie prawdopodobieństwo umieszczenia rekordu w Reduktorach kolumny/wiersza :

- kolumny mają prawdopodobieństwo $\frac{1}{3}$
- wiersze mają prawdopodobieństwo $\frac{1}{2}$

Każda komórka (Reduktor) tym samym otrzyma około $\frac{1}{6}$ wszystkich danych



Złączenia

Load Balancing Approach: spostrzeżenia

- Co potem?
  - Literatura mówi: W reduce() „użyj ulubionego algorytmu do Cross Join'a"
  - Każdy rekord z Cross Joina filtrujemy
- Podejście daje się uogólnić na przestrzeń o dowolnej liczbie wymiarów
- W przykładzie podział był 2 wiersze i 3 kolumny - [2,3] – co oznacza że rekordy z pliku 1 będą 2-krotnie zreplikowane, a z pliku 2 – 3-krotnie
- Shuffle jako faza najbardziej czasochłonna – a my wysyłamy więcej danych
- Filtrowanie Cross Join'a - żałosna selektywność selekcji
- Podział wpływa bezpośrednio na efektywność rozwiązania – da się mierzyć

Złączenia

Load Balancing Approach: optymalizacja

Dla zadanej liczby Reduktorów R trzeba znaleźć optymalny podział plików

Zakładając dwa pliki o rozmiarach F_1, F_2 szukamy takich liczb całkowitych p_1 i p_2 które minimalizują koszt fazy Shuffle:

$c(p_1, p_2) = R\left(\frac{F_1}{p_1} + \frac{F_2}{p_2}\right)$, gdzie $R = p_1 p_2$

Po uogólnieniu na N plików:

$c(p_1, \dots, p_N) = R\sum_i^N \frac{F_i}{p_i}$, przy ograniczeniu $R = \prod_i^N p_i$

Graficznie: kwadraty, sześciany, hipersześciany itd.

Model programistyczny

Load Balancing Approach: Mój stosunek do modelu Map Reduce

1. „W rozważaniach teoretycznych przyjmuje się następujący model”
2. Programiści łamiący model
   - Podejście burzy determinizm
   - Programista przepisuje Partitioner i Mapper - ogólnie rzecz biorąc - odseparowano programistę od systemu, a ten przepisuje jego funkcjonalność
   - Grupowanie, a w zasadzie dzielenie danych ze względu na sztuczny klucz – liczba wywołań reduce() jest równa liczbie Reduce’rów
   - Niestety to jedyny sposób żeby wykonać pełnoprawne Theta-złącze w modelu Map-Reduce (ale nie w jego rozszerzeniach)

Czas na Plot Twist

Antywzorce

Złączenia

Antywzorce

Applications not using a higher-level interface such as Pig unless really necessary

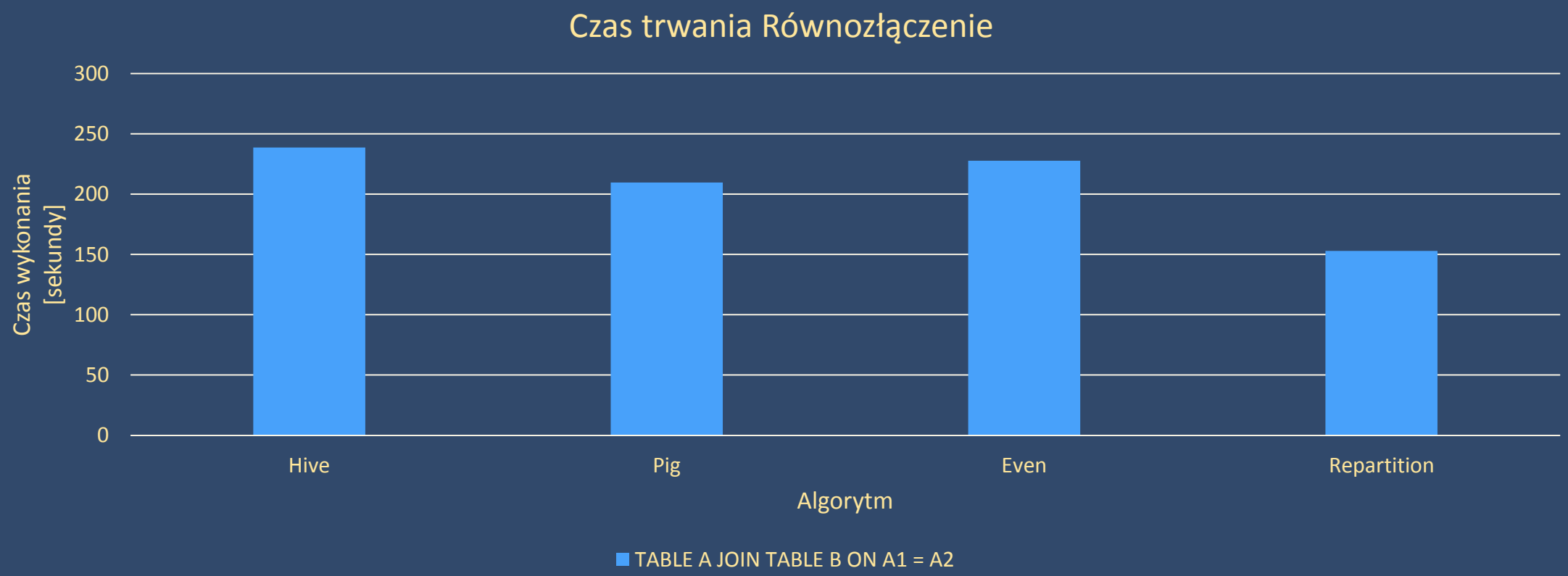
Źródło: https://developer.yahoo.com/blogs/hadoop/apache-hadoop-best-practices-anti-patterns-465.html

Wydajność

- Wydajność rozumiana jako czas utylizacji klastra – czas wykonania Job'a
- Test:
  - Klaster 5 węzłów (chmura Amazona)
  - Złączenie dwóch tabel, relacja 1-N
  - Zbiór danych TPC-H – 1 GB
  - Obfuskacja danych (kolumn)
  - Dwa zapytania
    - Równozłączenie: TABLEA JOIN TABLEB ON A1 = A2
    - Theta-złączenie: E1 <= date_add(K2, 30) AND E1 >= date_sub(K2, 30)

Wydajność

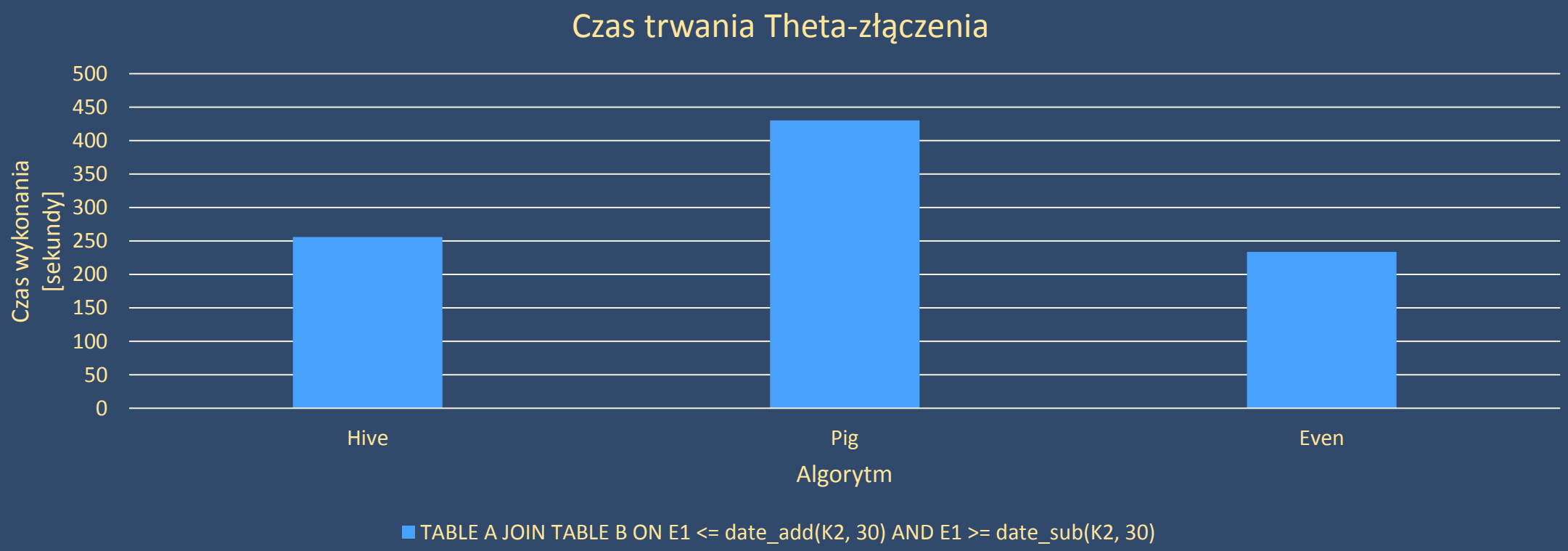
Równozłączenie



* https://cwiki.apache.org/Hive/languagemanual-joins.html
* http://grokbase.com/t/hive/user/123dnszatz/non-equality-joins

Wydajność

Theta-złączenie



* https://cwiki.apache.org/Hive/languagemanual-joins.html
* http://grokbase.com/t/hive/user/123dnszatz/non-equality-joins

Podsumowanie

- Nad każdym klastrem powinien wisieć Framework (co najmniej jeden)
- Kodzimy w ostateczności
- Złączenia nierównościowe w ogólnej postaci **nie** są możliwe do wyrażenia za pomocą grupowania – agregacji
- Systemy Map Reduce NIE są kompletne obliczeniowo i NIE powinny być używane jak *złoty młotek*
- Istnieją inne modele programistyczne np. PACT, który rozszerza liczbę możliwych faz

Materiały

- Hadoop The Definitive Guide 4th edition, Tom White
- Hive reference: link
- MapReduce Design Patterns, Donald Miner & Adam Shook
- MapR Academy: link
- Hortonworks VM: link
- Join algorithms in Map/Reduce, Maciej Penar, praca magisterska, Politechnika Wrocławska

Dziękuję za uwagę

Pytania

Slajdy dodatkowe

Hive

Złączenia

Frameworki

Frameworki na licencji Apache:

- **Hive**
- Pig
- Drill
- Tajo
- Spark
- Inne…

Złączenia

Frameworki

Języki:

- **Hive - HiveQL**
- Pig – PigLatin <- proceduralny
- Drill - SQL
- Tajo – TSQL (TajoSQL)
- Spark – Spark SQL

Nie są mi znane języki operujące w ramach Hadoopa które potrafią wykonać złączenie nierównościowe

Złączenia

HiveQL - Tabele zewnętrzne – jak SQL

```
CREATE EXTERNAL TABLE `default.TABLEA`(
        `A1` int ,
        `B1` int ,
        `C1` string ,
        `D1` double,
        `E1` string,
        `F1` string,
        `G1` string,
        `H1` string,
        `I1` string)
ROW FORMAT DELIMITED FIELDS TERMINATED BY "|"
STORED AS TEXTFILE
LOCATION '/user/peczwy/data/1n/VOLUME/orders';
```

Złączenia

HiveQL

```
SELECT page_views.*
FROM page_views
WHERE page_views.date >= '2008-03-01'
        AND page_views.date <= '2008-03-31,
```

https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Select

Złączenia

HiveQL – rodzaje

Złączenia:

1. x JOIN y ON c
2. x {LEFT|RIGHT|FULL} [OUTER] JOIN y ON c
3. x LEFT SEMI JOIN y ON c
4. x CROSS JOIN y

https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Joins

Złączenia

HiveQL – Złączenia nierównościowe

```
SELECT a.val, b.val, c.val
FROM a CROSS JOIN b
WHERE ABS(a.key - b.key) < 5;
```

Kontrowersyjna wydajność, ale działa

https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Select

Złączenia

HiveQL - Annotacje

STREAMTABLE – strumieniowanie podanej tabeli (domyślnie najbardziej z *prawej* strony:

SELECT */*+ STREAMTABLE(b) */* a.val, b.val, **c**.val

FROM a **JOIN** b **ON** (a.**key** = b.key1) **JOIN c ON** (**c**.**key** = b.key1)

MAPJOIN – zaproszenie do złączenia po stronie Mappera

SELECT /*+ MAPJOIN(b) */ a.key, a.value

FROM a **JOIN** b **ON** a.key = b.key

https://cwiki.apache.org/confluence/display/Hive/LanguageManual+JoinOptimization

Hive

"Hive does not support join conditions that are not equality conditions as it is **very difficult** to express such conditions as a map/reduce job.„*

* https://cwiki.apache.org/Hive/languagemanual-joins.html
* http://grokbase.com/t/hive/user/123dnszatz/non-equality-joins

Dziękuję za uwagę

Pytania