Hadoop - wprowadzenie

Łukasz Król

Hadoop - wprowadzenie

obszar działalności:

- hurtownie danych
- programowanie ETL
- Business Intelligence
- Big Data

- programowanie obliczeń rozproszonych
- uczenie maszynowe
- statystyka

narzędzia:

- SQL Server, Oracle
- Cloudera (Hadoop)
- SAS
- R, Scala, Java
- Akka
- Informatica PowerCenter
- IBM Cognos

Hadoop - wprowadzenie

Hadoop - wprowadzenie

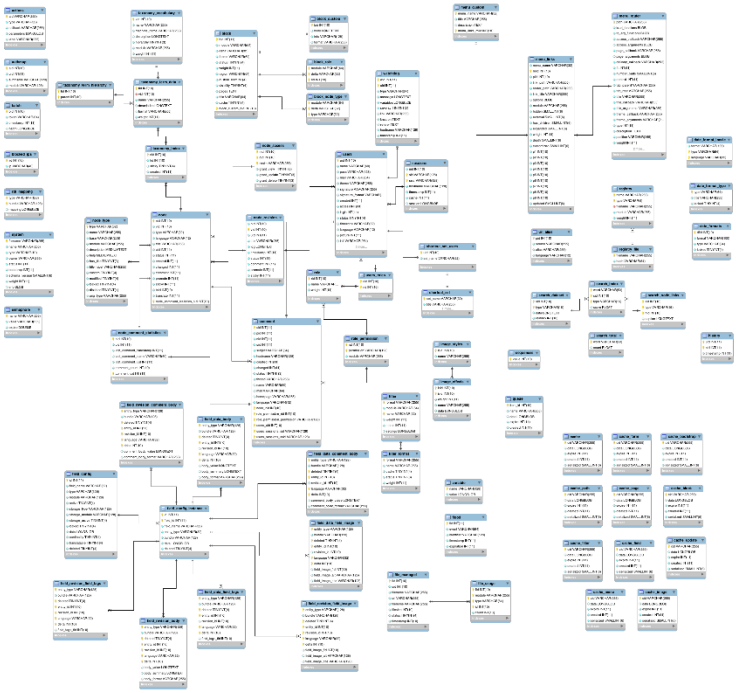
Cel: zaawansowana analiza danych od małych do średnich rozmiarów (do kilkuset, praktycznie kilkudziesięciu GB) na jednej maszynie.

cel Hadoop 1.0: **jakakolwiek** analiza danych o rozmiarach rzędu kilkuset TB (problemy skali Google, Facebook).

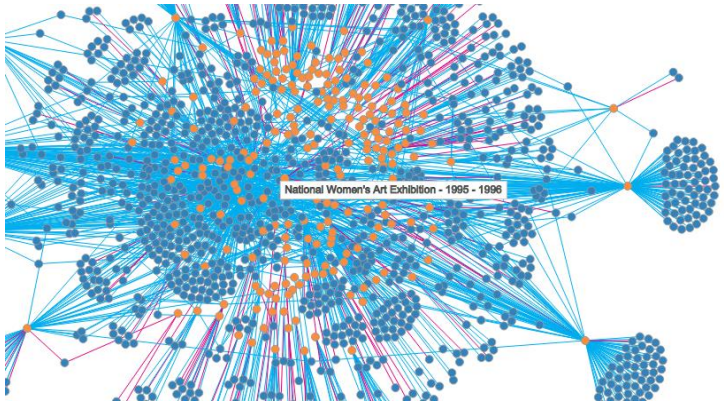
techniczny cel Hadoop 2.0: rodzaj „systemu operacyjnego” dla klastra kilkunastu do kilkuset **tanich** maszyn pozwalającego na analizę off-linową (batch) jak i on-linową (datastream processing) różnorodnych danych.

Hadoop - mity

SQL (przeszłość):



NoSQL (przyszłość):

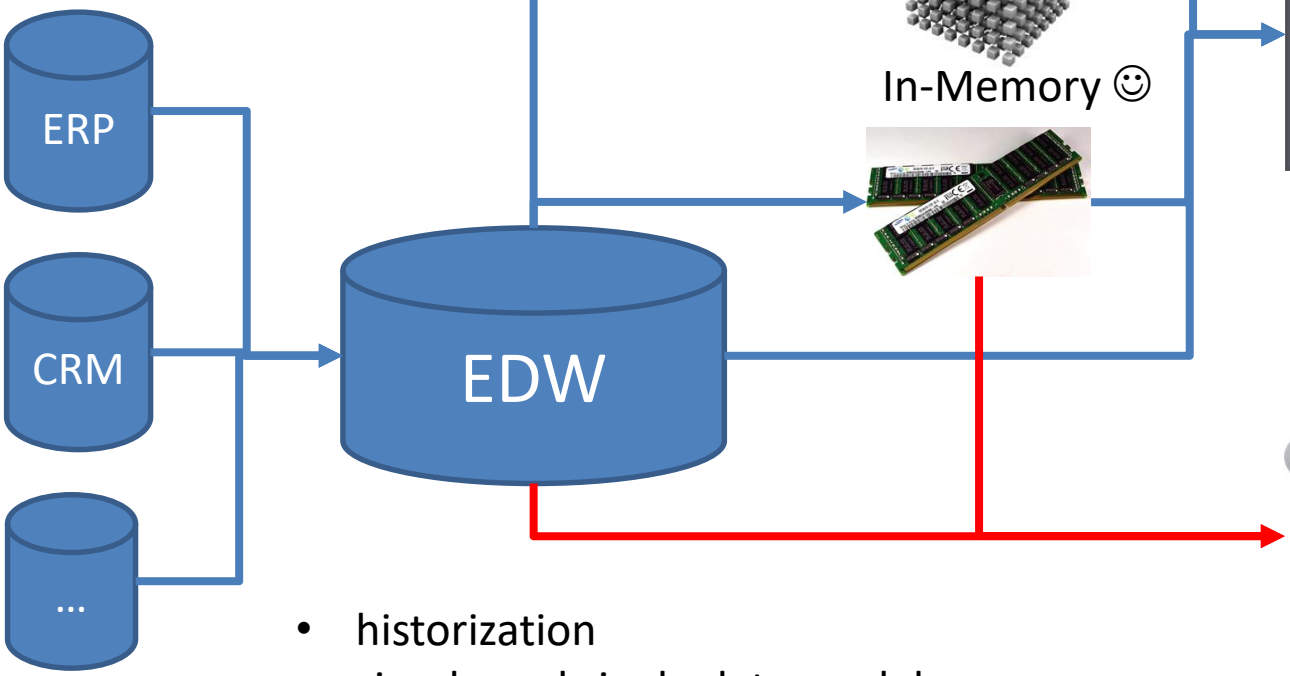


Hadoop - mity

mit	wyjaśnienie
Hadoop to narzędzie Zaawansowanej Analizy Danych.	Hadoop to platforma rozproszonego przetwarzania danych.
Hadoop pozwala na rozproszenie dowolnych obliczeń.	Hadoop wymusza na programiście stosowanie określonych abstrakcji, często ograniczających ekspresyjność.
Hadoop to najwydajniejsza platforma przetwarzania i analizy danych.	Hadoop uzyskuje przewagę nad systemami scentralizowanymi dopiero przy naprawdę dużych rozmiarach danych.
Hadoop może w swojej obecnej formie zastąpić klasyczne systemy transakcyjne.	Hadoop w założeniu nie jest systemem transakcyjnym.
Hadoop to jedyna droga do wdrożenia Business Intelligence i Zaawansowanej Analityki w firmie.	Hadoop służy rozwiązaniu problemów rozmiaru danych nie występujących w większości projektów.
Hadoop to jednolity produkt łatwy w użyciu i administracji.	Hadoop to zbiór różnorodnych systemów o bardzo zróżnicowanych właściwościach.
Dane nieustrukturalizowane (NoSQL) są łatwiejsze w analizie od ustrukturalizowanych (SQL).	Dane ustrukturalizowane są prostsze w analizie, ale nie każde dane da się ustrukturalizować.

Hadoop a Hurtownia Danych

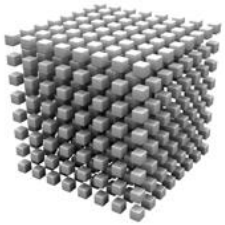
Data Integration
Data Warehousing
ETL (Extract Transform Load)



- historization
- simple and single data model
- aggregation optimized
- no extensive querrying of production systems
- (usually) no transaction processing

Hadoop a Hurtownia Danych

FILE



sas

RDBMS

Microsoft SQL Server 2016

ORACLE

IBM DB2

APPLIANCE

Hadoop a Big Data

Big Data Triple V: Volume, Velocity, Variety

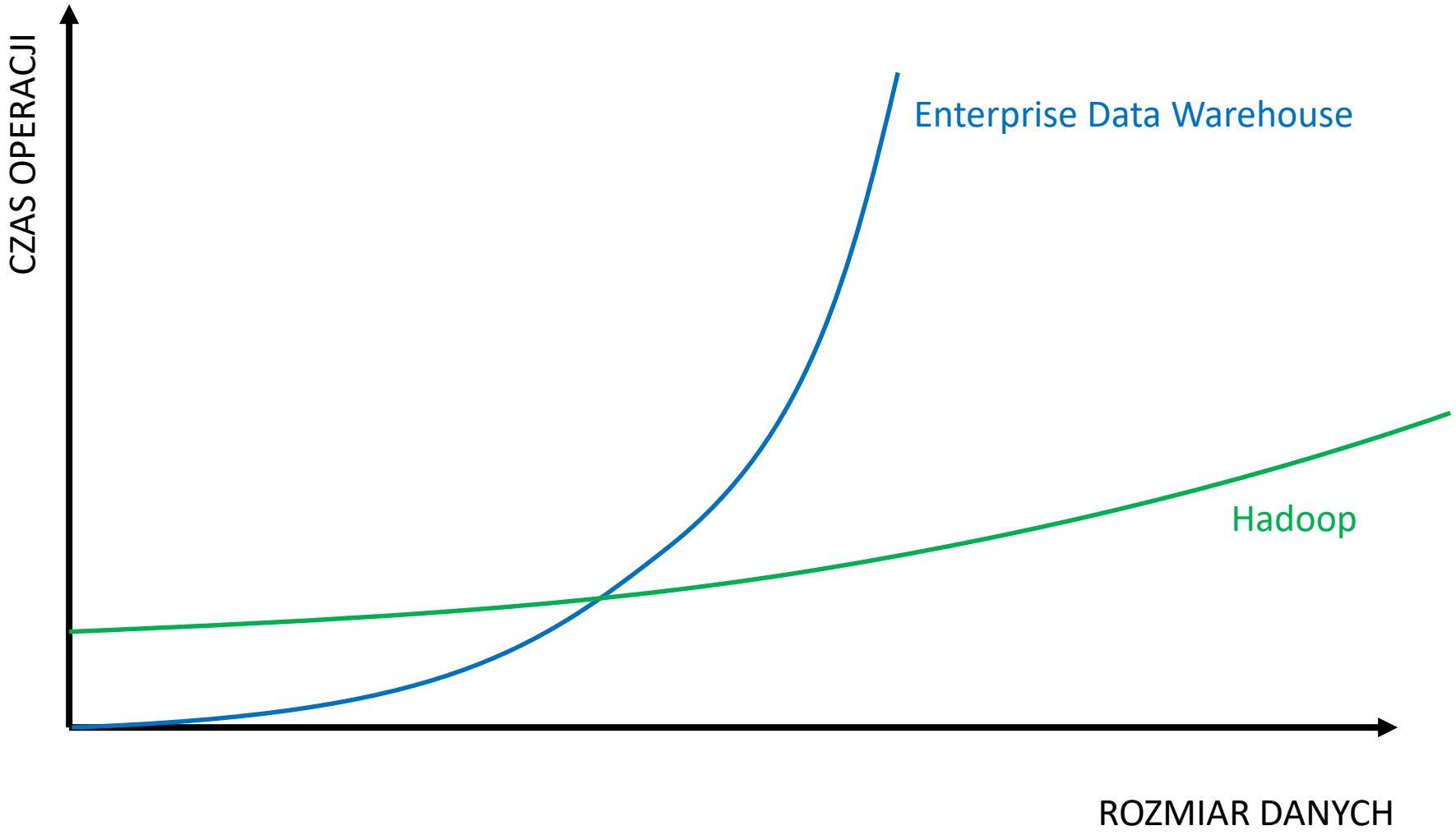
- Danych jest tak dużo, że trzeba je rozproszyć na wiele maszyn.
- Dane napływają tak szybko, że jedna maszyna nie poradzi sobie z ich przetworzeniem.
- Dane są tak różnorodne, że nie da się ich przechowywać w schemacie relacyjnym.

Hadoop a Big Data

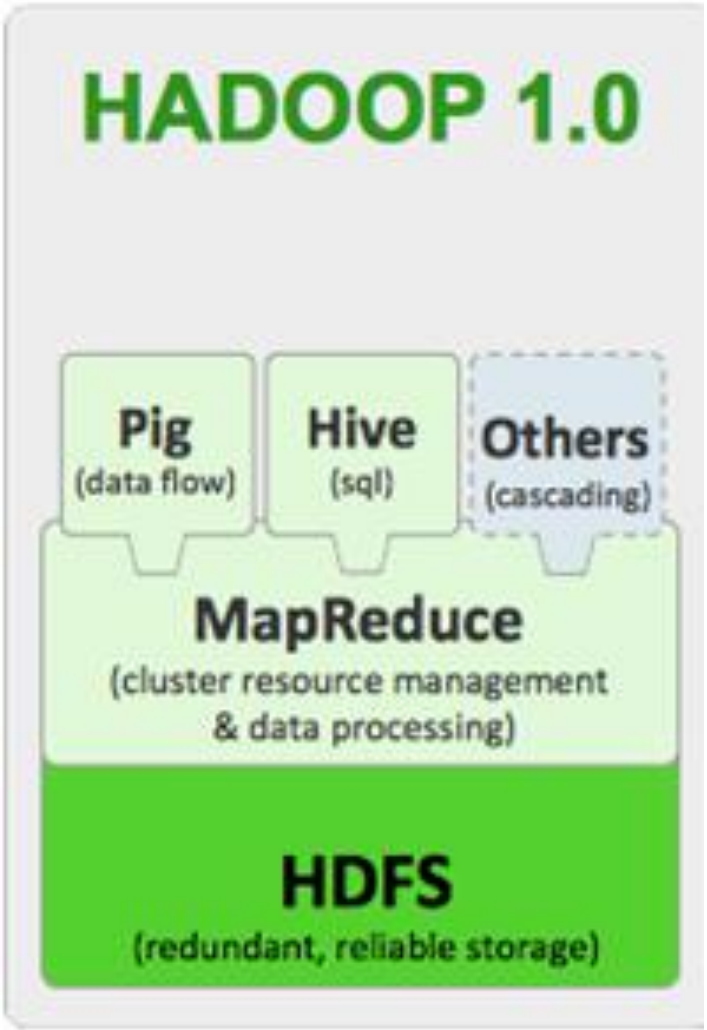
Źródła Big Data

- Social Media
- Internet of Things
- Duże zbiory danych biologicznych

Hadoop vs EDW



Hadoop 1.0



HDFS:

Rozproszony „system plików”.

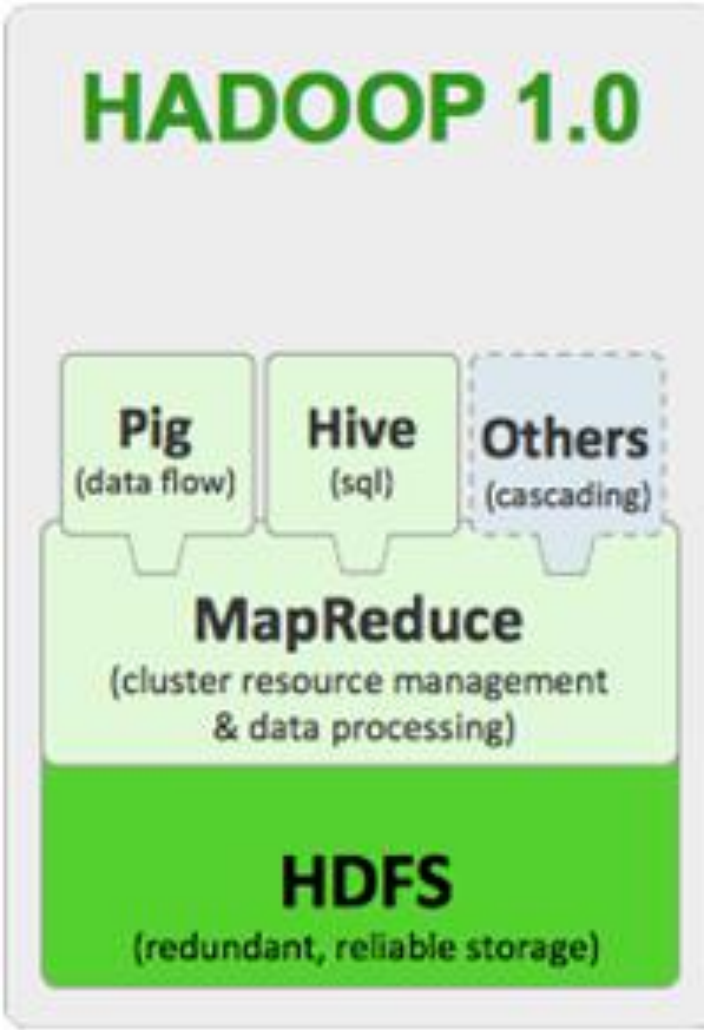
W rzeczywistości zestaw usług w Java (!) korzystających z lokalnych systemów plików węzłów klastra.

Domyślny rozmiar bloku to 64MB (!).

Zapewnia niezawodność przechowywania przez skopiowanie każdego bloku na domyślnie 3 różne węzły (3 razy więcej danych…).

Delikatnie scentralizowany.

Hadoop 1.0



MapReduce:

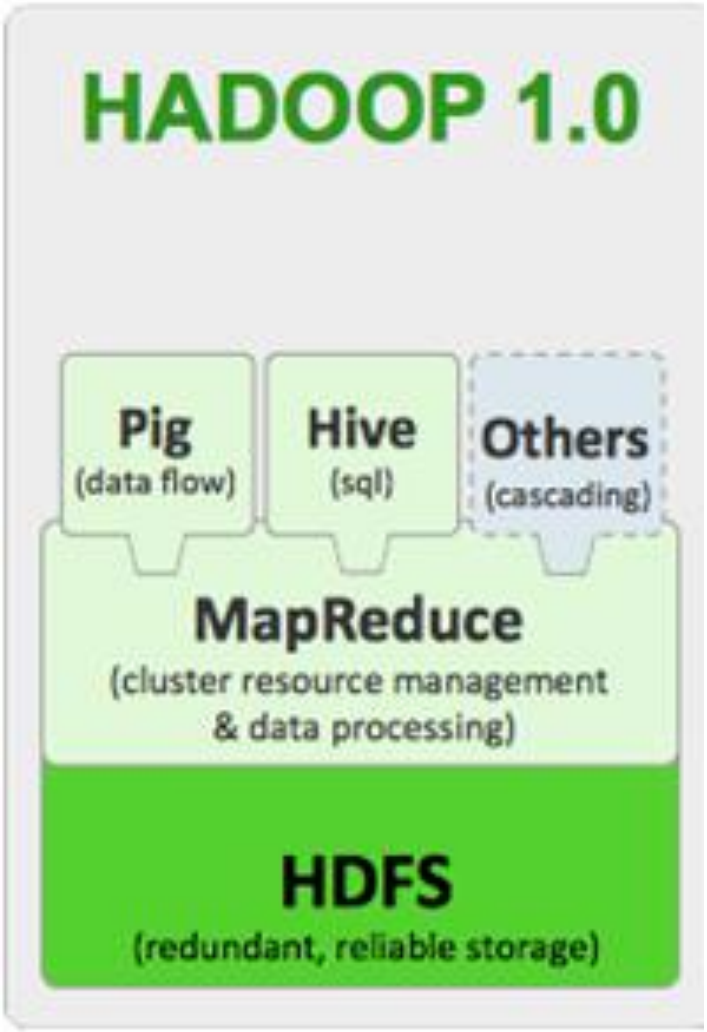
Usługa (Java) wykonywania rozproszonych obliczeń.

Dane i wyniki pobierane i zapisywane z powrotem do HDFS.

Obliczenia definiowane poprzez skompilowanie klasy Java przeciążającej konkretny interfejs i rozesłanie pliku *.jar.

Delikatnie scentralizowany.

Hadoop 1.0

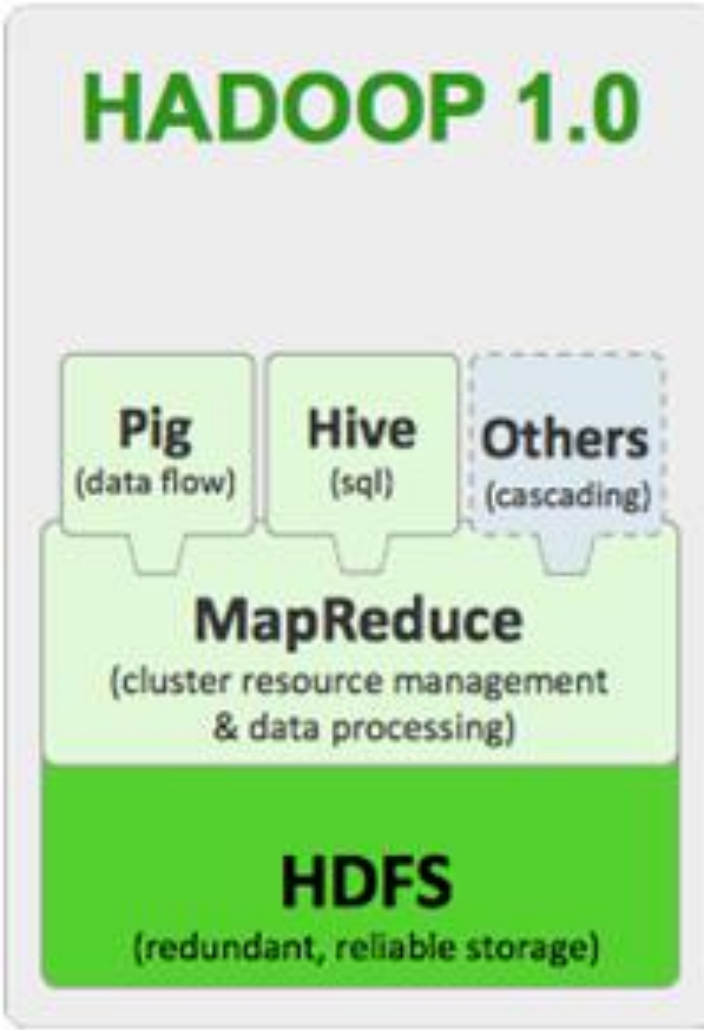


Pig Latin:

Język ułatwiający pisanie procesów przetwarzania danych. Pig Latin jest „kompilowany” do jarów wykonywalnych przez MapReduce.

Wypierany przez Sparka.

Hadoop 1.0



Hive:

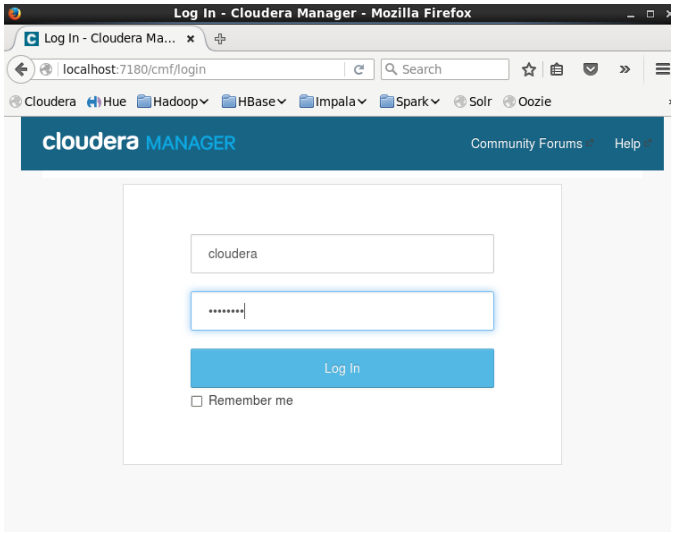
Interfejs SQL do MapReduce.

Hive SQL jest „kompilowany” do jarów wykonywalnych przez MapReduce.

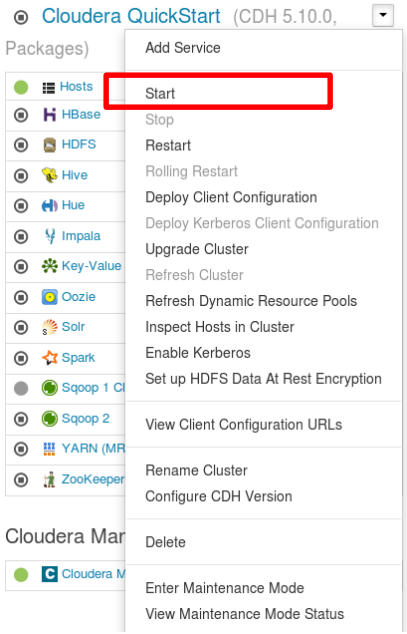
W praktyce oferuje funkcjonalność rozproszonej bazy danych – bez indeksów, transakcji i zaawansowanego zarządzania uprawnieniami.

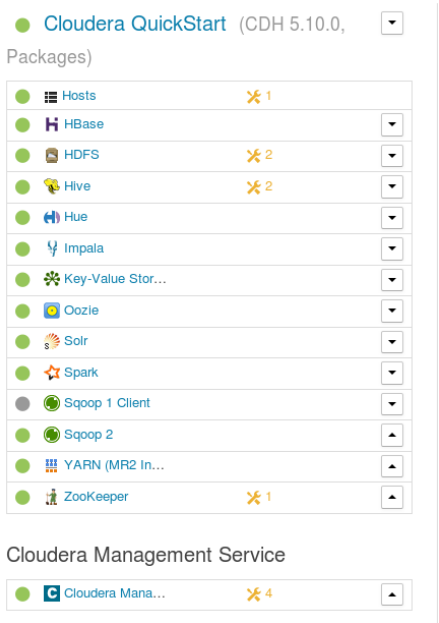
W nowszych projektach wykorzystywany w oderwaniu od MapReduce (Tez).

"podniesienie" środowiska



cloudera/cloudera





HDFS – ćwiczenie (1/3)

```
#uruchomić terminal (np. gnome-terminal)

#sprawdzić aktualny folder lokalny
pwd

#wylistować  zawartość lokalnego folderu
ls –l

#przejść do folderu /home/cloudera/lab/00_data
cd lab/00_data

#wygenerować trochę losowych danych
cat /dev/random > garbage.txt
```

HDFS – ćwiczenie (2/3)

```
#wyświetlić pomoc hadoop fs
hadoop fs –help

#wylistować zawartość głównego folderu w HDFS
hadoop fs –ls /

#wylistować zawartość folderu /user/cloudera
hadoop fs –ls /user/cloudera

#przekopiować plik garbage.txt na HDFS
hadoop fs –copyFromLocal garbage.txt /user/cloudera

#wylistować zawartość /user/cloudera w trybie „human readable”
hadoop fs –ls –h /user/cloudera
```

HDFS – ćwiczenie (3/3)

#obejrzeć zawartość HDFS za pomocą przeglądarkowego interfejsu:
http://quickstart.cloudera:50070/

#obejrzeć zawartość HDFS za pomocą Hue
http://quickstart.cloudera:8888/

Hive – ćwiczenie (1/2)

```
#uruchomić interpreter Hive SQL
hive

#sprawdzić istniejące bazy danych
show databases;

#podłączyć się do lab
use lab;

#sprawdzić istniejące tabele
show tables;

#wypisać na ekran zawartość departments
select * from departments;
```

Hive – ćwiczenie (2/2)

```
#wypisać szczegółowe informacje na temat departments
describe formatted departments;

#odczytać gdzie w HDFS znajduje się tabela
#...

#wyjść z interpretera Hive SQL
exit;

#wypisać zawartość tabeli za pomocą hadoop fs:
hadoop fs –cat [sciezka z Hive]/part-m-00000

#w jaki sposób jest przechowywana „fizycznie” tabela?
```

Hadoop MapReduce

Zadanie – policzyć liczbę wystąpień każdego słowa w pliku.

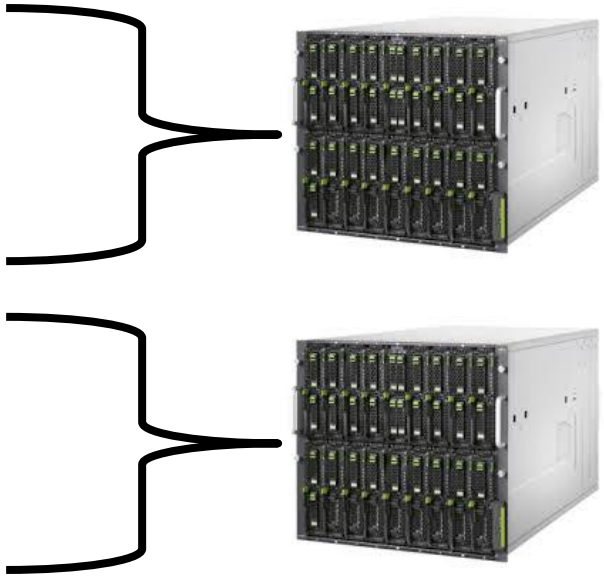
A B B B C C C
#
B B
A A A A
#
B B B C
A #
B
B B A C
# # #

Model scentralizowany – można wykorzystać np. tablicę mieszającą i sekwencyjnie przeskanować plik. Problem – dane mogą nie mieścić się na dysku jednej maszyny.

Hadoop MapReduce

Zadanie – policzyć liczbę wystąpień każdego słowa w pliku.

A B B B C C C
#
B B
A A A A
#
B B B C
A #
B
B B A C
# # #



Model rozproszony – można wykonać obliczenia na części danych i scalić wyniki na jednej maszynie.
Na której maszynie (ip, hostname, użytkownik, hasło, protokół?!).
Programista nie musi znać tych informacji. Ponadto, maszyna wyznaczona do scalania może ulec awarii.
Potrzebny jest wyższy poziom abstrakcji przy definiowaniu obliczeń rozproszonych.

Hadoop MapReduce

1. Operujemy na parach klucz-wartość (K,V).
2. Operacja Map definiuje przekształcenie (K1,V1) w (K2,V2).
3. Operacja Reduce przekształca 2 pary (K,V) w jedną, dla tych samych wartości klucza.
4. Dodatkowa operacja FlatMap to modyfikacja Map pozwalająca zwracać dla jednego wejściowego (K,V) od 0 do n, wyjściowych (K,V), zamiast dokładnie jednego.

Hadoop MapReduce

(A B B B C,NULL)
(#,NULL)
(B B, NULL)
(A A , NULL)
(#, NULL)

(B B B C, NULL)
(A #, NULL)
(B, NULL)
(B B A C, NULL)
(# # #, NULL)

Hadoop MapReduce

(A B B B C,NULL)
(#,NULL)
(B B, NULL)
(A A , NULL)
(#, NULL)

FLAT MAP

(A,1)
(B,3)
(C,1)

(B,2)

(A,2)

(B B B C, NULL)
(A #, NULL)
(B, NULL)
(B B A C, NULL)
(# # #, NULL)

FLAT MAP

(B,3)
(C,1)

(A,1)

(B,1)

(B,2)
(A,1)
(C,1)

Hadoop MapReduce

(A B B B C,NULL)
(#,NULL)
(B B, NULL)
(A A , NULL)
(#, NULL)

FLAT MAP

(A,1)
(B,3)
(C,1)

(B,2)

(A,2)

(B B B C, NULL)
(A #, NULL)
(B, NULL)
(B B A C, NULL)
(# # #, NULL)

FLAT MAP

(B,3)
(C,1)

(A,1)

(B,1)

(B,2)
(A,1)
(C,1)

(A,1)
(A,2)
(A,1)
(A,1)

Hadoop MapReduce

(A B B B C,NULL)
(#,NULL)
(B B, NULL)
(A A , NULL)
(#, NULL)

FLAT MAP

(B B B C, NULL)
(A #, NULL)
(B, NULL)
(B B A C, NULL)
(# # #, NULL)

FLAT MAP

(A,1)
(B,3)
(C,1)
(B,2)
(A,2)
(B,3)
(C,1)
(A,1)
(B,1)
(B,2)
(A,1)
(C,1)

(A,1)
(A,2)
(A,1)
(A,1)
(B,3)
(B,2)
(B,3)
(B,1)
(B,2)

Hadoop MapReduce

(A B B B C,NULL)
(#,NULL)
(B B, NULL)
(A A , NULL)
(#, NULL)

FLAT MAP

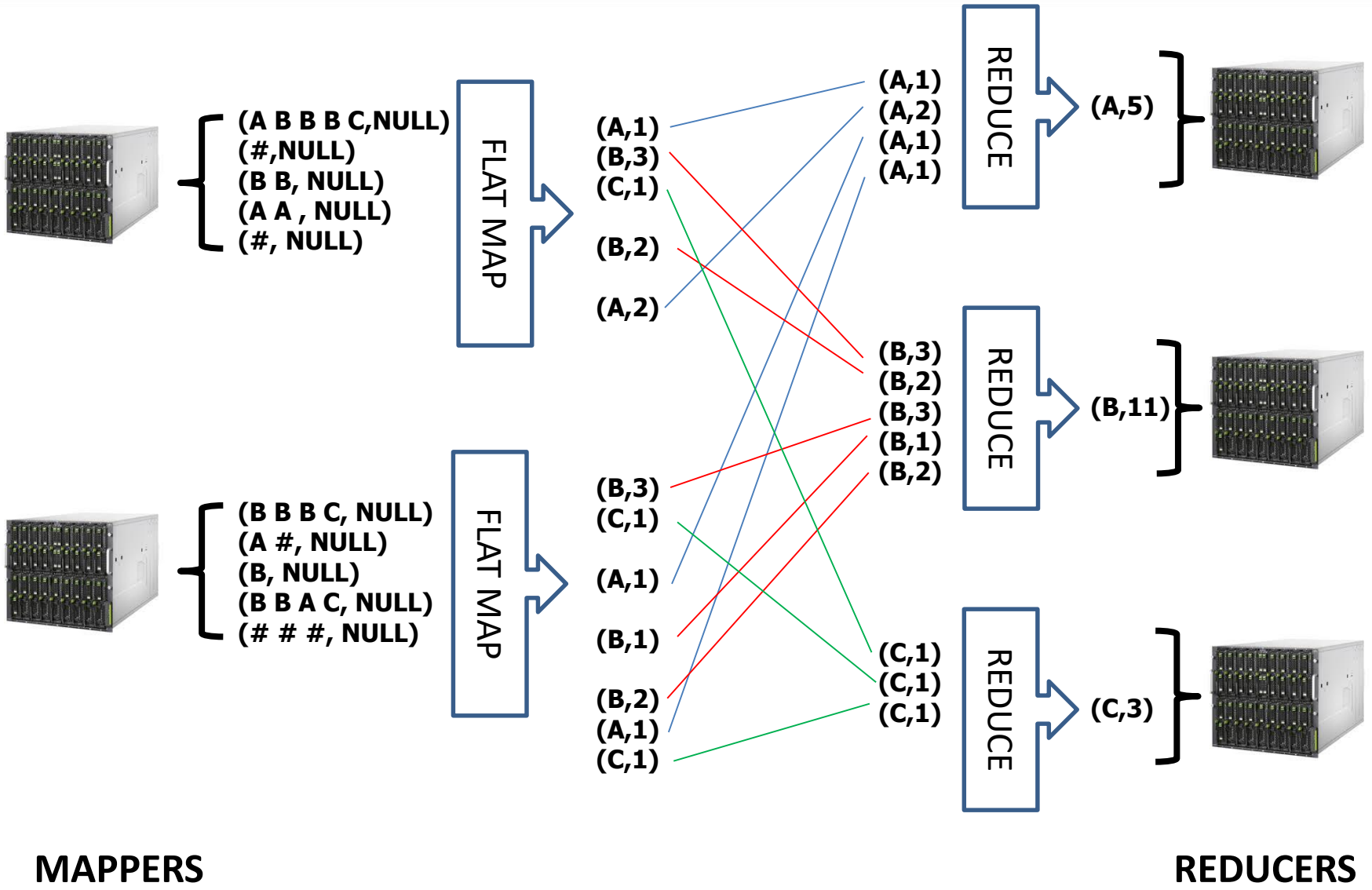
(B B B C, NULL)
(A #, NULL)
(B, NULL)
(B B A C, NULL)
(# # #, NULL)

FLAT MAP

(A,1)
(B,3)
(C,1)
(B,2)
(A,2)
(B,3)
(C,1)
(A,1)
(B,1)
(B,2)
(A,1)
(C,1)

(A,1)
(A,2)
(A,1)
(A,1)
(B,3)
(B,2)
(B,3)
(B,1)
(B,2)
(C,1)
(C,1)
(C,1)

Hadoop MapReduce



MapReduce – ćwiczenie

Zaproponować sposób na wyliczenie średniej w grupach za pomocą MapReduce.

Podpowiedź: Jako wartość można traktować całą strukturę. Można wykonać dwa następujące po sobie mapowania i redukcje.

(A,1)
(B,2)
(A,3)
(A,6)
(B,7)

MapReduce – ćwiczenie

MAP1:
(A,1) -> (A,(1,1))
(B,2) -> (B,(2,1))
(A,3) -> (A,(3,1))
(A,6) -> (A,(6,1))
(B,7) -> (B,(7,1))

REDUCE1:
(A,(1,1))
(A,(3,1))
(A,(6,1)) -> (A,(10,3))

(B,(2,1))
(B,(7,1)) -> (B,(9,2))

MAP2:
(A,(10,3)) -> (A,(3.33))

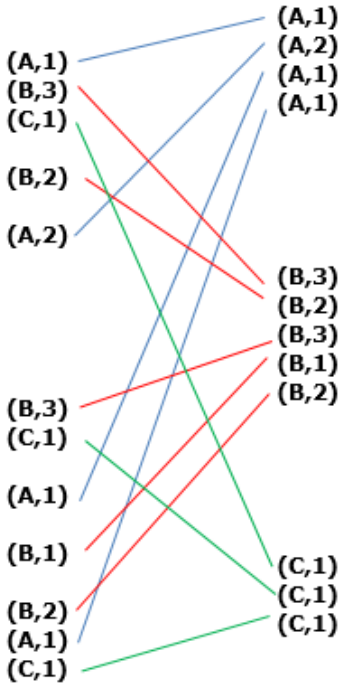
(B,(9,2)) -> (B,(4.5))

REDUCE2:
(A,3.33) -> (A,3.33)

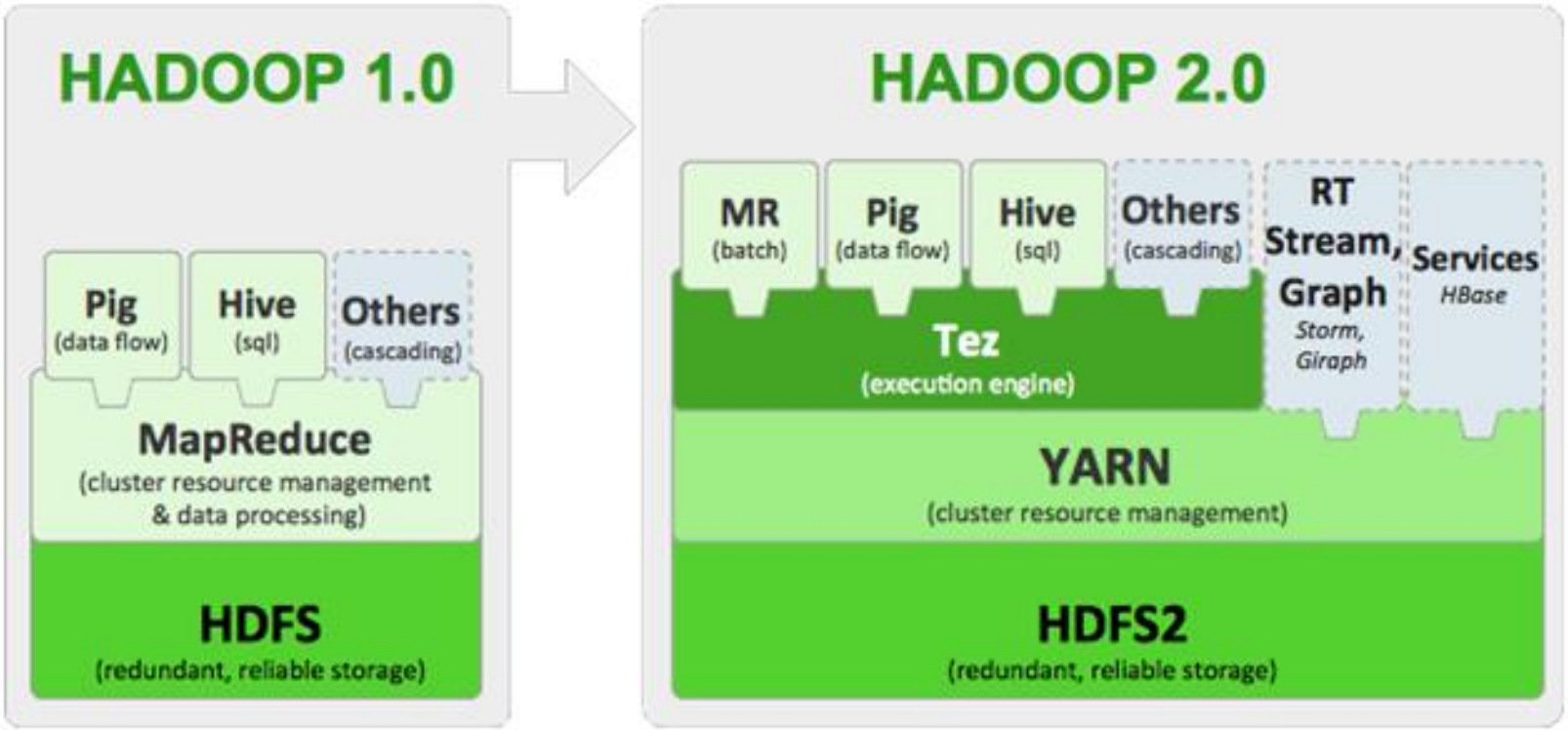
(B,4.5) -> (B,4.5)

MapReduce - krytyka

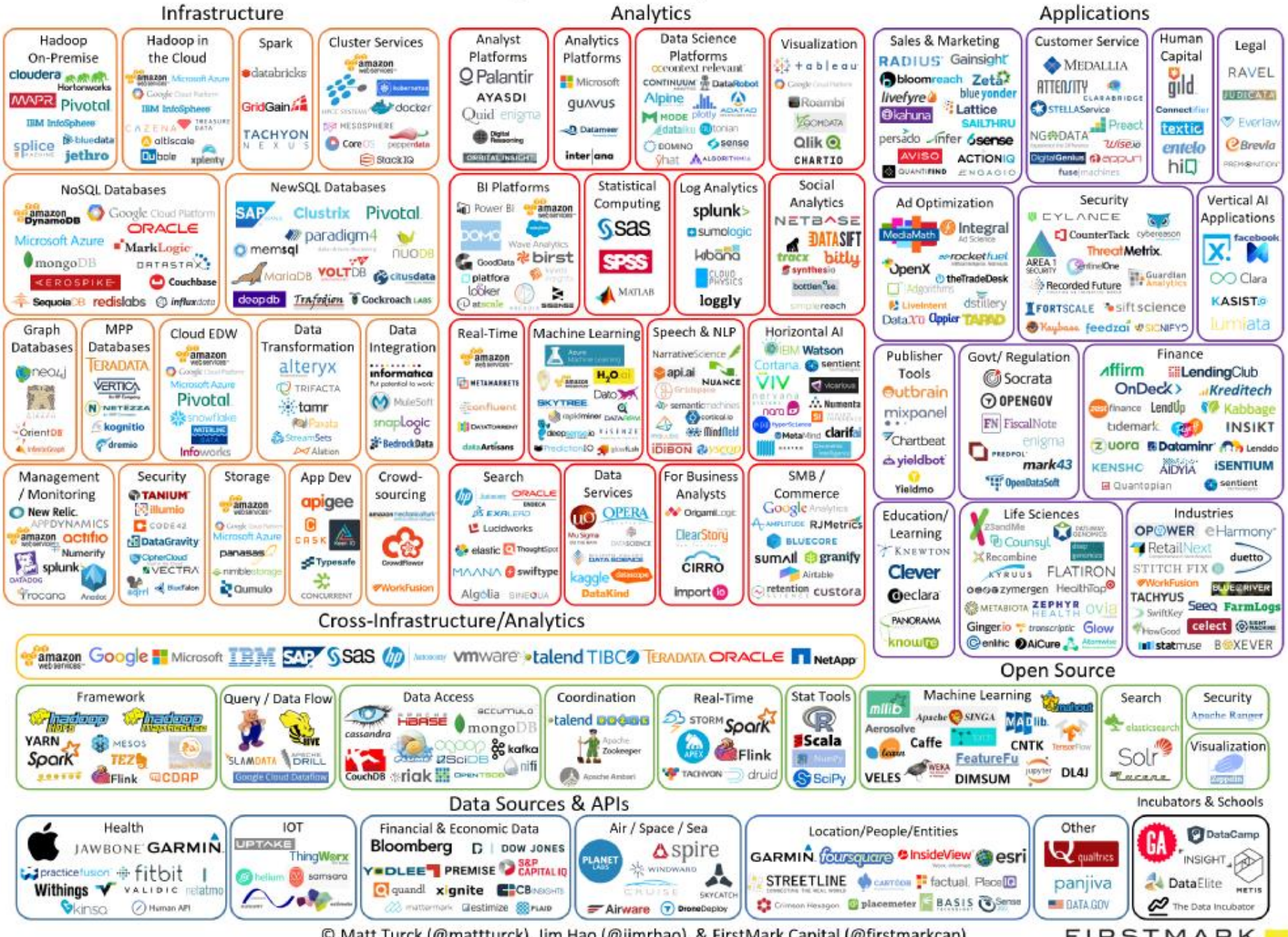
- Konieczność zapisywania danych do HDFS (dyski) co redukcję.
- Mało intuicyjny.
- Niemożliwy do wykorzystania do analizy on-line.
- Shuffle phase!



Hadoop 2.0


HADOOP 1.0
Pig
(data flow)
Hive
(sql)
Others
(cascading)
MapReduce
(cluster resource management
& data processing)
HDFS
(redundant, reliable storage)
HADOOP 2.0
MR
(batch)
Pig
(data flow)
Hive
(sql)
Others
(cascading)
RT
Stream,
Graph
Storm,
Giraph
Services
HBase
Tez
(execution engine)
YARN
(cluster resource management)
HDFS2
(redundant, reliable storage)

Big Data Landscape 2016



Dystrybucje Hadoop

cloudera

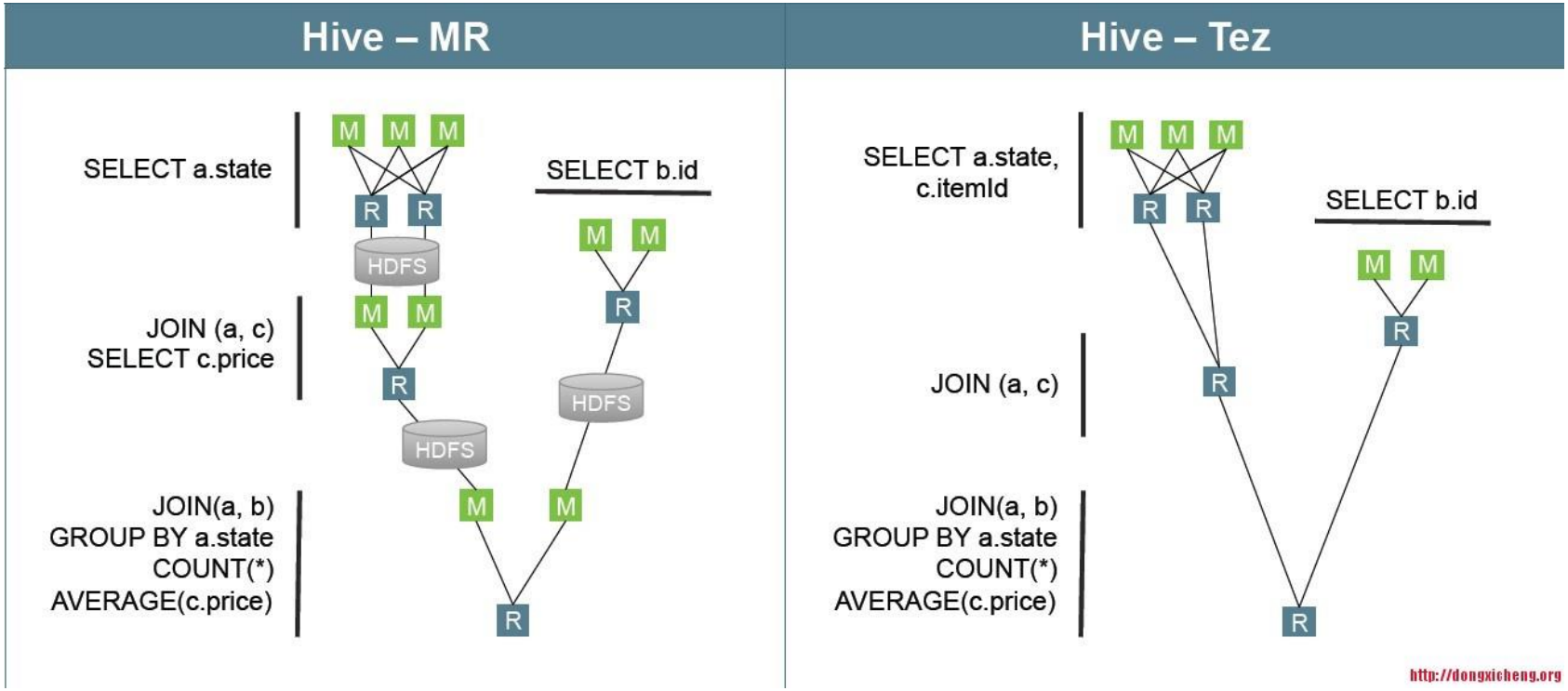
MAPR®

IBM InfoSphere
BigInsights™ for Hadoop
Ten Reasons to Love It.

Hortonworks (nie wspierane w Cloudera)

Hortonworks

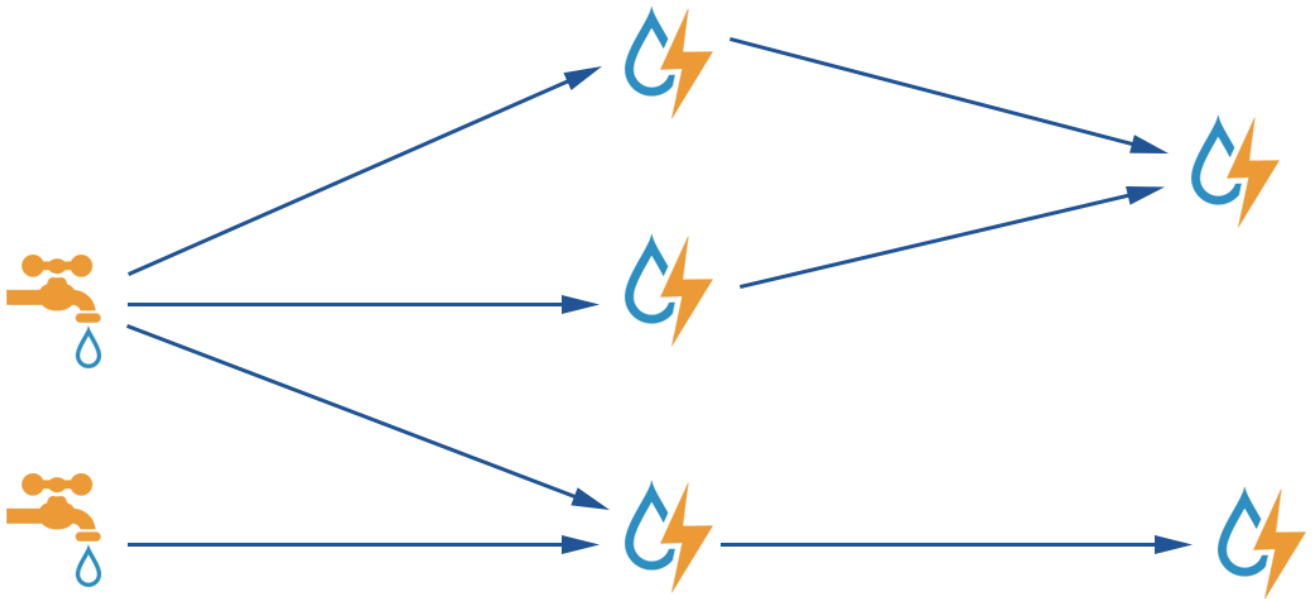
TEZ



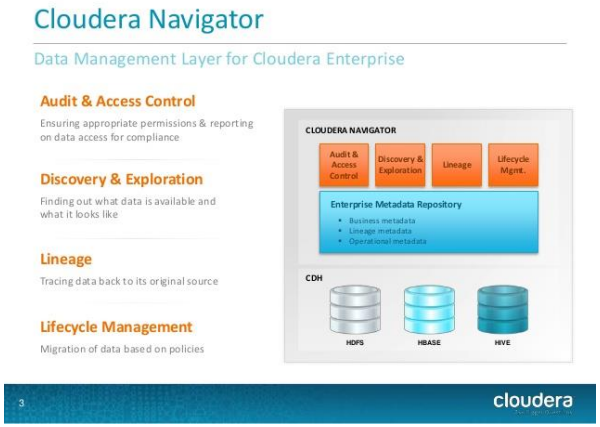
Hortonworks (nie wspierane w Cloudera)

Hortonworks

APACHE STORM™
Distributed • Resilient • Real-time



Cloudera – flagowe projekty



Cloudera – architektura

Service	Master Hosts	Worker Hosts	Utility Hosts
HDFS	Name Node Secondary NameNode	Data Node	

Cloudera – architektura

Service	Master Hosts	Worker Hosts	Utility Hosts
HDFS	NameNode Secondary NameNode	DataNode	
YARN	ResourceManager JobHistory Server	NodeManager	

Cloudera – architektura

Service	Master Hosts	Worker Hosts	Utility Hosts
HDFS	NameNode Secondary NameNode	DataNode	
YARN	ResourceManager JobHistory Server	NodeManager	
Hive	Hive Metastore Server HiveServer2		

Impala

- (Dużo) szybsza alternatywa dla Hive.
- Wykorzystuje HDFS jako system plików.
- Napisana w C++.
- Niezależna od MapReduce i od Tez.
- Zestaw własnych procesów.
- Open Source, ale stworzona przez Clouderę.
- Współdzieli metadane z Hive.
- Poprawnie interpretuje większość Hive SQL.
- Nie obsługuje transakcji.
- Stworzona z myślą o obciążeniu agregacyjnym.
- Wsparcie dla kolumnowych formatów tabel.
- Szerokie wsparcie dla dostępu z narzędzi Business Intelligence poprzez JDBC i ODBC.

Impala - ćwiczenie

```
#uruchomić impala-shell
impala-shell

#przejść do bazy lab i wylistować tabele
use lab;
show tables;

#wypisać pierwsze 10 wierszy tabeli lab_parquet
select * from lab_parquet limit 10

#wypisać 10 najczęściej występujących kombinacji kolumn
ref i alt wraz z licznością
#?

#powtórzyć w Hive
#ile warstw MapReduce zostało zaplanowanych przez Hive?
```

Impala - rozwiązanie

```
select
        ref, alt, count(*) as cnt
from
        lab_parquet
group by
        ref,alt
order by
        cnt desc
limit 10;
```

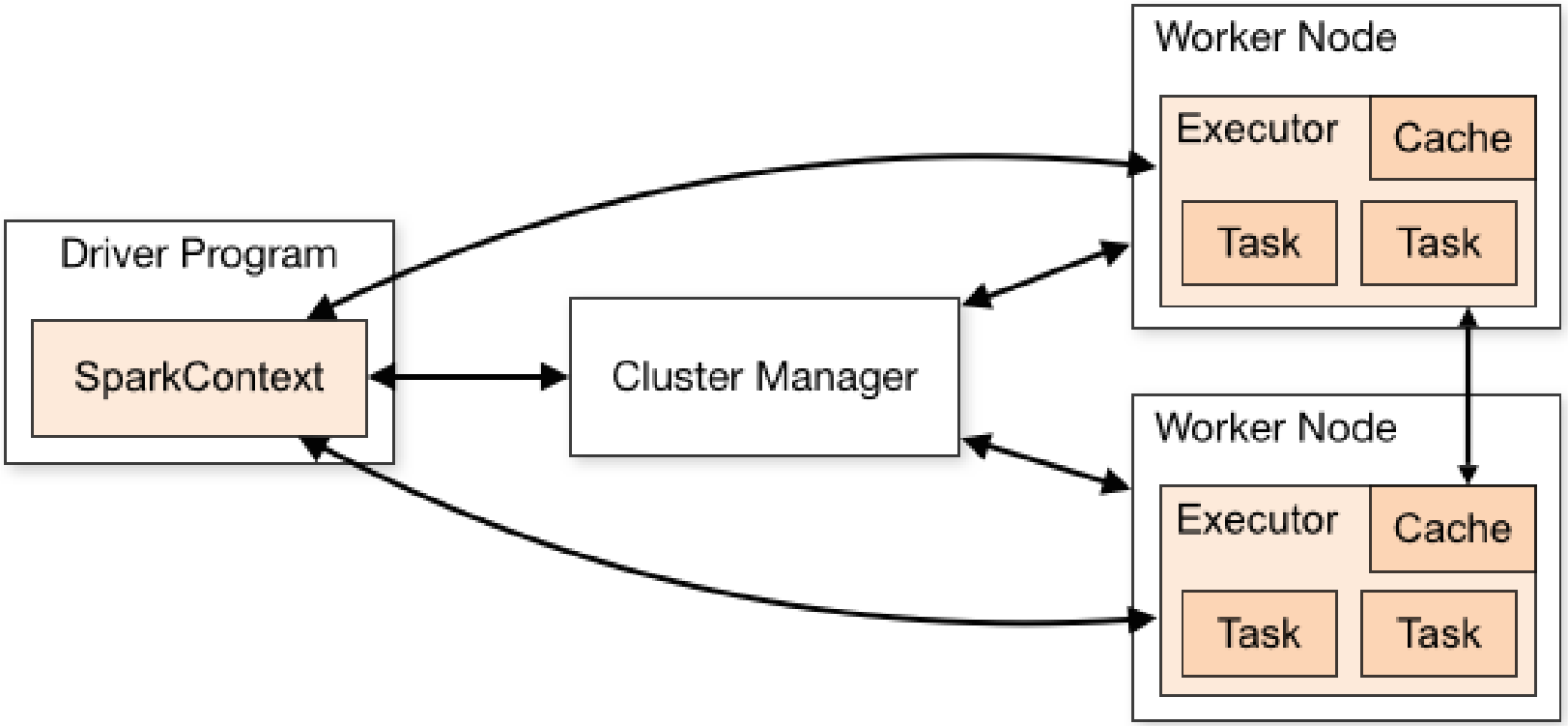
Cloudera – architektura

Service	Master Hosts	Worker Hosts	Utility Hosts
HDFS	NameNode Secondary NameNode	DataNode	
YARN	ResourceManager JobHistory Server	NodeManager	
Hive	Hive Metastore Server HiveServer2		
Impala	Impala Statestore	Impala Deamon	Impala Catalog

Spark

- Alternatywa zarówno dla klasycznego MapReduce, jak i dla Tez.
- Nowa abstrakcja – Resilient Distributed Dataset (RDD)
- Intensywne wykorzystanie pamięci operacyjnej węzłów.
- Szerszy zbiór operacji niż MapReduce.
- Nie ma własnych stałych usług – zależność od YARN.
- Umożliwia pisanie zadań w Java, Scala, Python.
- Koncepcja partycji.
- Niezawodność poprzez możliwość odtworzenia historii zdefiniowanej operacji.
- W dużej mierze niezależny od HDFS i YARN.

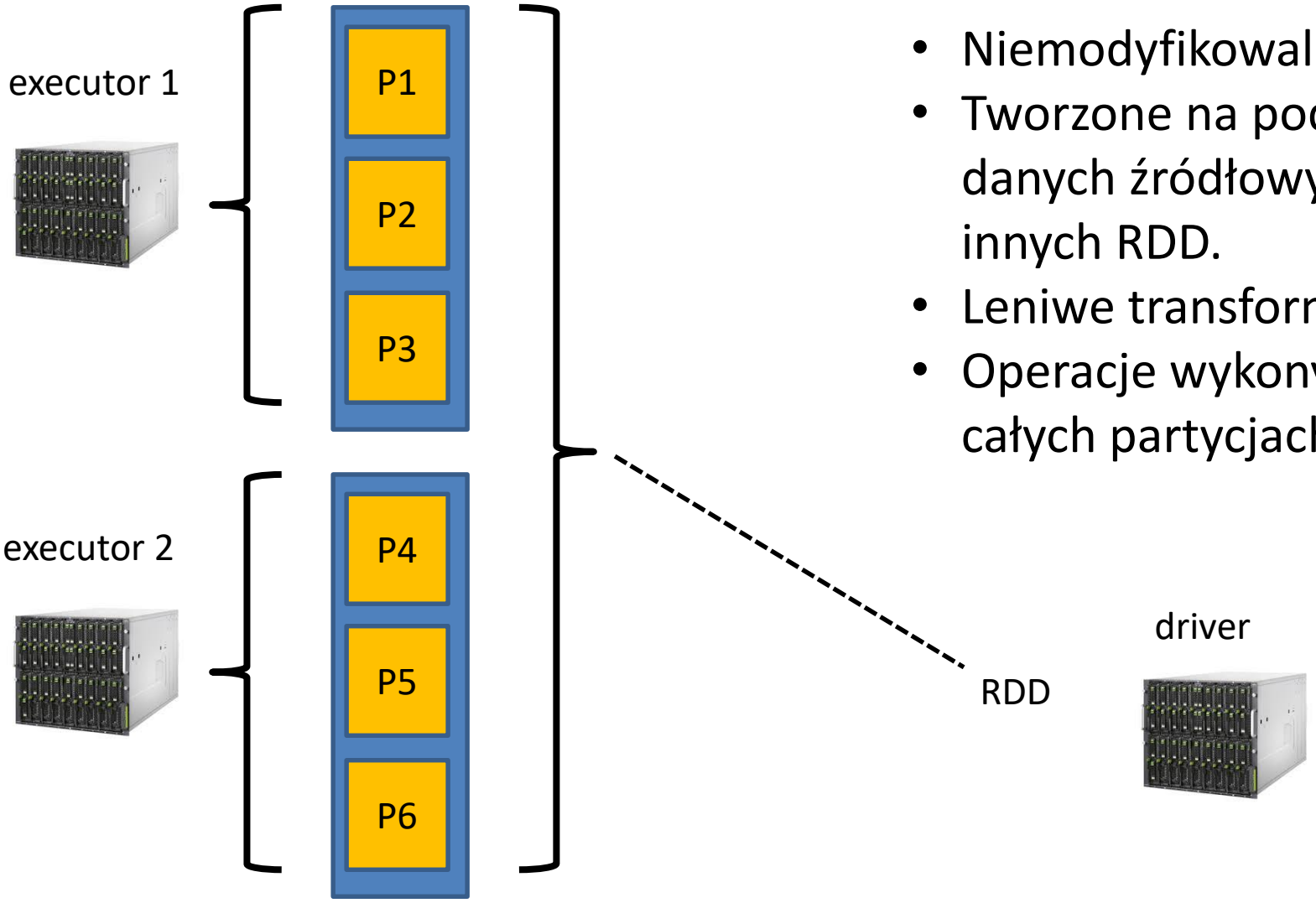
Spark - architektura



Spark – ćwiczenie 1

```
#uruchomić spark-shell jako aplikację YARN
spark-shell --master yarn

#sprawdzić w interfejsie webowym yarn status Sparka:
http://quickstart.cloudera:8088

#zamknąć sersję Sparka
exit

#uruchomić Sparka w trybie lokalnym zajmując 2 rdzenie
spark-shell --master local[2]

#sprawdzić w interfejsie webowym yarn status Sparka:
http://quickstart.cloudera:8088
```

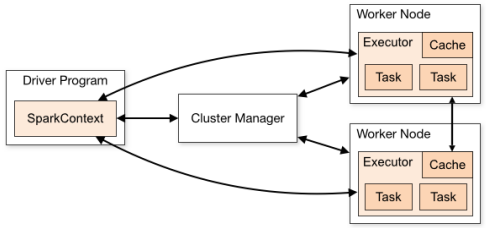
Spark – koncepcja partycji



- Niemodyfikowalne.
- Tworzone na podstawie danych źródłowych lub innych RDD.
- Leniwe transformacje.
- Operacje wykonywane na całych partycjach.

Spark – tworzenie RDD

```
#tworzenie RDD „w locie”
val localVector = 1 to 2000
val rdd_nums = sc.parallelize(localVector)

#tworzenie RDD na podstawie pliku w HDFS
val path = "/user/cloudera/flat/products/part-m-00000"
val rdd_products = sc.textFile(path)

#RDD na podstawie nieistniejącego pliku w HDFS
#transformacje sa leniwe, wiec wyrazenie nie bedzie zwalidowane
val path = "/path/does/not/exist"
val rdd_nonexistent = sc.textFile(path)

#sprawdzenie zawartości RDD:

rdd_nums.take(3)
rdd_products.take(3)
rdd_nonexistent.take(3)
```



Spark – sterowanie liczbą partycji

```
#tworzenie RDD „w locie”
val localVector = 1 to 1000000
val rdd_nums = sc.parallelize(localVector)

#sprawdzenie liczby partycji
rdd_nums.partitions.size

#zmiana liczby partycji
val rdd_repartitioned = rdd_nums.repartition(8)

rdd_nums.partitions.size
rdd_repartitioned.partitions.size
```

Spark – mapowanie

```
#tworzenie RDD na podstawie pliku w HDFS
val path = "/user/cloudera/flat/orders/part-m-00000"
val rdd_strings = sc.textFile(path)
rdd_strings

#map: String -> Array[String]

#...
```

Spark – mapowanie

```
#String -> Array[String]

def stringSplitter1(input: String): Array[String] = {
        val output =  input.split(",")
        return output
}
def stringSplitter2(input:String) = input.split(",")



val rdd_arrayOfStrings = rdd_strings.map(stringSplitter1)
val rdd_arrayOfStrings = rdd_strings.map(stringSplitter2)
val rdd_arrayOfStrings = rdd_strings.map(v => v.split(","))

rdd_arrayOfStrings.take(3)
```

Spark – filtrowanie

```
#String -> Array[String]
val rdd_fraud = rdd_arrayOfStrings.filter(
        v => v(3)=="SUSPECTED_FRAUD"
)

rdd_fraud.count(rdd_fraud.take(5)
)
```

Spark – redukcja

```
val path = "/user/cloudera/flat/orders/part-m-00000"
val rdd_strings = sc.textFile(path)

val rdd_arrayOfStrings = sc.map(v => v.split (","))

val rdd_statuses = rdd_arrayOfStrings.map( v=> v(3))

val rdd_KV = rdd_statuses.map( v => (v,1) )

val rdd_agg = rdd_KV.reduceByKey( (v1,v2) => v1+v2)

val localResult = rdd_agg.collect()
```

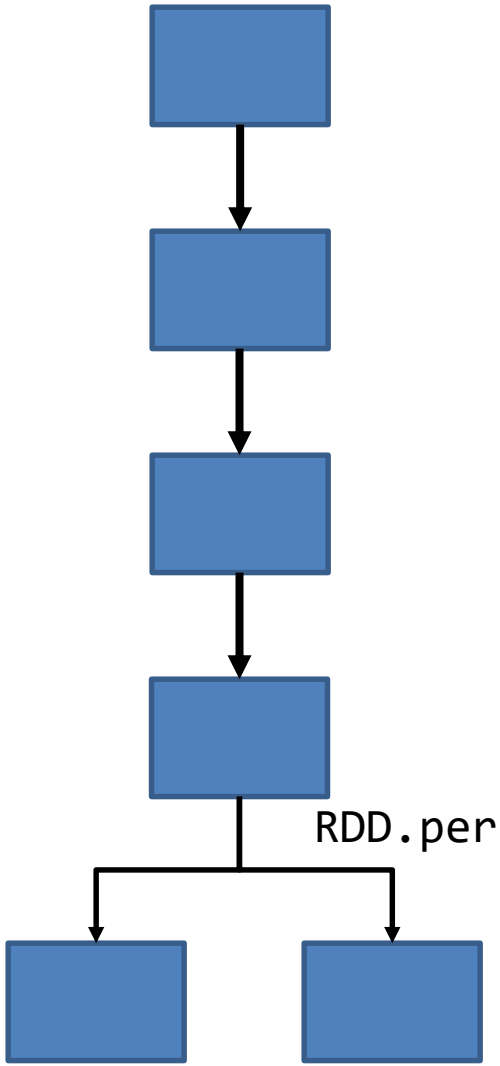
Spark – sortowanie

```
val rdd_aggSorted = rdd_agg.sortBy(
        v => v._2,
        false           //not ascending
)
```

Spark – zapis do HDFS

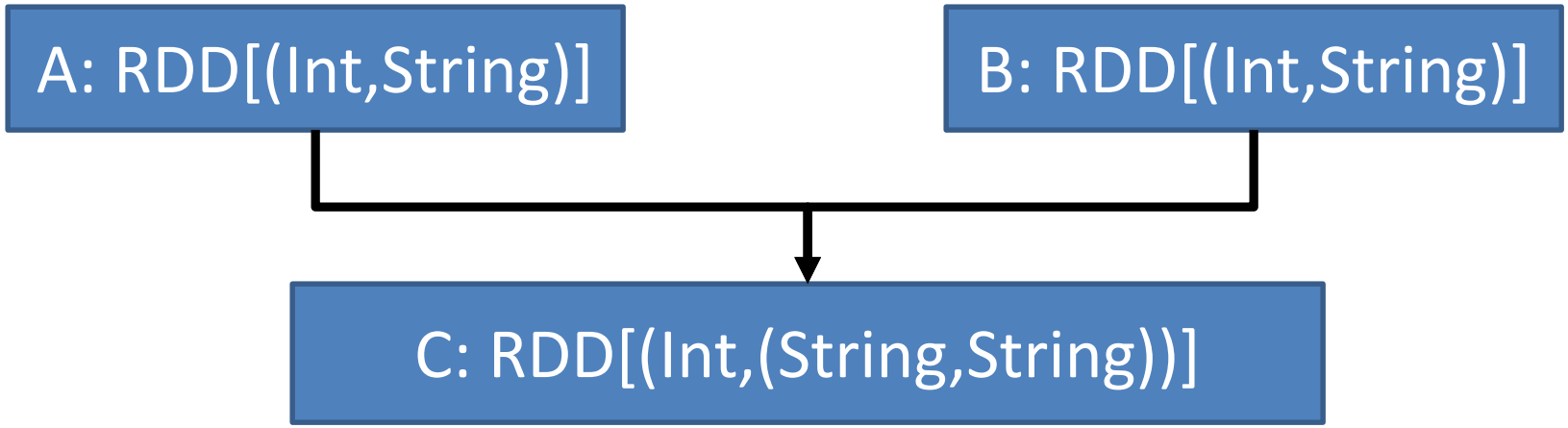
```
rdd_fraud.
repartition(1).
saveAsTextFile("/user/cloudera/flat/fraud")
```

Spark – RDD caching



RDD.persist(StorageLevel.MEMORY_ONLY)

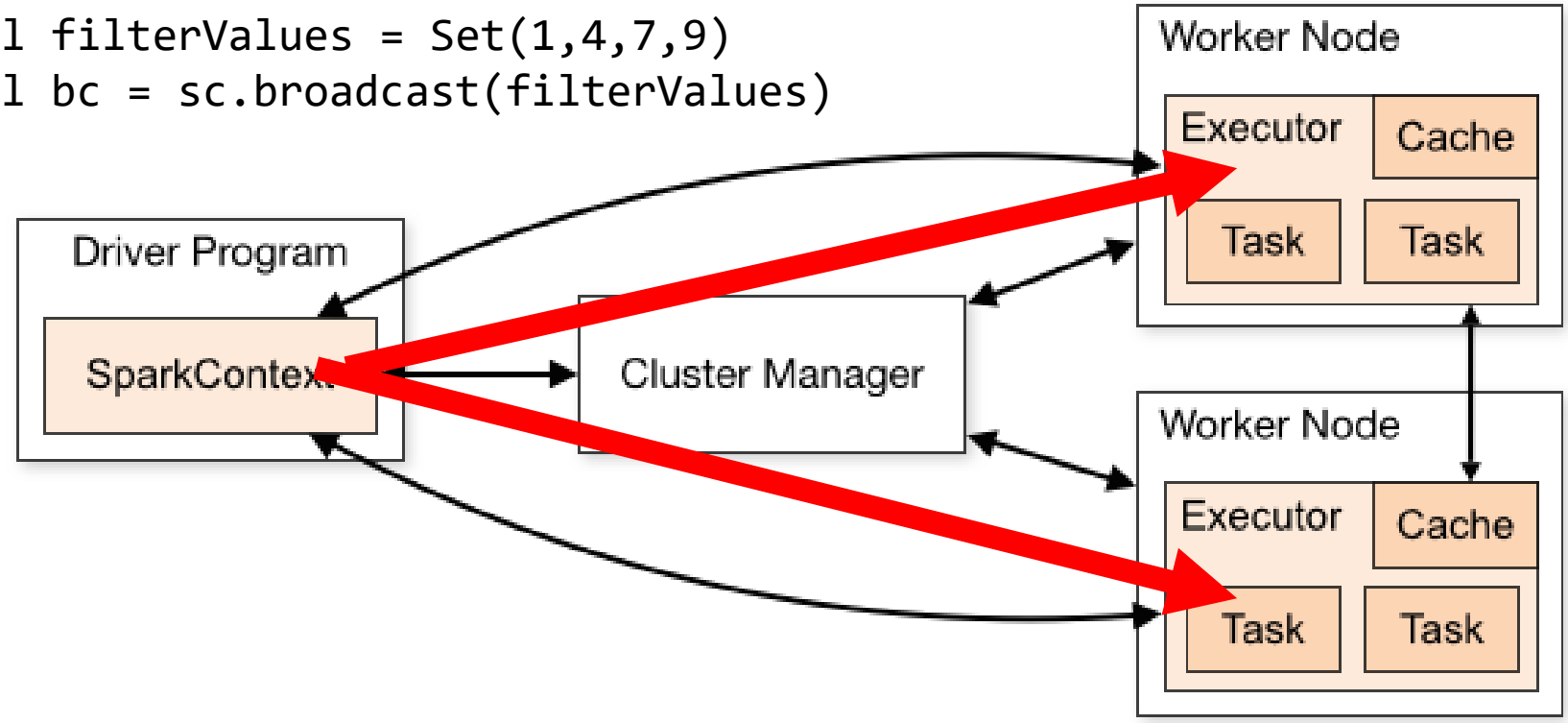
Spark - RDD join




```
val C = A.join(B)
```

Spark – broadcast

```
val filterValues = Set(1,4,7,9)
val bc = sc.broadcast(filterValues)
```



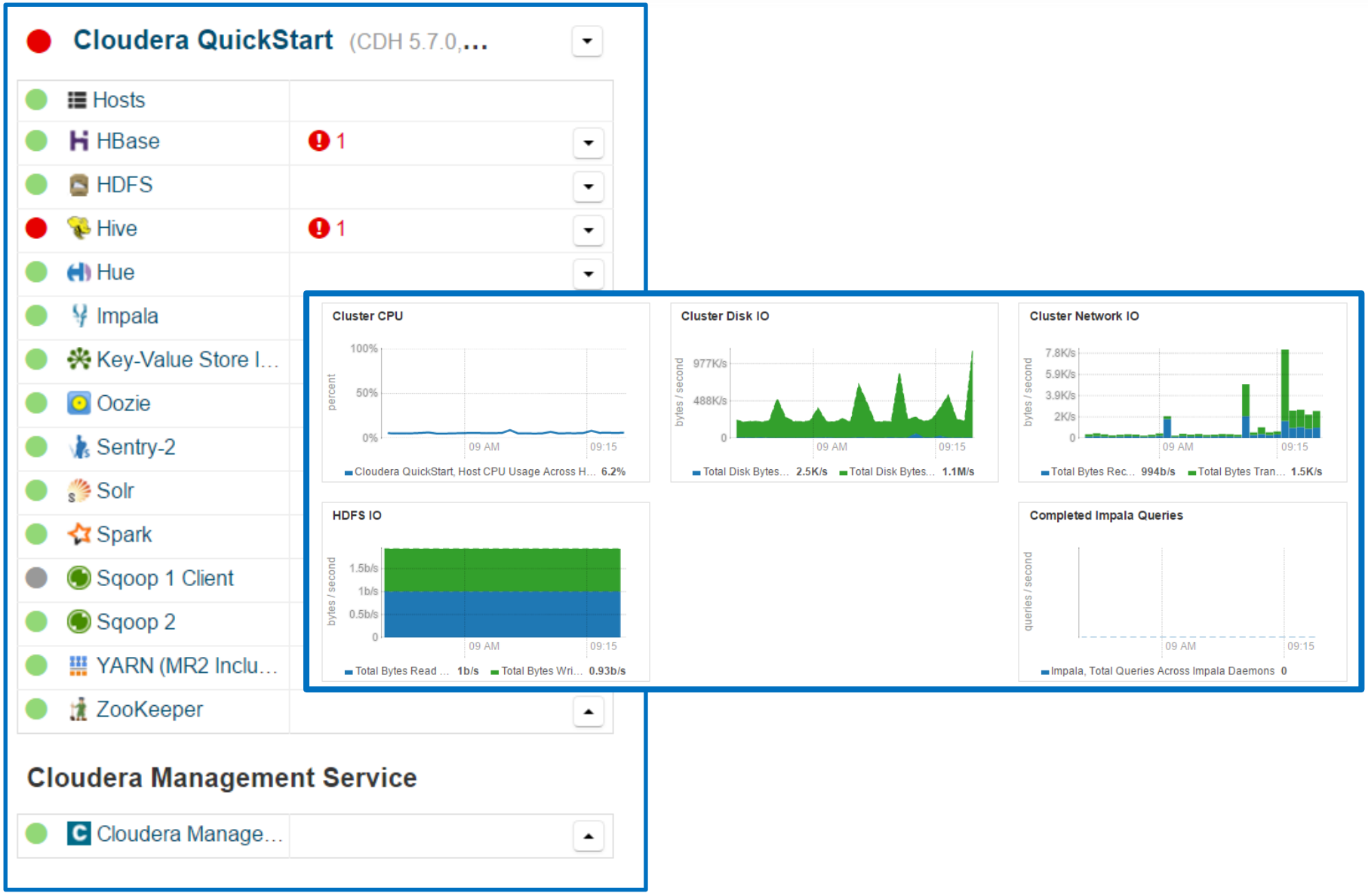

```
rdd.filter( v=> bc.value.contains(v._1))
```

Spark Streaming

Cloudera – architektura

Service	Master Hosts	Worker Hosts	Utility Hosts
HDFS	NameNode Secondary NameNode	DataNode	
YARN	ResourceManager JobHistory Server	NodeManager	
Hive	Hive Metastore Server HiveServer2		
Impala	Impala Statestore	Impala Deamon	Impala Catalog
Spark	JobHistory Server		

Cloudera Manager



Cloudera – architektura

Service	Master Hosts	Worker Hosts	Utility Hosts
HDFS	NameNode Secondary NameNode	DataNode	
YARN	ResourceManager JobHistory Server	NodeManager	
Hive	Hive Metastore Server HiveServer2		
Impala	Impala Statestore	Impala Deamon	Impala Catalog
Spark	JobHistory Server		
CM	CM Agent	CM Agent	CM Agent CM Server CM Database

Cloudera Navigator

Cloudera Navigator

Data Management Layer for Cloudera Enterprise

Audit & Access Control

Ensuring appropriate permissions & reporting on data access for compliance

Discovery & Exploration

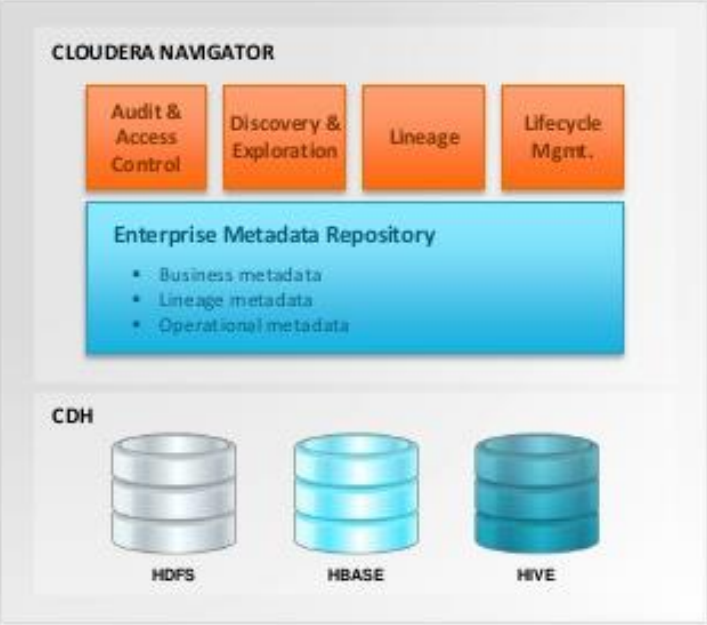
Finding out what data is available and what it looks like

Lineage

Tracing data back to its original source

Lifecycle Management

Migration of data based on policies



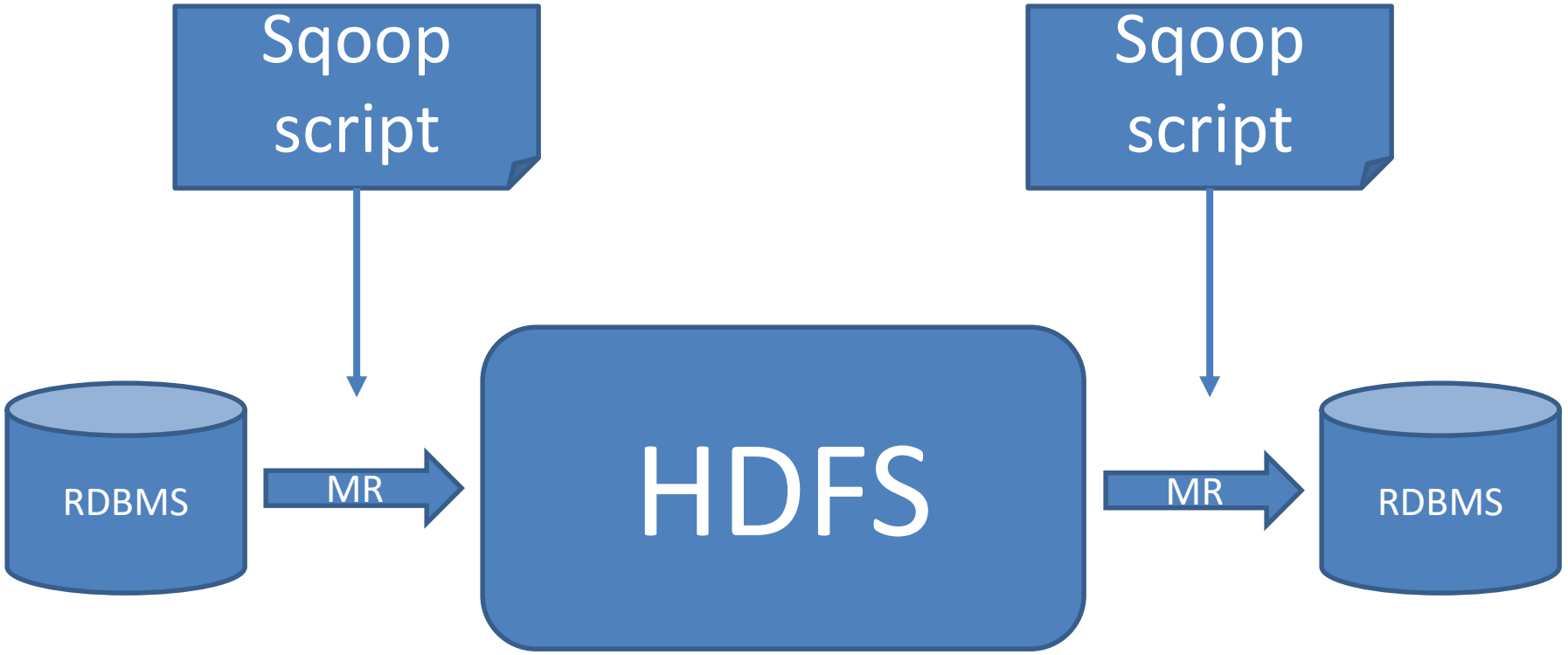
cloudera

Certyfikaty

cloudera

level	name	exam. code	Cloudera version	price	years valid	actual
Professional	Data Engineer	DE575	5.3.2, 7 nodes	400$	3	yes
Professional	Data Scientist	DS700 DS701 DS702	5.3.2, 7 nodes	600$ 600$ 600$	3	yes
Associate	Spark and Hadoop Developer	CCA175	5.3.2	295$	2	yes
Associate	Cloudera Certified Administrator for Apache Hadoop	CCA500	?	295$	2	yes

Sqoop

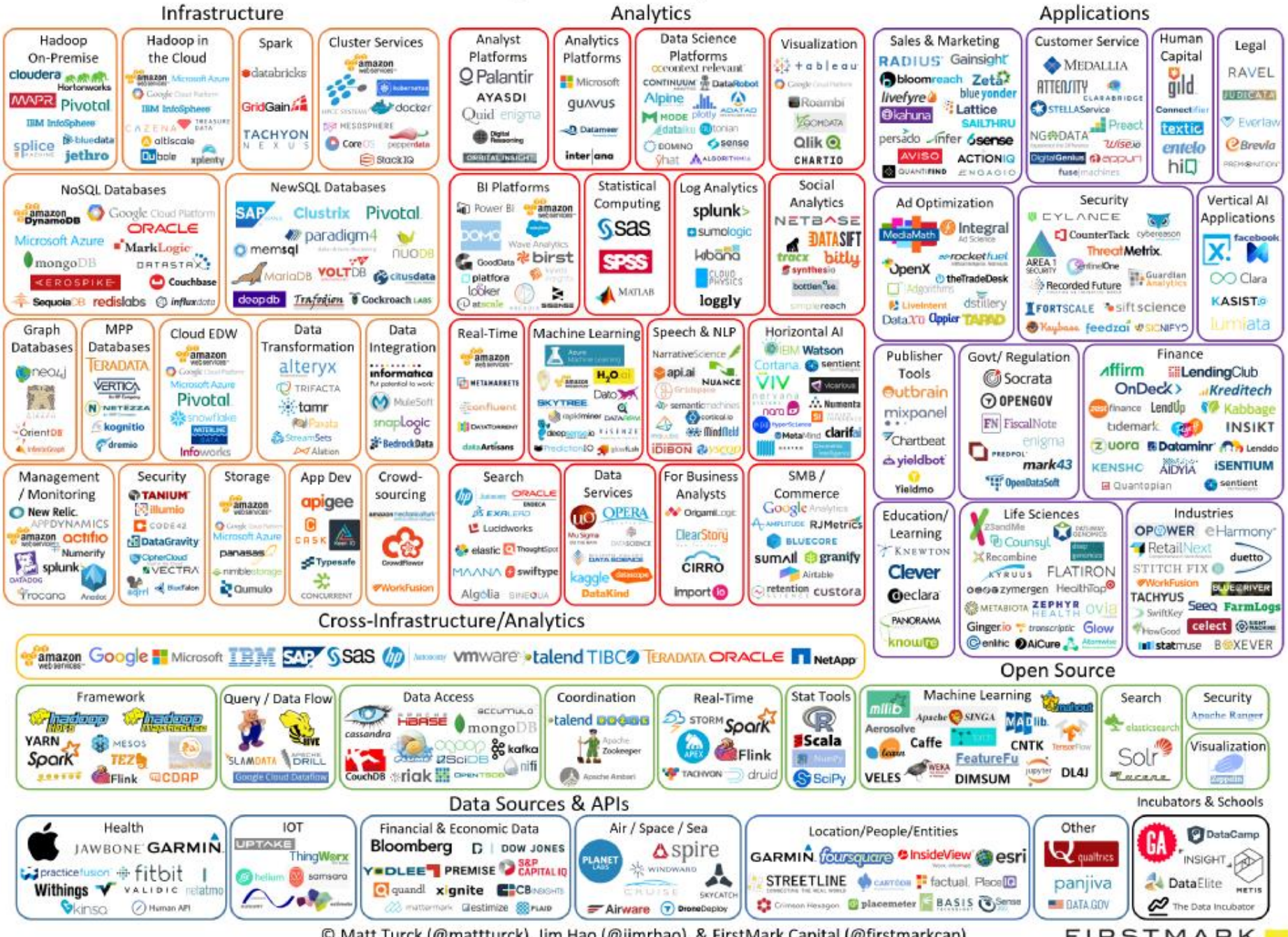


Sqoop

```
sqoop import \
--connect jdbc:mysql://localhost:3306/retail_db \
--username root \
--password cloudera \
--table products \
--target-dir /user/cloudera/flat/products \
--delete-target-dir \
-m 1
```

```
sqoop import \
--connect jdbc:mysql://localhost:3306/retail_db \
--username root \
--password cloudera \
--table departments \
--hive-import \
--hive-overwrite \
--hive-database lab \
--hive-table departments \
-m 1
```

Big Data Landscape 2016



Hadoop a bezpieczeństwo danych

- Kerberos
- Szyfrowanie danych w HDFS
- Szyfrowanie połączeń (TLS/SSL)
- Włączenie autentykacji w WebHDFS!
- Zarządzanie uprawnieniami: Sentry lub Ranger

Hadoop - przyszłość

- Enterprise Data Hub
- Data Lake

- EDW Symbiosis
- Data Lake

Hadoop - podsumowanie

- Hadoop służy z założenia rozwiązywaniu problemów analizy danych rzędu od kilku do kilkuset TB.
- Hadoop otwiera dla firm możliwość analizy danych „z zewnątrz”:
  - Internet of Things
  - Social Medial
  - …
- Nie brakuje inicjatyw zastępowania klasycznych systemów hurtowni danych przez Hadoopa (koszty licencji rozwiązań komercyjnych).
- Większość skomplikowanych problemów analizy danych nie wymaga przetwarzania zbiorów większych niż kilkadziesiąt GB, czyli mieszczących się w pamięci operacyjnej jednej maszyny.

Pytania