Paweł DRZYMAŁA[1], Henryk WELFLE[1], Agnieszka DRZYMAŁA[2]

Politechnika Łódzka, Instytut Mechatroniki i Systemów Informatycznych (1), Uniwersytet Łódzki, Instytut Ekonomii (2)

doi:10.15199/48.2019.01.08

Efektywne przetwarzanie i integracja dużych zbiorów danych w środowisku Hadoop

Streszczenie. *Rozwój nowych kanałów elektronicznej wymiany informacji przyczynia się do powstania coraz większej ilości danych. Dane te są często zróżnicowane, niejednorodne i składowane bez ściśle zdefiniowanej struktury. W ciągu ostatnich 2 lat przyrosło 90% danych, jakie zostały wygenerowane od początku istnienia ludzkości. W artykule zaprezentowano architekturę i możliwości środowiska Hadoop powstałego w celu efektywnego przetwarzania i integracji dużych zbiorów danych. Przedstawiono cechy tej platformy oraz jej skalowalność. Omówiono metodę działania systemu plików HDFS oraz odporności na błędy składowania tego systemu. Zaprezentowano ideę współpracy węzłów klastra Hadoop oraz wykonywania działań typu Map – Reduce.*

Abstract*. The development of new channels of electronic information exchange contributes to the emergence of more and more data. These data are often diverse, heterogeneous and stored without a strictly defined structure. Over the past two years, 90% of the data has been generated since the beginning of human civilization. The article presents the architecture and possibilities of the Hadoop environment for the effective processing and integration of large data sets. It also presents the features of this platform and its scalability as well as discussed the method of operation in the HDFS file system and the resistance to storage errors of this system. The scheme of cooperation of the Hadoop cluster nodes to perform MapReduce operation was presented. (**Effective processing and integration of large data sets in the Hadoop environment**).*

Słowa kluczowe: Big Data, Hadoop, HDFS,Hive, Pig Latin, Spark.
Keywords: Big Data, Hadoop, HDFS,Hive, Pig Latin, Spark..

Wprowadzenie

Pojęcie Big data ma zastosowanie wszędzie tam, gdzie dużej ilości danych cyfrowych towarzyszy potrzeba zdobywania nowych informacji lub wiedzy. W 2001 roku META Group opublikowała raport, który opisuje Big Data w modelu 3V:

- Volume - duża ilość danych,
- Velocity - duża zmienność danych,
- Variety - duża różnorodność danych,

Model ten w późniejszym okresie został rozszerzony o czwarte V:

- Value - weryfikacja posiadanych danych.

Istnieje umowny podział wolumenów danych na dwa typy:

- dane w ruchu – są to tzw. strumienie informacji (np. komentarze Twitter/Facebook, dane giełdowe, pochodzące z czujników – np. do przewidywania pogody lub pomiaru życiowych oznak noworodków czy też położenia produktów lub pojazdów),
- dane w spoczynku – są to tzw. oceany informacji (np. logi, poczta, media społecznościowe itp.), czyli dane, które nie zmieniają się już w czasie ale są składowane na nośnikach informacji.

Około 80% danych na świecie jest bez ściśle zdefiniowanej struktury. Obecnie co 2 dni wytwarzamy tyle danych, ile ludzkość wyprodukowała od początku cywilizacji do początku XXI wieku. Takich danych zazwyczaj nie można w standardowy sposób przetworzyć z użyciem metod znanych z baz relacyjnych stąd pojawiło się pojęcie Big Data.

W celu sprostania wyzwaniom opracowany został nowy rodzaj technologii ogólnie zwany Big Data. Wiele z tych nowych technologii zostało zgrupowanych pod pojęciem NoSQL.

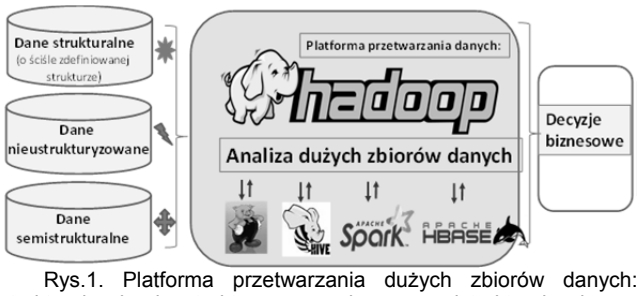
Wyzwania związane z dużymi zbiorami danymi są następujące: problem przechwytywanie tych danych, czas trwania i ich aktualności, analiza i problem przeszukiwania tak, by wydobyć istotną informację, dzielenie danych i ich agregowanie, transfer z wykorzystaniem sieci teleinformatycznej, prezentacja. Badania firmy Gartner wykazało, że tylko 1 do 15% przedsiębiorstw w pełni wykorzystuje zalety dużych zbiorów danych. Ale te przedsiębiorstwa, którym się uda analiza dużych zbiorów zgromadzonych danych, uzyskają przewagę nad nieprzygotowaną konkurencją w głównych wskaźnikach biznesowych.

Platforma Hadoop

Hadoop to platforma i projekt „open source" napisany w Javie, zoptymalizowany do obsługi ogromnych ilości danych przez zrównoleglenie przetwarzania.

Masywne przetwarzanie równoległe odbywa się z wielką wydajnością (są to jednak operacje wsadowe obsługujące ogromne ilości danych, więc czas odpowiedzi nie jest natychmiastowy). Niezawodność zapewniana jest przez replikację gdyż Hadoop replikuje dane na różnych maszynach.

Hadoop – umożliwia składowanie i analizę wszystkich rodzajów danych (Rys.1.) w tym: danych strukturalnych o ściśle zdefiniowanej strukturze (np. składowanych w bazach relacyjnych), nieustrukturyzowanych o dowolnych formatach zapisu oraz semistrukturalnych (np. danych XML), celem podejmowania na ich podstawie właściwych decyzji.



Rys.1. Platforma przetwarzania dużych zbiorów danych: strukturalnych, nieustrukturyzowanych oraz semistrukturalnych.

Platforma Hadoop - umożliwia pracę w środowisku wielu (nawet tysięcy) węzłów i przetwarzanie peta bajtów danych w sposób wysoce równoległy i opłacalny. Skalowanie architektury Hadoop dla wersji MR2 możliwe jest do ok. 6K węzłów na klaster. Daje to możliwość przetwarzania do około 200 PB danych na klaster oraz wsparcie dla przetwarzania blisko miliarda plików i bloków. Dodatkowo klastry można łączyć w federacje Hadoop co jeszcze zwiększa możliwości przetwarzania całego systemu.

Pojedynczy węzeł wnosi do sieci węzłów moc obliczeniową (CPU) oraz powierzchnię do składowania danych (dyski HDD). Dużą zaletą tej platformy jest to, że w razie potrzeby nowe węzły można dodawać bez zmian formatu analizowanych i już załadowanych danych oraz napisanych zadań.

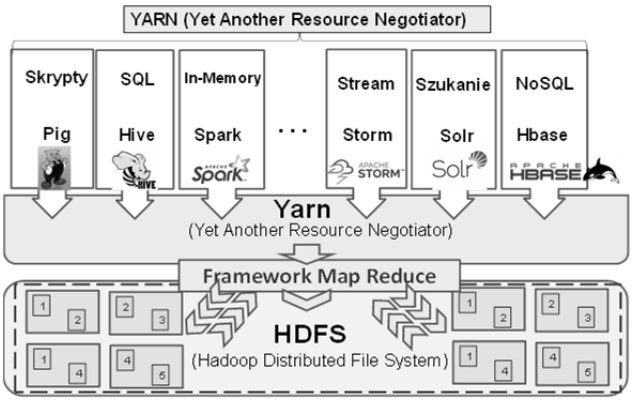
W starszej wersji architektury Hadoop – MR1 administrator musiał zadecydować ile gniazd wykonuje operacje mapowania i ile gniazd wykonuje operacje redukcji na każdym węźle. Wpływało to na wydajność całego klastra, która zależała od wydajności wykonywanych operacji w każdym węźle. W podejściu MR2 i bazującym na YARN – ręczne precyzowanie liczby operacji MapReduce na węzeł nie jest wymagane . Należy również podkreślić, że YARN MR2 jest kompatybilny z MR1 – i może pracować w tym trybie.

Rdzeń architektury Hadoop zawiera następujące główne składniki (Rys.2.):

- HDFS (Hadoop Distributed File System) - rozproszony system plików, który przechowuje dane na maszynach, zapewniając bardzo dużą zagregowaną przepustowość w klastrze,

- MapReduce Engine – silnik i zarazem framework umożliwiający wykonywanie obliczeń na danych składowanych w rozproszonym systemie plików (HDFS) i który jest nakładką na HDFS.

- Apache YARN "Yet Another Resource Negotiator" - to po części tzw. nakładka na MapReduce, odpowiedzialna za zarządzanie zasobami komputerowymi w klastrach i wykorzystania ich w zadaniach użytkowników. YARN stanowi i umożliwia korzystanie z szeregu frameworków będących całą platformą aplikacji. YARN jest architektonicznym centrum efektywnego wykonywania operacji Hadoop z użyciem różnych przewidzianych do tego dialektów języka i frameworków. Przykładowo HQL (Hive Query Language) dla framerorka Hvie czy język PigLatin.dla frameworka Pig.



Rys.2. Platforma Hadoop do przetwarzania dużych zbiorów danych z wybranymi frameworkami.

Przetwarzanie dużych zbiorów danych z wykorzystaniem platformy Hadoop rożni się od znanego z systemów RDBMS przetwarzania danych (Rys.3).

Podstawowe różnie przy przetwarzaniu danych w tych dwóch platformach to:

- schemat - dla danych relacyjnych schemat jest wymagany przed zapisem natomiast na platformie Hadoop schemat nie jest ważny dopóki dane nie są wykorzystywane przy odczycie,

- prędkość – w systemach RDBMS odczyty danych są szybkie w systemie Hadoop to zapisy danych są stosunkowo szybkie (ale niekoniecznie odczyty),

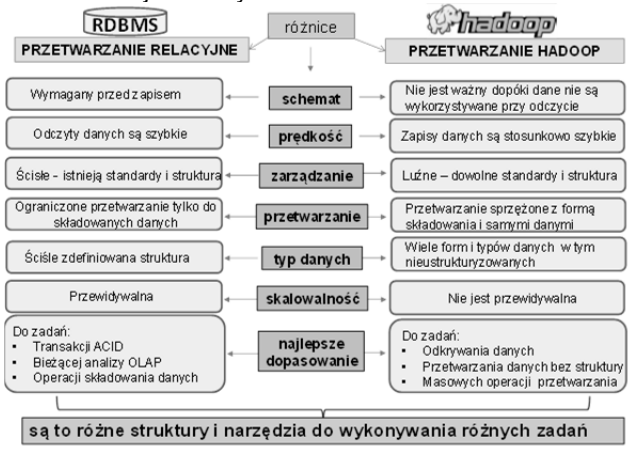
- zarządzanie – w bazach relacyjnych zarządzanie jest ścisłe, gdyż istnieją standardy i struktura. Na platformie Hadoop zarządzanie jest luźne, gdyż panują dowolne standardy i struktura,

- przetwarzanie – w typowych bazach relacyjnych przetwarzanie jest przede wszystkim ograniczone do składowanych danych w tych strukturach, w Hadoop przetwarzanie jest sprzężone z formą składowania i samymi danymi,

- typ danych – dla składowania w systemach relacyjnych są znane gdyż struktura do zapisu takich danych musi być ściśle zdefiniowana, platforma Hadoop posiada i wykorzystuje wiele form i typów danych w tym danych nieustrukturyzowanych,

- skalowalność – dla systemów RDBMS jest ona przewidywalna – można oszacować i przewidzieć zapotrzebowanie na miejsce i pojemność macierzy dyskowej oraz moc obliczeniową,

- dopasowanie – RDBMS o przetwarzaniu ROLAP i OLAP – dedykowane są do przeprowadzania zadań transakcji spełniających zasady ACID oraz zadań bieżących analiz OLAP w tym operacji składowania danych. Hadoop ma najlepsze dopasowanie do zadań: odkrywania danych i masowych operacji przetwarzania danych bez ściśle zdefiniowanej struktury.



Rys.3. Porównanie platformy Hadoop i systemów relacyjnych baz danych (RDBMS).

Hadoop distributed file system (HDFS)

Hadoop Distributed File System (HDFS) – Rys.4. to system plików, który działa na bazie istniejącego (w UNIX) systemu plików i który zaprojektowany został do obsługi bardzo dużych plików z wzorcami dostępu do transmisji strumieniowej. Używa on własnych bloków o zdefiniowanej pojemności do przechowywania pliku lub części pliku.

Hadoop działa najlepiej z bardzo dużymi plikami. Im większy plik, tym mniej czasu Hadoop spędza na poszukiwanie kolejnej lokalizacji danych na dysku i tym więcej czasu działania odbywa się na granicy przepustowości dysków. Poszukiwania są na ogół kosztownymi operacjami.

Ponieważ Hadoop jest przeznaczony do pracy na całym zbiorze danych, najlepiej jest zminimalizować poszukiwania, korzystając z dużych rozmiarów bloków plików HDFS.

Bloki plików w HDFS są rozmiarów: 64 MB, 128 MB (domyślnie), 256 MB (zalecane) - w porównaniu do 4 KB w systemie UNIX. Zatem jeden blok HDFS jest obsługiwany przez wiele bloków systemu operacyjnego. Należy przy tym wspomnieć, że bloki HDFS idealnie nadają się do replikacji w celu zapewnienia odporności na błędy i dostępności danych.

Zalety bloków HDFS:

- mają stały rozmiar – dzięki temu łatwe staje się obliczenie, jak wiele takich bloków można zmieścić na

dysku i jak one pasują do rozmiaru fizycznych bloków dysku,

- bloki można rozłożyć na wiele węzłów – dzięki temu plik HDFS może być większy niż pojedynczy dysk w klastrze,

- bloki HDFS nie „marnują" miejsca. Jeśli plik nie jest równy wielokrotności rozmiaru bloku to tzw. reszta nie zajmuje miejsca całego bloku. Przykładowo: plik 420 MB megabajtów zużywa cztery bloki o rozmiarze 128MB, ale czwarty blok nie zużywa pełnej 128 MB.

Klaster HDFS

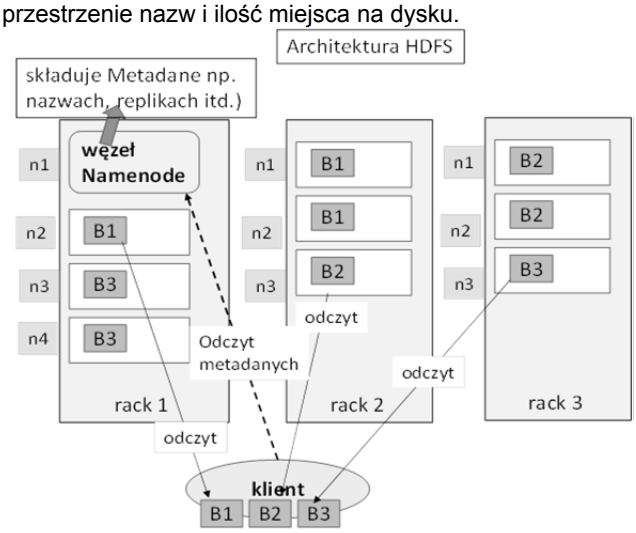
Klaster HDFS składa się z:

- węzła (a od MR2 węzłów - w tym jednego aktywnego i jednego w trybie stand by) NameNode, który zarządza metadanymi klastra, i który aktywnie monitoruje liczbę replik bloków propagowanych danych. Gdy replika bloku zostanie utracona z powodu awarii dysku, NameNode tworzy inną replikę tego bloku. NameNode zachowuje drzewo obszaru nazw i mapowania bloków dla węzłów danych tzw. DataNodes, trzymając cały obraz obszaru nazw w pamięci RAM.

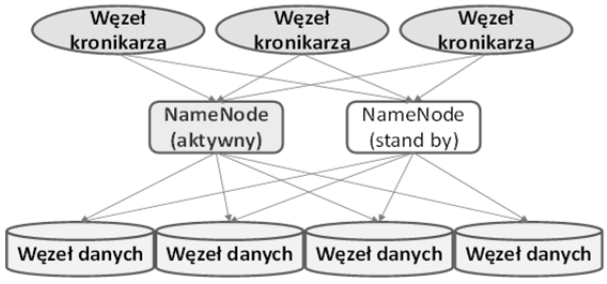
- węzłów składowania i przetwarzania danych DataNodes, które przechowują powielone redundantnie dane.

W MR2 wprowadzono również węzły kronikarzy JournalNode których liczba powinna być nieparzysta. Węzły te decydują który węzeł NameNode powinien być w danej chwili aktywny: podstawowy czy ten znajdujący się w w trybie stand by. Zaprezentowane zostało to na rysunku. Rys.5.

Pliki i katalogi są reprezentowane na węźle NameNode przez węzły wewnętrzne. Węzły wewnętrzne rejestrują atrybuty takie jak uprawnienia, czasy modyfikacji i dostępu, przestrzenie nazw i ilość miejsca na dysku.



Rys.4. Architektura HDFS.



Rys.5. Model w oparciu o dwa węzły NameNode oraz węzły kronikarzy JournalNode które decydują który węzeł NameNode powinien być w danej chwili aktywny.

Praca z interfejsem wiersza poleceń HDFS

HDFS może być obsługiwany przez:

- interfejs wiersza polecenia (HDFS Command line interface). Komendy takie zaczynają się od "hadoop", spacji i "fs" (lub "dfs").

- środowisko graficzne dostarczone przez określonego dostawcę usługi dla obsługi klastra Hadoop. W takim przypadku obsługa jest uproszczona. Jednak bez znajomości podstawowych komend nie umożliwia elastycznego i efektywnego wykonywania zadań na klastrze.

W pierwszym przypadku praca z systemem plików HDFS, wymusza znajomości podstawowych poleceń: hadoop fs. Polecenia powłoki systemu plików HDFS zaczynają się od znaku kreski. komendy powłoki systemu plików HDFS mogą przyjmować ścieżki jako argumenty. Ścieżki te można wyrazić w postaci jednolitych identyfikatorów URI. Format URI składa się z schematu, uprawnień do struktury katalogów oraz plików i ścieżki. Polecenia Hadoop fs obsługują są różne systemy składowania. Lokalny system plików ma schemat "file". HDFS ma system składowania zwany "hdfs". Przykładowo aby wyświetlić katalogi określonej hierarchii struktury katalogu głównego należy wydać polecenie:

```
hadoop fs -ls /
```

Do zakładana katalogu w systemie składowania HDFS można wykorzystać polecenie:

```
hdfs dfs -mkdir /user/test1
```

Można również potokować (używając znaku |) dowolne polecenie HDFS i polecenia takie mogą być używane z powłoką Linux. Przykładowo można łatwo używać polecenia grep ze strukturą HDFS, wykonując:

```
hadoop fs -ls /user/test1 | grep test
```

Podobnie można zmienić uprawnienia z użyciem poleceń HDFS:

```
hdfs dfs -chmod 777 /user/test1
```

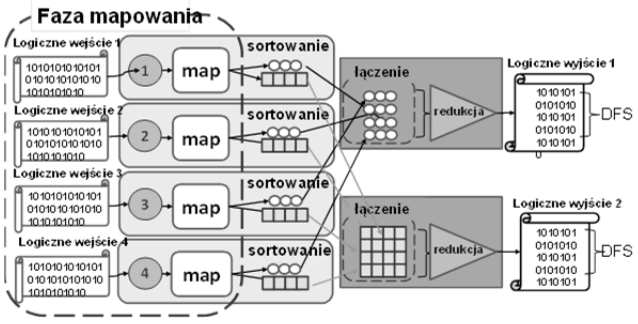
Innym przykładem jest np. sprawdzanie przez terminal lub putty, czy plik został prawidłowo załadowany do systemu plików Hadoop odwołując się do dfs:

```
hdfs dfs -ls /user/test1/data
```

Operacje mapowania i redukcji

Aby zrozumieć operacje MapReduce, należy rozbić te zadania na składowe: mapowania i redukcji (Rys. 5, Rys.6).

Obie te operacje pochodzą z funkcyjnych języków programowania. Są to języki, które pozwalają przekazać funkcje jako argumenty do innych funkcji.



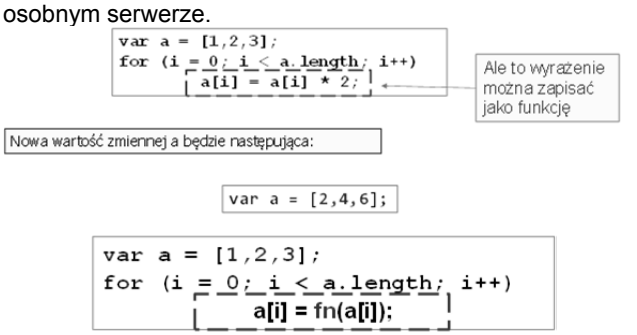
Rys.5. Fazy operacji mapowania i redukcji na danych.

Operacje MapReduce w ogólnym skrócie realizowane są podczas dwóch podstawowych kroków (chociaż pamiętać należy o tym, że w węzłach wykonywane są

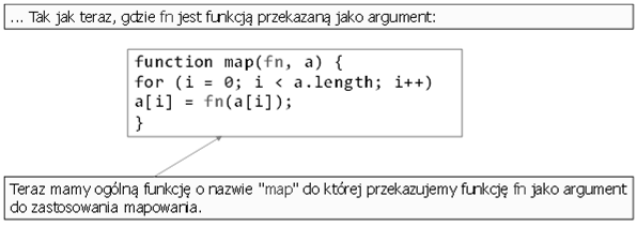
również inne operacje np. tasowania i sortowania oraz łączenia):

- krok: "map" – polega na tym, że węzeł nadzorczy pobiera dane z wejścia i dzieli je na mniejsze pod zadania. Następnie przesyła je do węzłów roboczych. Każdy z węzłów roboczych może albo dokonać kolejnego podziału na podzadania, albo przetworzyć problem i zwrócić odpowiedź do głównego programu.
- krok: "reduce" - główny program „zbiera” odpowiedzi ze wszystkich podzadań i łączy je w jeden wynik - odpowiedź na główny problem.

Zatem zaletą MapReduce jest umożliwienie łatwego rozproszenia operacji gdyż każda z operacji "map" jest niezależna od pozostałych, i może być ona realizowana na osobnym serwerze.



Rys.6. Cała "pętla for" może zostać przepisana jako funkcja mapy



Rys.7. Przykładowa funkcja „map” do której przekazujemy funkcję fn jako argument do zastosowania mapowania

Operacja mapowania - jest wykonywaniem tego samego działania dla każdego elementu tablicy. Mapper natomiast to zazwyczaj mały program, propagowany w klastrze do węzłów lokalnych z danymi. Każdy z nich przeprowadza operacje: cięcia, tasowania i filtrowania oraz transformacji danych wejściowych czego rezultatem są pogrupowane pary klucz – wartość <key, value>.

Na rysunku Rys.8. zaprezentowany został schemat przepływu danych oraz wykonywane w klastrze operacje. W przykładzie założono, że należy przekształcić dane wejściowe, składające się z listy powtarzających wystąpień do postaci na wyjściu czyli do konkretnej liczby tych wystąpień dla danej wejściowej.

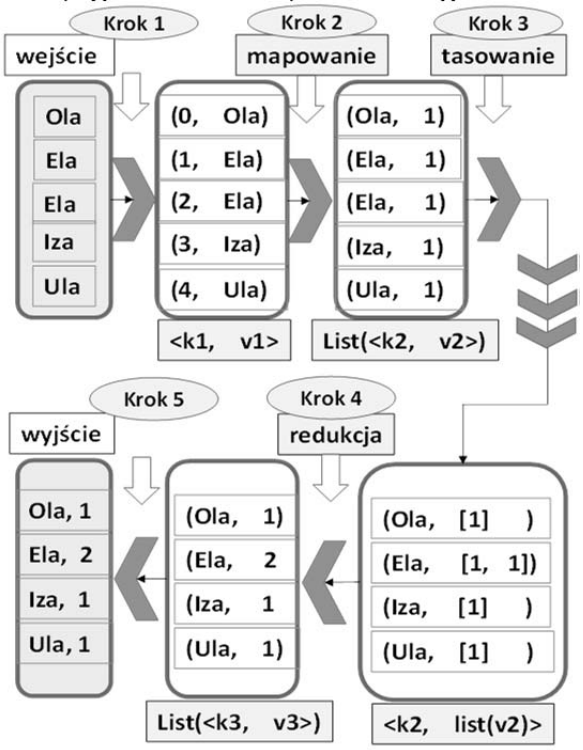
Hadoop w pierwszym kroku operacji indeksuje dane wejściowe i łączy w pary: klucz-wartość. Klucz dostarczany jest na podstawie rosnącego numeru z sekwencji. W dalszym kroku tworzone są rozbiory wejściowe. Hadoop oblicza, jak podzielić dane wejściowe, aby kod mapujący (i tzw. mapery) działał równolegle.

W kroku drugim funkcja, którą pisze się dla mapera, musi wziąć pary klucz-wartość i utworzyć coś, co może zostać zredukowane w następnym kroku. Najprostszym rozwiązaniem jest uczynienie każdego imienia kluczem i wprowadzenie każdej wartości typu „para”: <imię 1>. Wynikiem będzie zatem lista <k2, v2>.

W kroku trzecim program tasujący grupuje rekordy które mają ten sam klucz razem.

W kroku czwartym następuje redukcja w której sumuje się wartości, Wynikiem tej operacji jest suma dla każdego klucza.

Krok piąty to zwrócenie odpowiedzi na wyjście.



Rys.8. Schemat logiki przepływu danych przy wykonywaniu operacji MapReduce.

Podsumowanie

Big Data ma zastosowanie wszędzie tam, gdzie dużej ilości danych cyfrowych towarzyszy potrzeba ich analizy.

Kluczowym elementem architektury jest system plików HDFS oraz MapReduce umożliwiający efektywne wykonywanie obliczeń na danych w rozproszonym systemie plików HDFS

W artykule zaprezentowano architekturę i możliwości środowiska Hadoop powstałego w celu efektywnego przetwarzania i integracji dużych zbiorów danych.

Przedstawiono cechy tej platformy oraz jej skalowalność. Omówiono metodę działania systemu plików HDFS oraz odporności na błędy składowania tego systemu.

Zaprezentowano ideę współpracy węzłów klastra Hadoop oraz wykonywania działań typu Map – Reduce.

***Autorzy**: dr inż. Paweł Drzymała, Politechnika Łódzka, Instytut Mechatroniki i Systemów Informatycznych, Stefanowskiego 18/22, 90-924 Łódź, E-mail: pawel.drzymala@p.lodz.pl; dr inż. Henryk Welfle, Politechnika Łódzka, Instytut Mechatroniki i Systemów Informatycznych, Stefanowskiego 18/22, 90-924 Łódź, E-mail: henryk.welfle@p.lodz.pl. dr Agnieszka Drzymała, Uniwersytet Łódzki, Instytut Ekonomii, Katedra Gospodarki Światowej i Integracji Europejskiej ul. Rewolucji 1905 Nr. 41, 90-255 Łódź, agnieszka.drzymala@uni.lodz.pl.*

LITERATURA

[1] Docs http://hadoop.apache.org/docs/current/
[2] IBM Data source: https://developer.ibm.com/hadoop/
[3] T. White : Hadoop: The Definitive Guide, Fourth Edition, O'REILLY 2015,
[4] Rymaszewski J., Lebioda M., Korzeniewska E Simulation of the loss of superconductivity in a three-dimensional model of the metal-superconductor, Przegląd Elektrotechniczny, 2012 R.88, nr 12B, s. 183-186.
[5] Stempien Z., Kozicki M., Pawlak R. Korzeniewska E., Owczarek G., Poscik A., Sajna D. , Ammonia gas sensors ink-jet printed on textile substrates, Conference: 15th IEEE Sensors, Orlando, FL Date: OCT 30-NOV 03, 2016