MapReduce (9.5.2011)

Hadoop MapReduce

tutoriál

Prečo MapReduce?

- HPC, gridové riešenia, MPI
  - orientované na výpočet, špecializované HW riešenia, dáta nie sú dominujúce a zvyčajne sú v špecializovaných dátových uzloch
- Čo so spracovaním veľkých (veľa) údajov?
  - Google (2008): **20PB** údajov denne
  - Facebook (2009): **2.5PB** s denným nárastom **15TB**
- Komplexnosť a náročnosť implementácie rastie, ak chceme škálovateľnosť, odolnosť voči chybám, ...

MapReduce

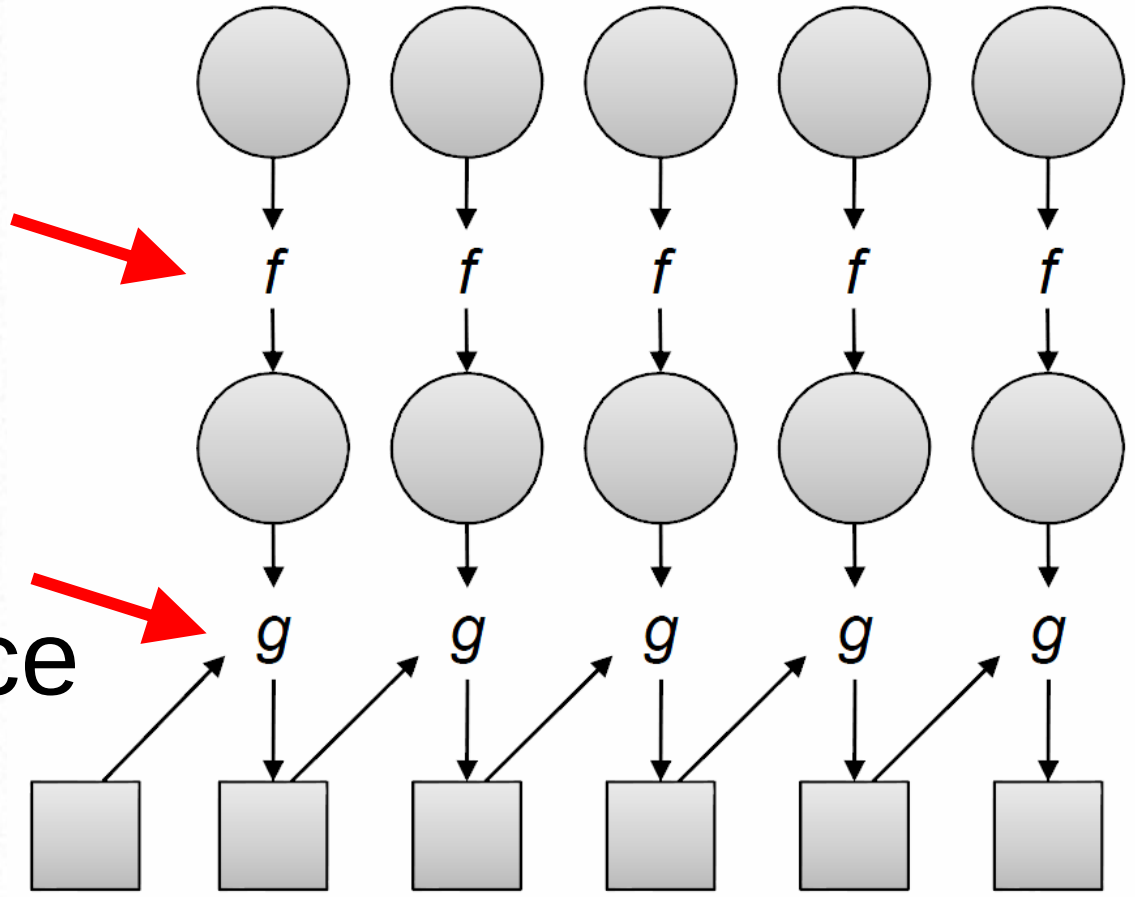
- SW framework vyvinutý spoločnosťou **Google** na spracovanie veľkých dátových súborov
  - open-source implementácie v projekte Apache Hadoop (založená na Jave)
- Inšpirovaný **funkcionálnym** programovaním
  - Map & Fold
- Stratégia **rozdeľuj a panuj**
- Určený pre clustre **z bežných PC**
  - dôraz na odolnosť voči chybám

Map & Fold

Priestor na paralelizmus?

map

fold/
reduce



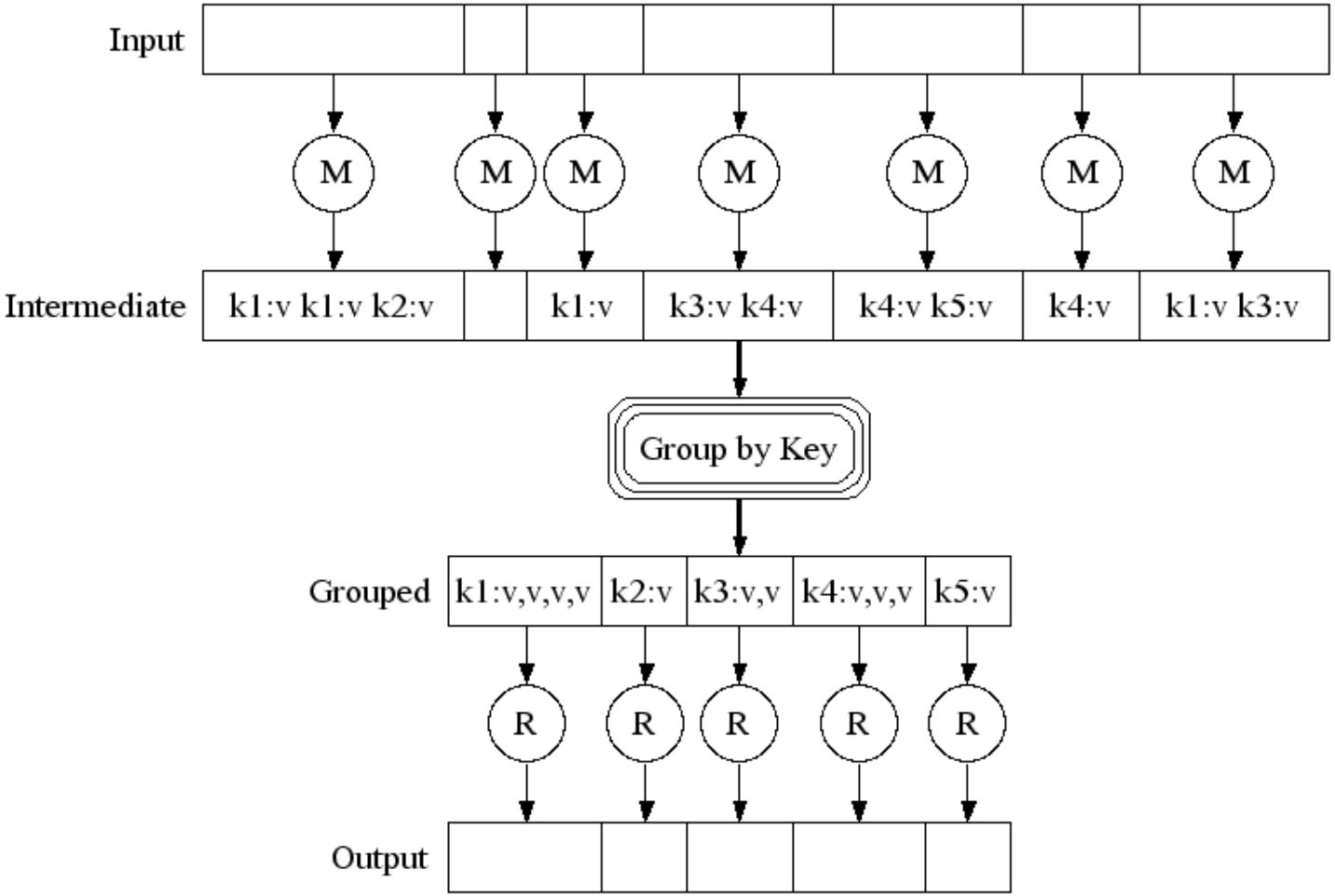
MapReduce - princíp

- Vstup:
  - množina dvojíc **(kľúč, hodnota)**
  - rovnaký typ všetkých kľúčov
  - rovnaký typ všetkých hodnôt
- Operácia **map**:
  - aplikovaná na každú dvojicu na vstupe
  - výsledkom je množina dvojíc (kľúč, hodnota)
  - **map(k, v) -> {(k_1, v_1), (k_2, v_2), ...}**
  - kľúče v emitovaných dvojiciach musia byť rovnakého typu a musia byť navzájom porovnateľné

MapReduce - princíp

- Operácia **reduce**
  - vstupom pre operáciu reduce je kľúč ***k*** a (neusporiadaný) **zoznam všetkých hodnôt**, ktoré boli v operácii **map** emitované s kľúčom ***k***
  - výstupom je množina dvojíc **(kľúč, hodnota)**
- Mapper
  - realizuje operáciu **map** nad časťou vstupu
- Reducer
  - realizuje operáciu **reduce** nad časťou dvojíc emitovaných všetkými mappermi

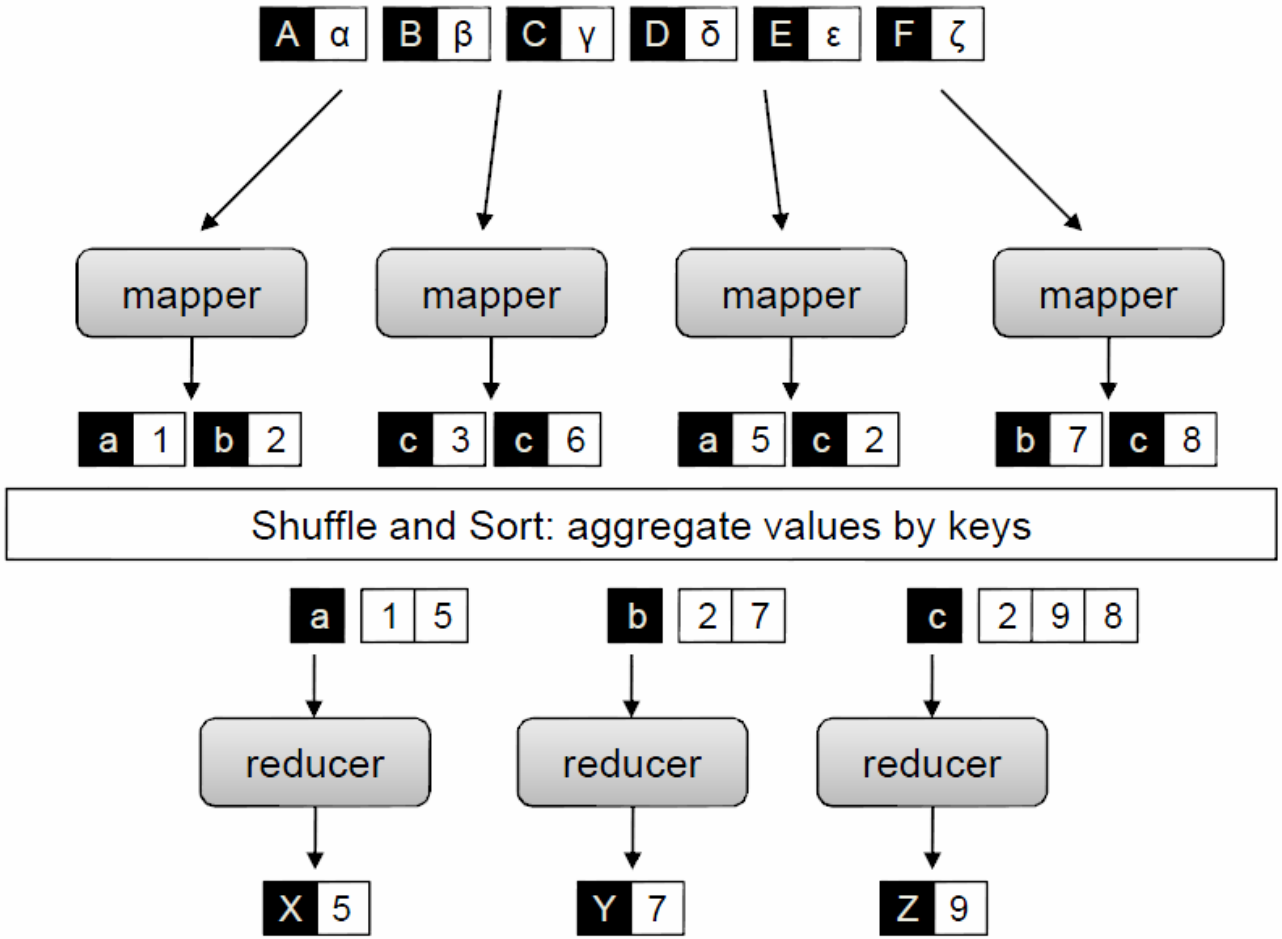
MapReduce - vykonávanie



MapReduce - vykonávanie

- **Mapper**
  - spracúva nejakú časť vstupu
  - počet mapperov je určený systémom
- **Reducer**
  - spracúva jeden alebo viac kľúčov
  - pre každý kľúč realizuje **reduce** nad všetkými hodnotami, ktoré boli emitované s daným kľúčom všetkými mappermi
  - spracúva kľúče vždy v „rastúcom" poradí
  - počet určený job-om, vlastný výstupný súbor

MapReduce - vykonávanie



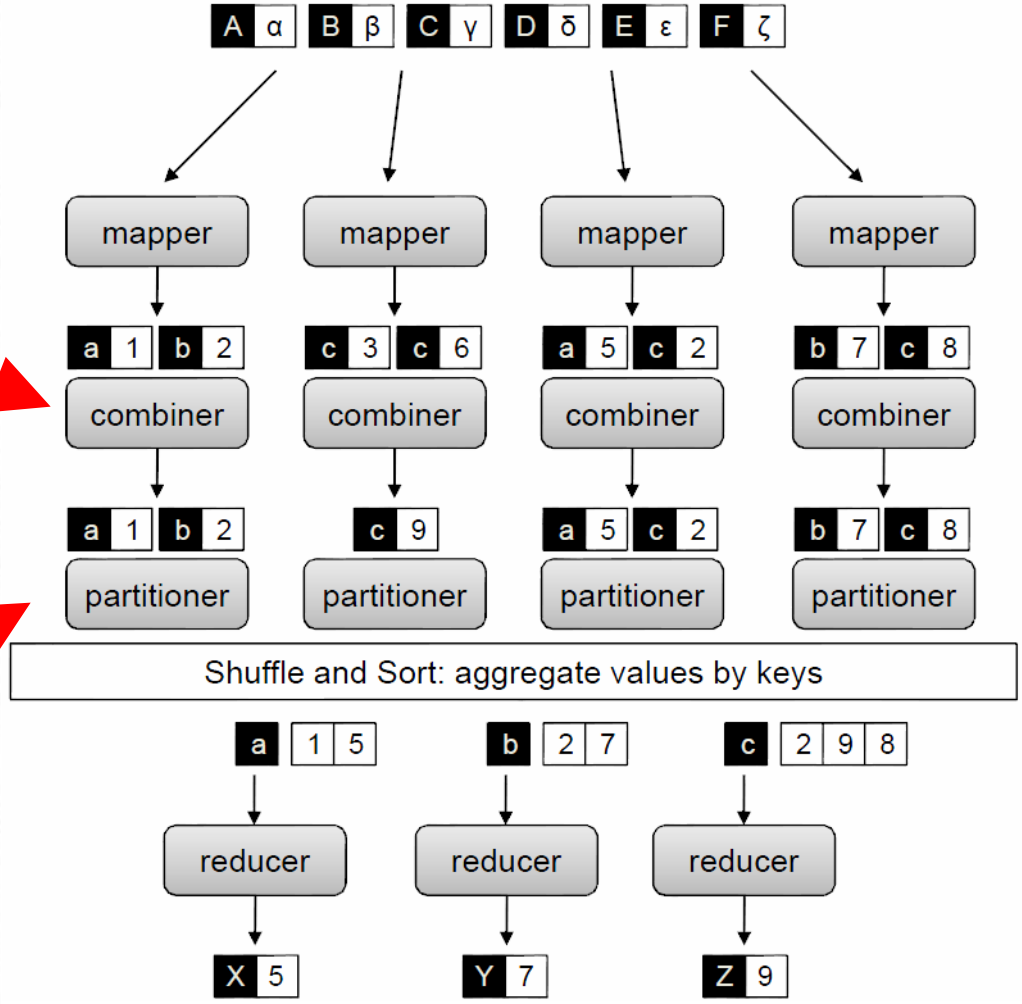
Problémy, otázky

- Ak ten istý mapper vygeneruje viacero dvojíc s rovnakou hodnotou, nemôžeme ich čiastočne „zredukovať" lokálne u mappera a ušetriť tak čiastočne komunikáciu?
  - v prípade asociatívnej operácie
- Počet reducerov je určený fixne. Viem určiť, ktorý reducer bude redukovať konkrétny kľúč a tak napríklad využiť výsledky redukcie viacerých kľúčov?
  - napr. využitie nerovnomerného rozdelenia kľúčov

Combiner a Partitioner

Lokálne „malé“ redukcie

Rozhodnutie o tom, na ktorý reducer bude kľúč (dvojica) poslaný.



Príklad - počítanie slov

- Vstupom sú textové súbory vo forme:

  (offset v súbore, riadok)

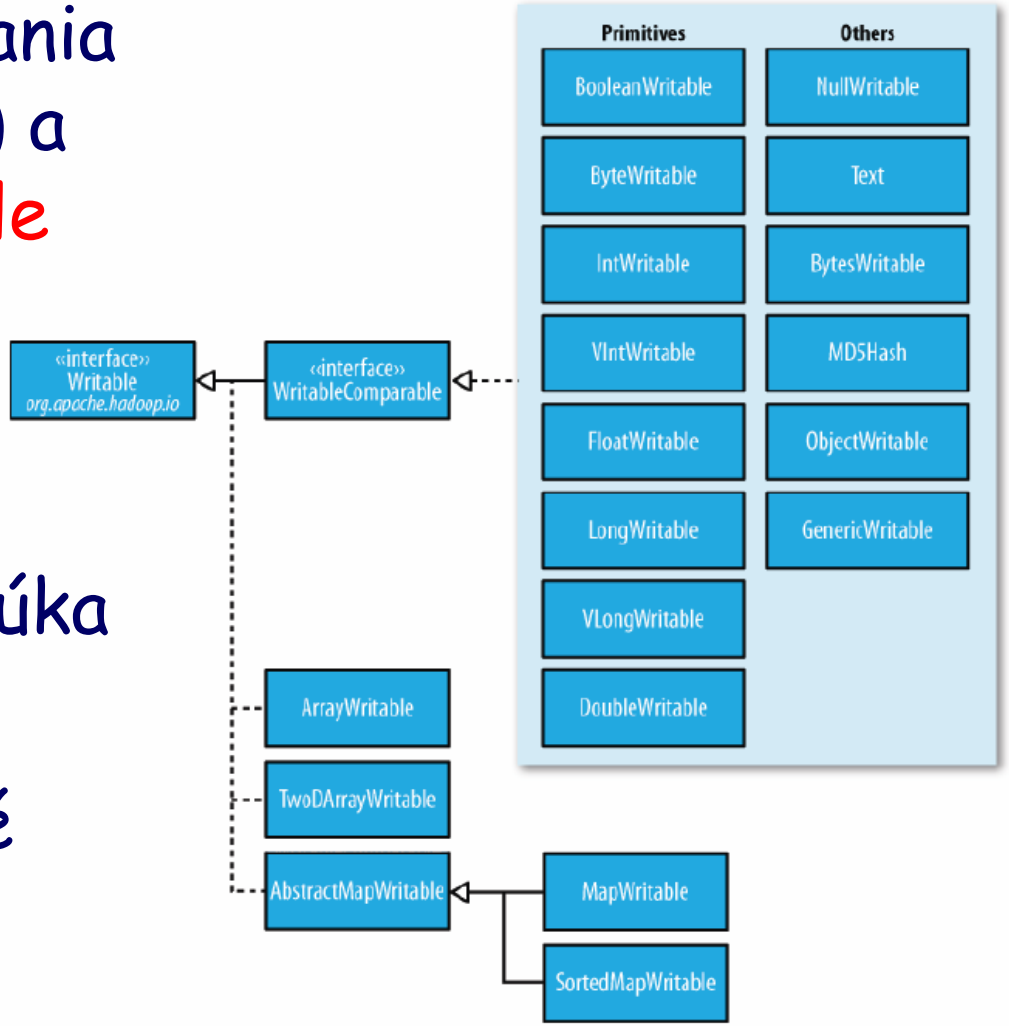
- Pre každé slovo chceme spočítať počet výskytov vo forme:

  (slovo, počet výskytov)

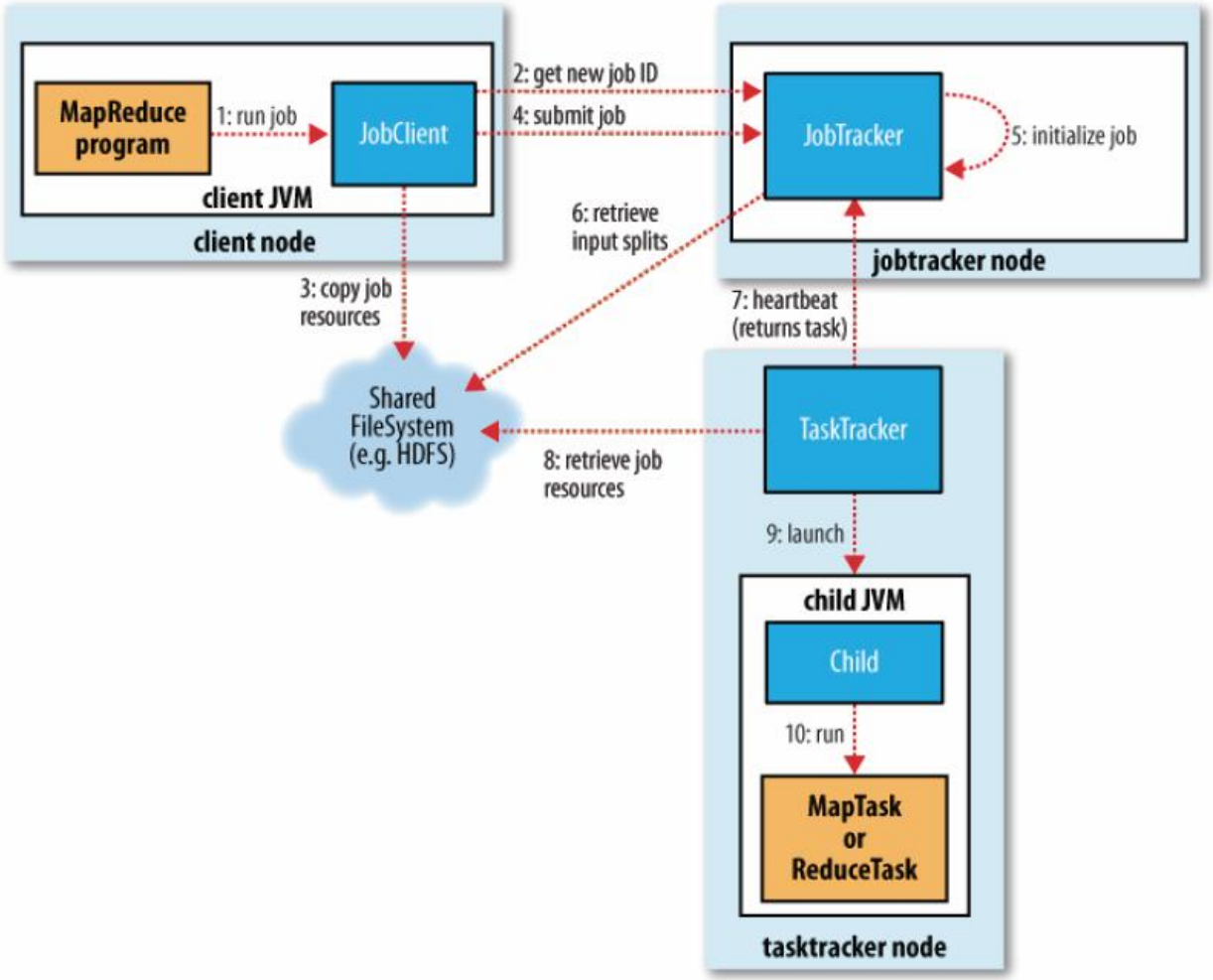
Ako na to v Hadoop-e?

Dátové typy v Hadoop

- Základom sú rozhrania Writable (hodnoty) a WritableComparable (kľúče)
- Využívajú sa kvôli efektívnejšej serializácii než ponúka Java
- Sú modifikovateľné narozdiel od wrap. tried



Job vs. task


MapReduce program
1: run job
JobClient
client JVM
client node
2: get new job ID
4: submit job
JobTracker
5: initialize job
jobtracker node
6: retrieve input splits
3: copy job resources
7: heartbeat (returns task)
Shared FileSystem (e.g. HDFS)
8: retrieve job resources
TaskTracker
9: launch
child JVM
Child
10: run
MapTask or ReduceTask
tasktracker node

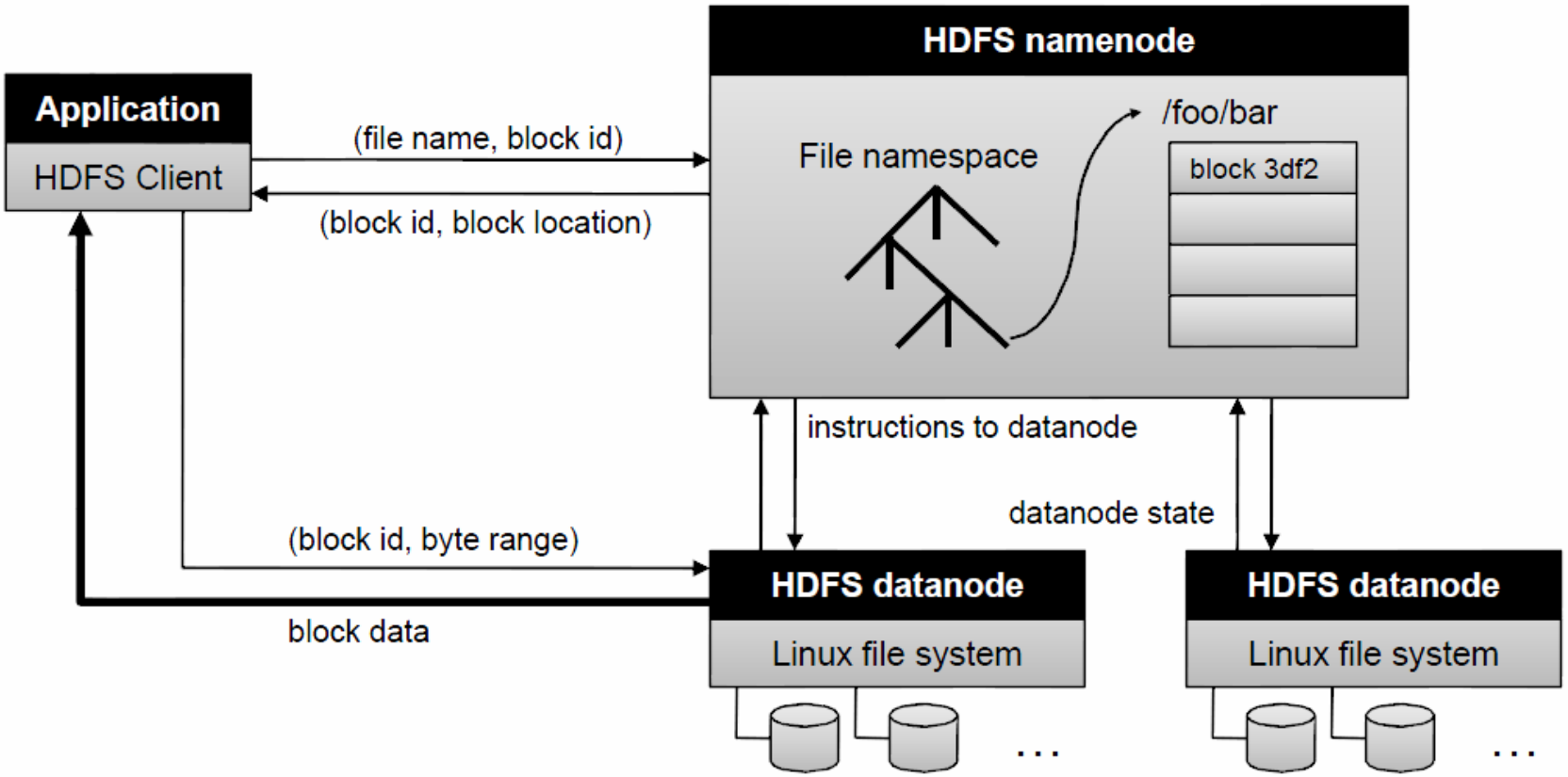
HDFS alebo odkiaľ sú dáta

- Predpokladajú sa vstupné súbory veľkosti niekoľko GB
  - Ako doručiť údaje k taskom?
  - Ako zabezpečiť odolnosť voči chybám?
  - Ako garantovať efektívnosť spracovávania?

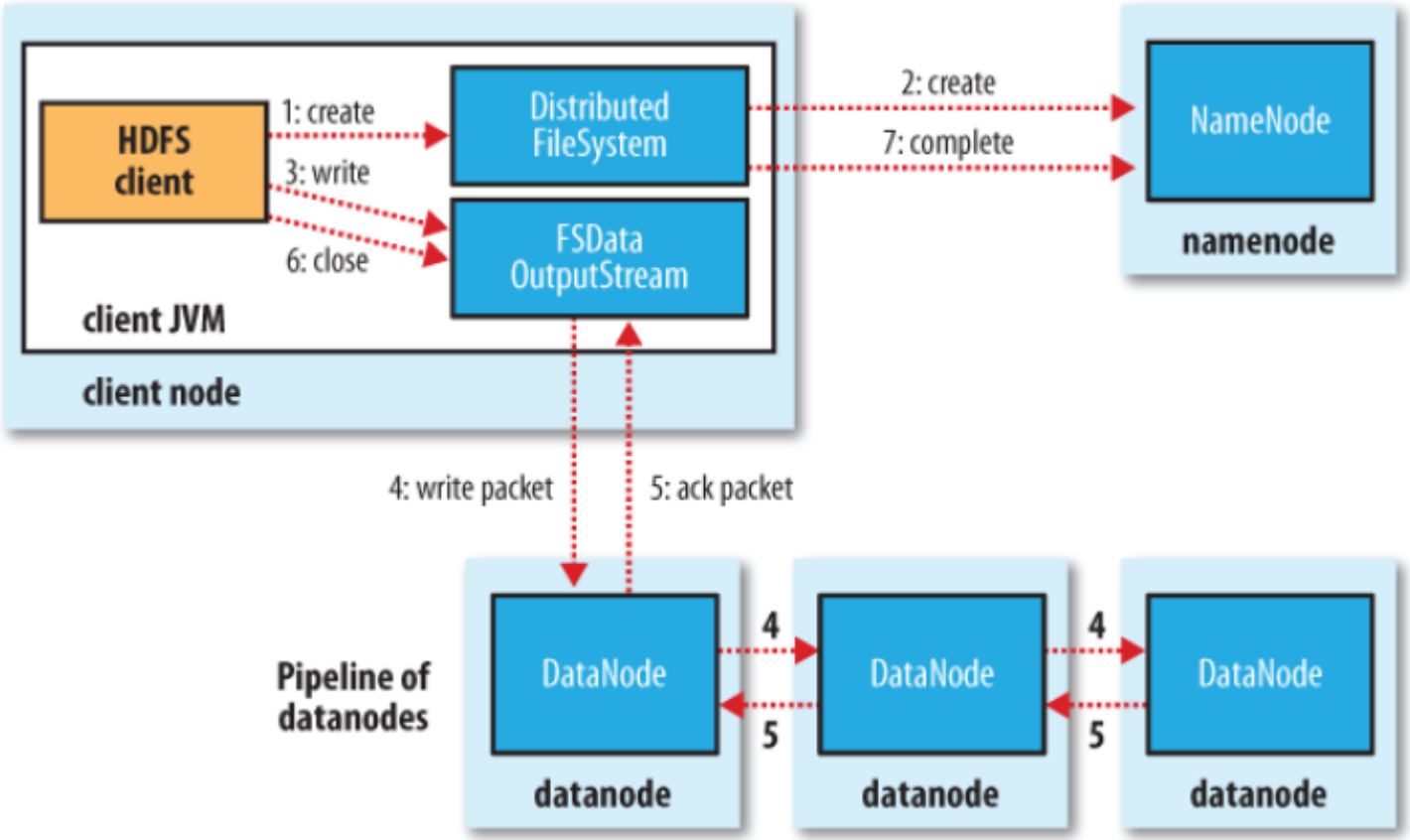
HDFS

- HDFS je distribuovaný súborový systém
- Súbory sú rozdelené do blokov, štandardná veľkosť bloku je **64MB** (resp. 128MB)
- Bloky sú **replikované** na viacerých uzloch clustra
- Namenode
  - len jeden pre cluster, udržiava informácie o súboroch a umiestnení blokov (metadáta)
- Datanode
  - obsahuje bloky jednotlivých súborov

HDFS - schéma


HDFS namenode
Application
HDFS Client
(file name, block id)
(block id, block location)
File namespace
/foo/bar
block 3df2
instructions to datanode
datanode state
(block id, byte range)
block data
HDFS datanode
Linux file system
HDFS datanode
Linux file system

Replikovanie blokov



HDFS

- HDFS manažuje umiestňovanie blokov tak, aby sa **optimalizoval prístup** (komunikácia v racku je rýchlejšia než medzi rackmi – princíp lokality) a zároveň **fault-tolerance** (ak padne rack, dáta sú dostupné)
- **Garbage collection**
- **Load-balancing**
  - optimalizované voľba datanodu, z ktorého sa dáta v bloku čítajú, resp. do ktorého sa dáta zapisujú

HDFS a MapReduce

- **TaskTracker** a **DataNode** sú na tom istom uzle clustra
- **Tasky idú za dátami**:
  - mapper tasky sa spúšťajú na tých uzloch, ktoré obsahujú bloky súboru, ktoré majú spracovať
  - zvyčajne jeden mapper task sa aplikuje na 1 blok súboru

MapReduce a fault-tolerance

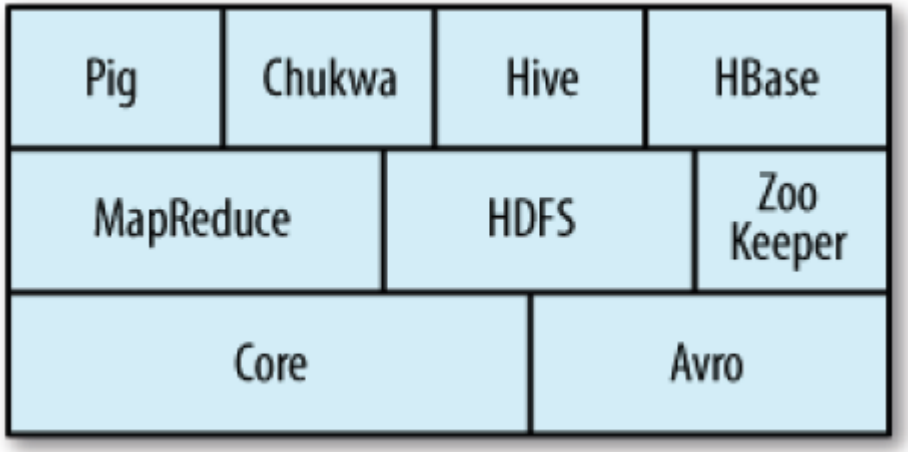
- Ak task zlyhá, spustí sa na inom uzle
- Špekulatívne vykonávanie
  - ten istý task beží na dvoch uzloch, zoberú sa výsledky rýchlejšieho
- Ak task viac krát zlyhá na tých istých záznamoch, tieto záznamy sa preskočia (chyby v aplikáciách tretích strán)
- JobTracker pravidelne **monitoruje** TaskTrackerov (analogicky aj v HDFS)

MapReduce - sumarizácia

- MapReduce je vhodný pre istý typ úloh pracujúcich **s veľkými dátami** (nie je vhodný na všetko)
  - overené v Google
- Programátor sa **koncentruje na problém** (map, reduce, ...), o zvyšok sa stará systém
- Optimalizované na clustre tvorené **bežným hardvérom** (dôraz na fault-tolerance)

Hadoop a jeho aplikácie

- Hbase
  - ekvivalent BigTable
  - distribuovaná stĺpcovo-orientovaná databáza
- ZooKeeper
  - distribuovaná koordinačná služba (distribuované zámky)
- Hive
  - distribuovaný dátový sklad
- Nasadenie Hadoop: Facebook, Last.fm, ...



Ďakujem za pozornosť !