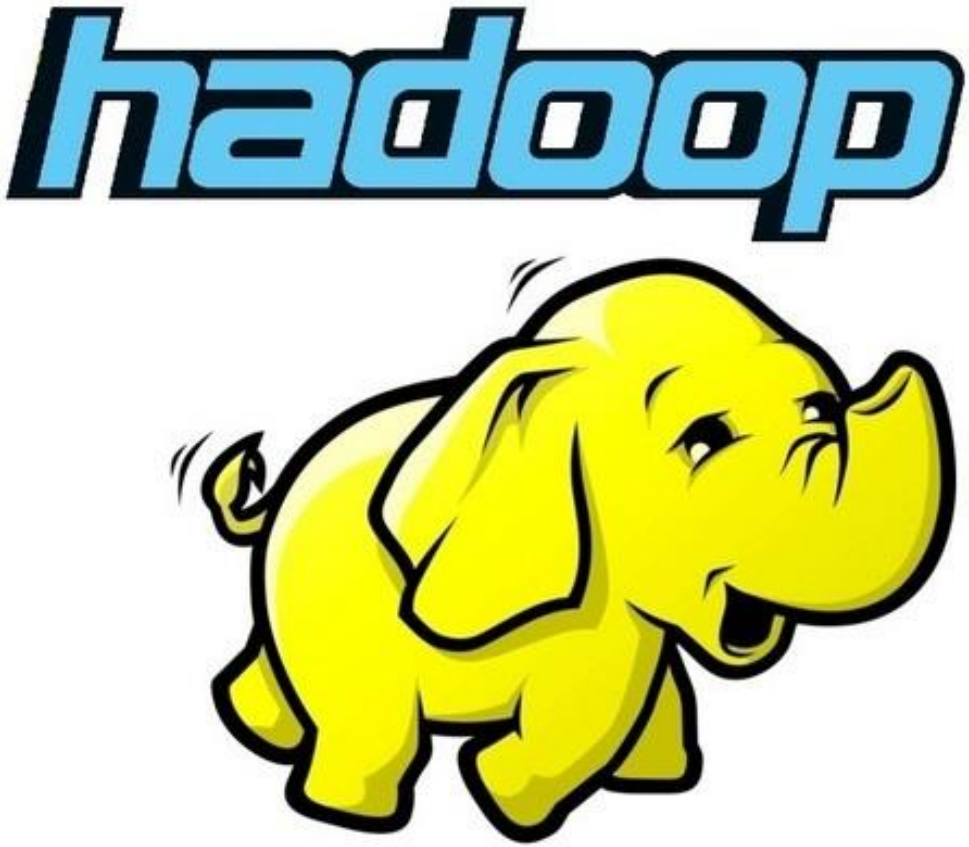
TSVD 7

Technológie spracovania veľkých dát

Peter Bednár, Martin Sarnovský

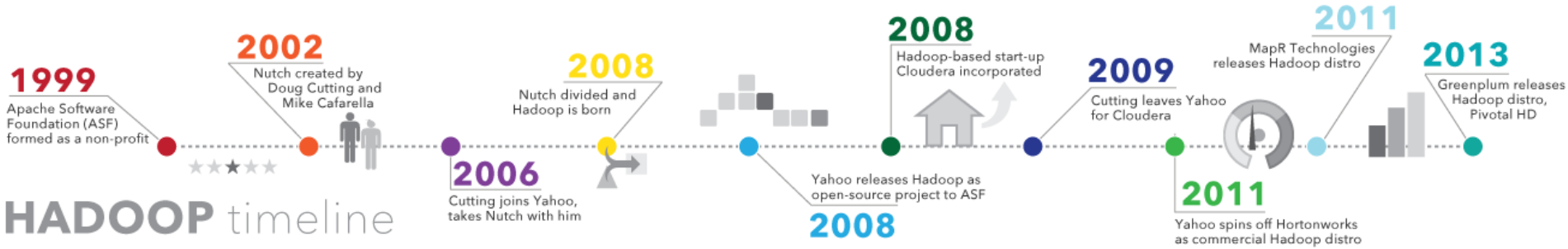

hadoop

Hadoop

- Framework pre vývoj a spúšťanie aplikácií na veľkých klastroch postavených na tzv. „commodity hardware“
- Určený pre spracovávanie veľkých dát (Petabyty) na klastroch pozostávajúcich z tisícok uzlov
- Základné komponenty:
  - Úložisko: HDFS
  - Spracovanie dát: MapReduce
- Požiadavky
  - Náklady – minimalizácia vďaka použitiu uvedeného HW
  - Jednoduchosť použitia
    - Používatelia nemusia poznať prostriedky distribuovaného počítania
  - Spoľahlivosť: fault tolerance

Hadoop - vlastnosti

- Commodity Hardware
  - Schopnosť postaviť klaster a rozširovať ho, použitím štandardných, nie drahých serverov
  - Predpokladá sa, že servery a diskové úložiska nebudú vysoko dostupné a spoľahlivé
- Využitie replikácie dát v klastri
- Separovanie metadát od dát
  - Server špecificky pre správu metadát (Namenode)
  - Servery, ktoré dáta ukladajú (Datanode)
- Presunutie výpočtov smerom k dátam
  - Servery v klastri slúžia na 2 účely: ako úložisko dát a ako výpočtové uzly
  - Single ‘storage + compute’ cluster vs. Separate clusters

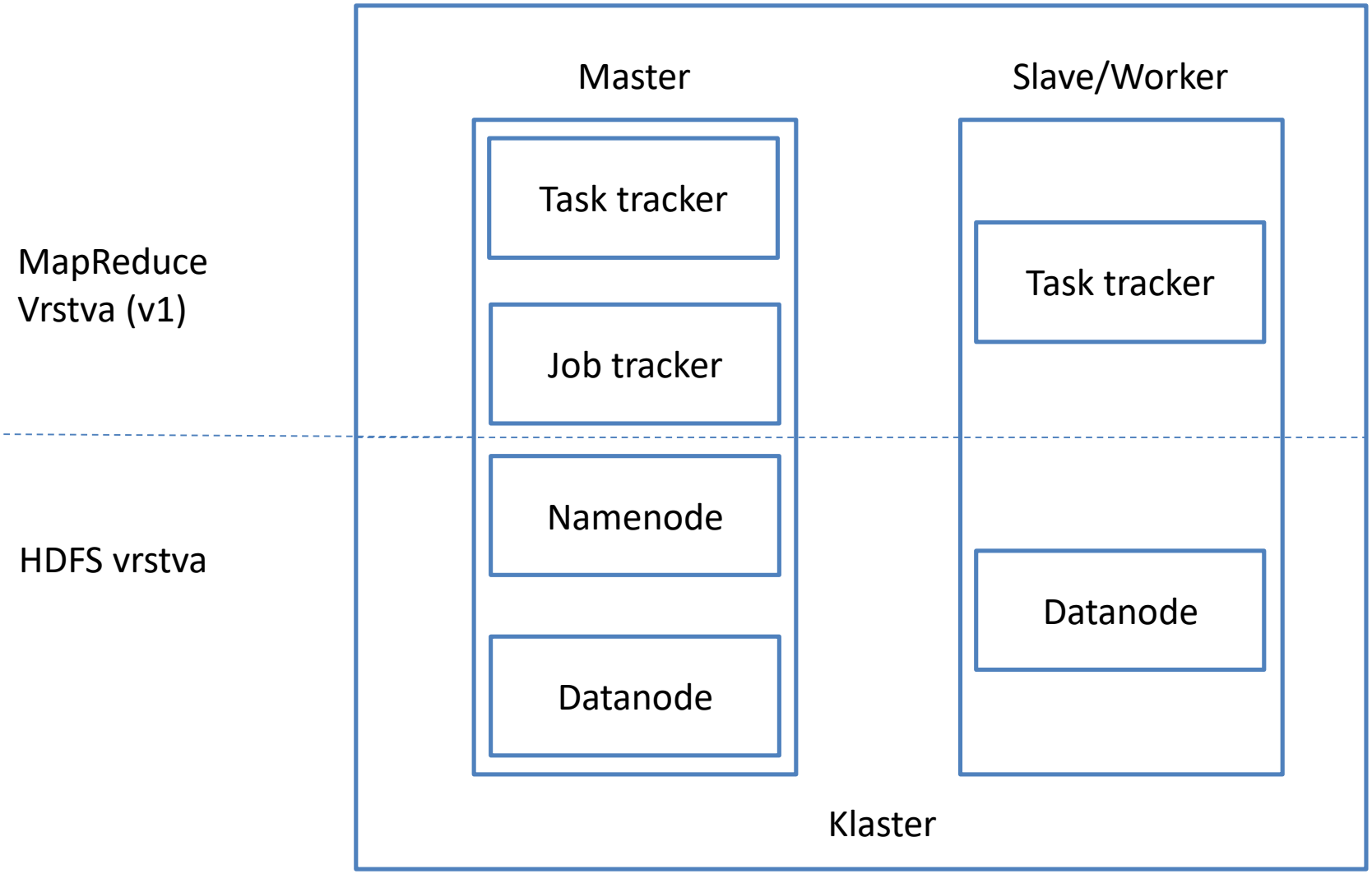


- Vývoj Lucene a Nutch (D. Cutting, M. Cafarella)
- Inšpiráca Google
  - GFS – Google File System (2003)
  - Článok „MapReduce: Simplified Data Processing on Large Clusters“ (Jeffrey Dean, Sanjay Ghemawat) – 2004
- 2006 – Nutch sa transformuje na Hadoop, v apríli release 0.1.0. verzie
- 2007 – Yahoo Hadoop cluster (1000 počítačov)
- 2008 – cca 20 spoločností používa Hadoop
- 2009 – Yahoo! – 17 klastrov s cca 24,000 zariadeniami

Hadoop – základné komponenty

- HDFS – Hadoop distributed file system
- Hadoop Core
  - Množina spoločných utilít a modulov spoločné pre všetky verzie Hadoopu
- Hadoop MapReduce
  - Programovací model pre distribuované aplikácie
- YARN – Yet another resource negotiator

Hadoop “High-level” architektúra



Hadoop klastre - hardware

- Hadoop klastre pozostávajú z hardware dvoch rozdielnych typov:
  - Master
  - Slave/worker
- Master – uzol obvykle viac robustný a odolnejší voči hardvérovým zlyhaniam, obvykle na ňom beží viacero kritických služieb/rolí Hadoop platformy
- Worker – pre tieto uzly sa naopak očakáva, že zlyhávajú pravidelne, tzn. konfigurácia menej robustná

Klastre – konfigurácia v praxi

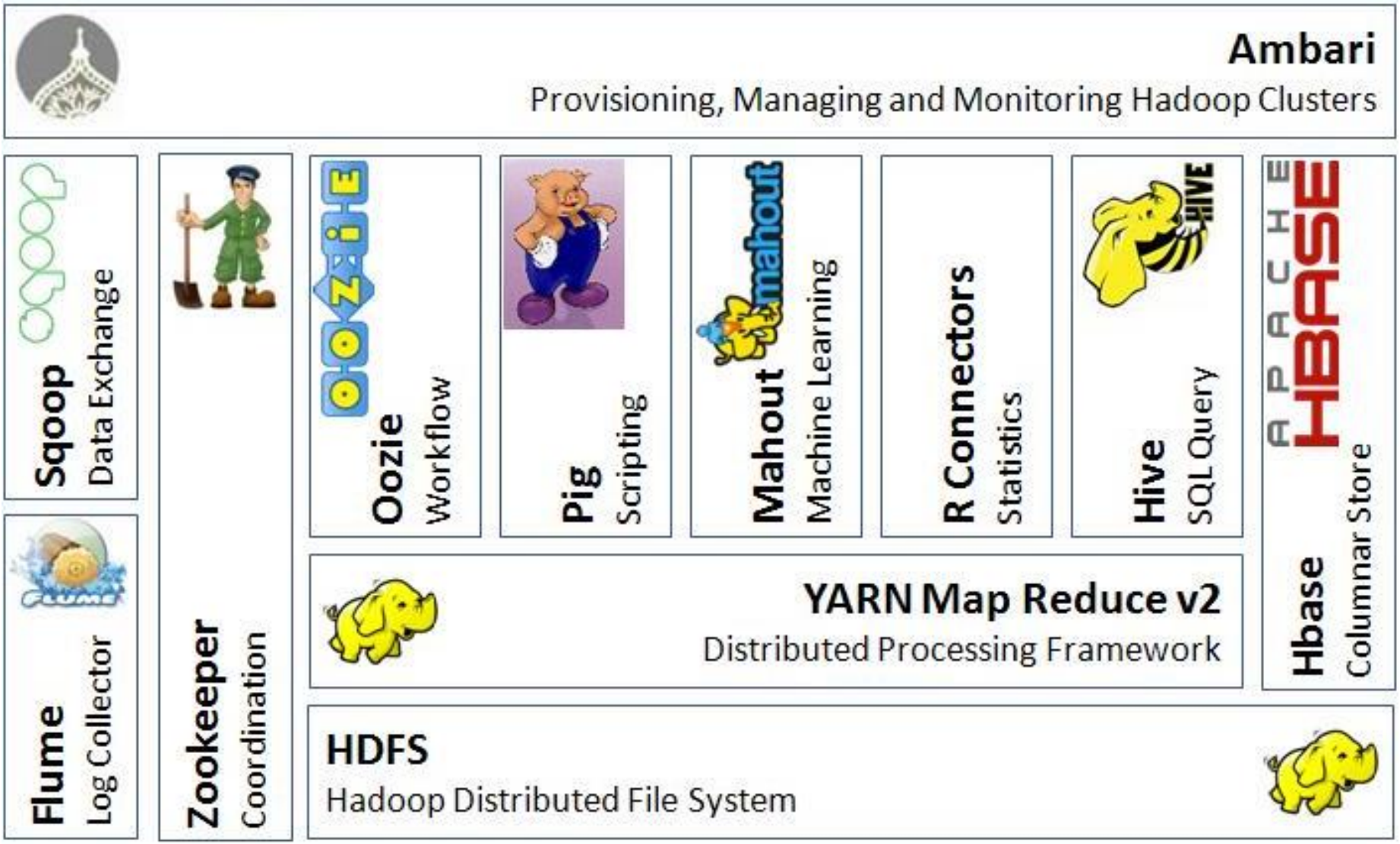
- Malé klastre – s menej ako 20 worker nodes - Baseline - 2 x 6 core 2.6 Ghz CPU, 24 GB RAM, dual 10 Gb Ethernet
- Klastre do 300 uzlov - mid-size kategória, štandardne cca 24 - 48 GB RAM
- Veľké klastre – viac ako 300 uzlov – minimum 96 GB RAM pre mastra
- Worker – 2 x 6 core 2,9GHz CPU, 64/96 GB RAM, 12x3 TB HDD,
- Úložisko – v závislosti na úlohe/spracovávaných dátach
  - Nutné vziať do úvahy replikáciu dát, produkovanie medzivýsledkov a iné, ktoré štandardne zaberú 20-30% dátovej kapacity

Distribúcie Hadoop

- Apache
  - "čistý" Hadoop
- Cloudera
  - 1. spoločnosť poskytujúca distribúciu Apache Hadoop, rozširuje o Cloudera Management Suite, automatizuje inštaláciu, ponúka real-time manažment, najširšia používateľská základňa
- Hortonworks
  - Inovácie (YARN), poskytovateľ pre podnikovú sféru, kompatibilta aj s Windows, Microsoft používa túto distribúciu a „zabaľuje" ju ako MS Azure službu HDInsight
- MapR
  - Ponúka Hadoop ako komponent integrovaný do Ubuntu OS, viaceré obmedzenia vo voľných verziách


cloudera
Fact: Hadoop Isn't Just Hadoop Anymore
The stack is continually evolving and growing!
Core Hadoop (HDFS, MR)
2006
HBase
ZooKeeper
Core Hadoop
2008
Hive
Pig
Mahout
HBase
ZooKeeper
Core Hadoop
2009
Sqoop
Whirr
Avro
Hive
Pig
Mahout
HBase
ZooKeeper
Core Hadoop
2010
Flume
Bigtop
Oozie
MRUnit
HCatalog
Sqoop
Whirr
Avro
Hive
Pig
Mahout
HBase
ZooKeeper
Core Hadoop +YARN
2011
Spark
Impala
Solr
Kafka
Flume
Bigtop
Oozie
MRUnit
HCatalog
Sqoop
Whirr
Avro
Hive
Pig
Mahout
HBase
ZooKeeper
Core Hadoop +YARN
2012
Parquet
Sentry
Spark
Impala
Solr
Kafka
Flume
Bigtop
Oozie
MRUnit
HCatalog
Sqoop
Whirr
Avro
Hive
Pig
Mahout
HBase
ZooKeeper
Core Hadoop +YARN
Present

Hadoop ekosystém




HDFS

HDFS

- Hadoop Distributed File System
- Postavený na GFS (Google File System)
- Veľký, distribuovaný súborový systém
  - Schopný bežať na 10K uzloch, s 100 mil. súbormi, spracovať 10 PB dát
- Poskytuje veľmi dobrú podporu z hľadiska manažmentu klastrov
  - Load balancing
  - Odolný voči zlyhaniam (replikáciou dát)
  - Ľahko rozšíriteľný (pridaním nových uzlov)
- Optimalizovaný pre dávkové spracovávanie dát
  - Umožňuje presunúť výpočet bližšie k dátam

HDFS II.

- Jeden Namespace (menný priestor) pre celý klaster
- Koherentnosť dát
  – HDFS používa *Write-once-read-many* model prístupu k dátam
  – Klient teda nemôže prepisovať súbory, ale iba zapisovať na koniec
- Súbory nahrávané do HDFS sú rozdelené do blokov
  – Štandardne má jeden blok veľkosť 128 MB
  – Každý blok sa potom replikuje naprieč súborovým systémom, tak, aby jeho kópie existovali na viacerých uzloch

Prístup klienta k dátam

– Klient potom môže nájsť umiestnenie bloku s dátami
– Klient potom pristupuje k dátam priamo z uzla, na ktorom sú uložené

HDFS Architektúra

- DFS Master – Namenode
  - Spravuje Namespace (menný priestor) súborového systému
  - Udržiava mapovanie súborov na jednotlivé bloky a na ich umiestnenie v klastri
  - Spravuje alokáciu a replikáciu blokov
  - Checkpoints namespace and journals namespace changes for reliability
  - Riadi prístup do menného priestoru
- DFS Slave - Datanodes
  - Primárny cieľ z hľadiska HDFS – ukladať bloky dát
  - Klientské aplikácie pristupujú k blokom dát priamo z týchto uzlov
  - Periodicky posiela reporty o stave blokov na Namenode
  - Periodicky kontroluje integritu blokov

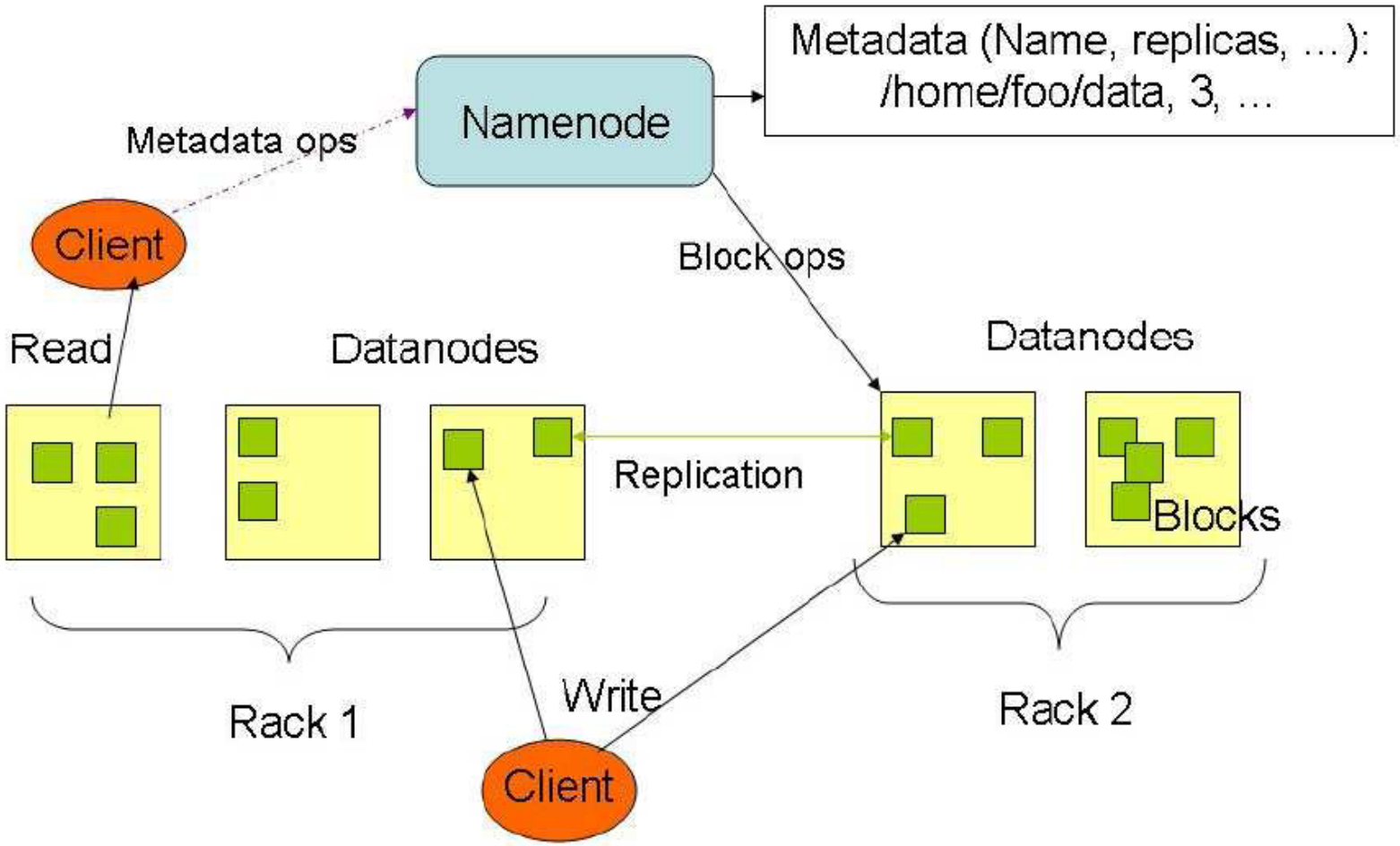
NameNode

- Manažuje Namespace súborového systému
  - Mapuje názov súboru na množinu blokov na ktoré je rozdelený
  - Mapuje blok na Datanode, kde sa nachádza
- Konfiguračný manažment klastra (HDFS) – správa infraštruktúry
- Replikačný engine pre bloky
- Metadáta
  - Všetky metadáta sú uložené v pamäti
  - Neexistuje požiadavka na stránkovanie metadát
- Typy metadát
  - Zoznam súborov
  - Zoznam blokov pre každý súbor
  - Zoznam Datanodov pre každý blok
  - Atribúty súborov, ako napr. čas vytvorenia, faktor replikácie, a iné
- Záznamy transakcií
  - Zaznamenáva vytvorenie súborov, mazanie, atď.

Datanode

- Úložisko pre bloky
  - Ukladá bloky v lokálnom súborovom systéme uzla (napr. ext3)
  - Ukladá metadáta o blokoch (napr. CRC)
  - Poskytuje dáta a metadáta klientom
- Reporting blokov
  - Periodicky posiela report a status všetkých existujúcich blokov na Namenode
- Data Pipelining
  - Forwarduje dáta na ostatné konkrétne Datanody
- Datanody posielajú pravidelný „heartbeat“ hlavnému Namenode
  - Štandardne nastavený heartbeat je každé 3 sekundy
- Namenode potom používa heartbeat ako spôsob pre detekciu zlyhania Datanodov

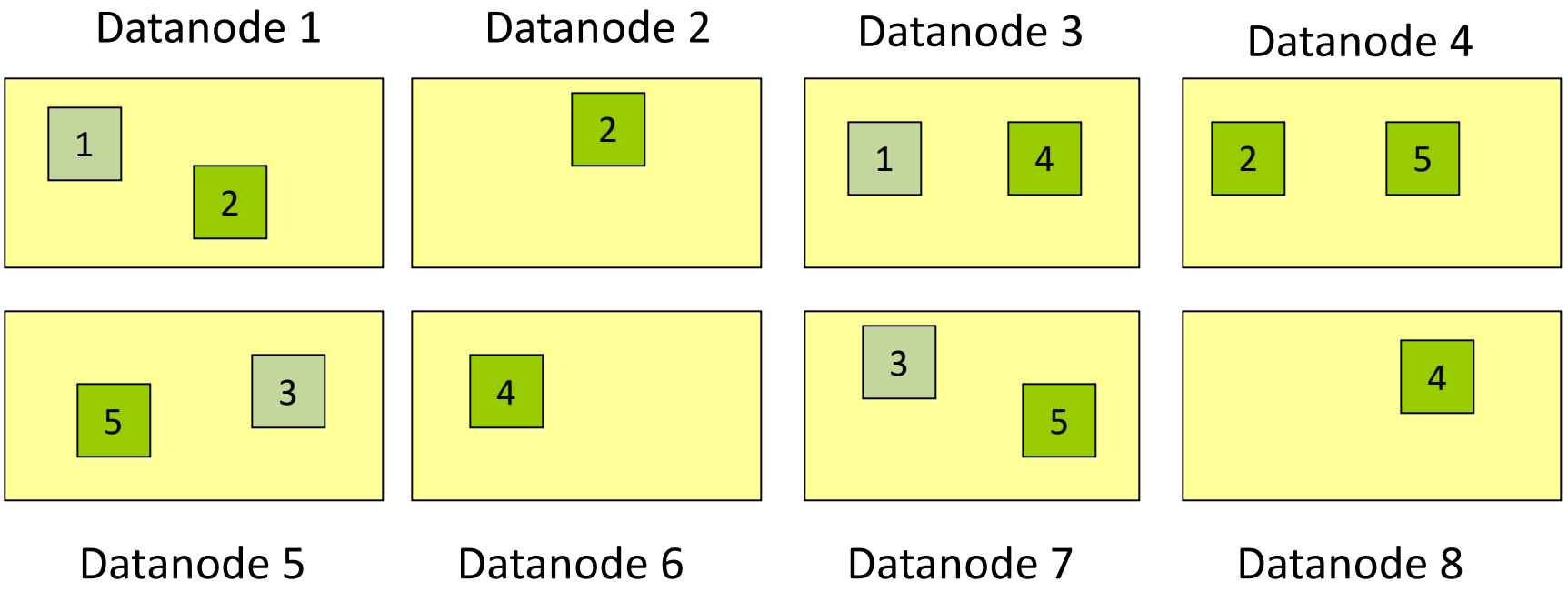
HDFS Architektúra



HDFS Data model

Namenode – súbor, replikácie, block-id

/users/user1/data/part-0, r:2, {1,3}, …
/users/user1/data/part-1, r:3, {2,4,5}, …



Replikácia dát v HDFS

- Faktor replikácie (počet kópií blokov) pre súbor je možné nastaviť, aj rôzne pre rôzne súbory (prednastavená hodnota parametra je 3)
- Umiestnenie blokov - rack aware
  - V prípade, že je klaster na viacerých rackoch, garantuje umiestnenie replík na dvoch rozdielnych rackoch
  - Prvá replika na lokálnom uzle, ďalšie repliky sa kopírujú pokiaľ možno na uzly na inom racku
- Aplikácia číta dáta tak, že nájde a používa „najbližšiu“ repliku
- Pod-replikácia a nad-replikácia (málo, alebo zbytočne veľa replík) sú detegované Namenodeom
- Implementovaný Balancer sa stará o vyvážené využitie Datanodov a ich obsadenie jednotlivými blokmi

HDFS rozhrania

- **Java API**
- **Command Line**
  - hadoop dfs -mkdir /foodir
  - hadoop dfs -cat /foodir/myfile.txt
  - hadoop dfs -rm /foodir myfile.txt
  - hadoop dfsadmin -report
  - hadoop dfsadmin -decommission datanodename
- **Web Interface**
  - HUE, Hortonworks HDP, atď.


MapReduce

Terminológia

- Job – “celý program” – vykonanie/spustenie Mappera a Reducera na celom datasete
- Task *(úloha)* – spustenie/vykonanie Mappera alebo Reducera na časti dát
- Task Attempt (pokus o vykonanie úlohy) – pokus o konkrétne vykonanie úlohy na danom uzle

Terminológia - príklad

- Napr. spustenie „WordCount” na databáze 20 súborov predstavuje jeden *job*
- Na 20 súboroch teda bude vykonaných 20 map tasks *(úloh)* + niekoľko reduce tasks *(úloh)*
- Najmenej 20 map pokusov o vykonanie úloh bude vykonaných - viac, napr. keď dôjde k zlyhaniu

MapReduce

- MapReduce je programovací model, ktorý sa implementuje pre úlohy spracovávajúce/narábajúce s veľkými dátovými množinami
- Programátor špecifikuje *map* funkciu, ktorá spracuje dáta vo forme *kľúč/hodnota* párov a vygeneruje medzivýsledky rovnako vo forme kľúč/hodnota
- *Reduce* funkcia potom agreguje všetky medzivýsledky, ktoré sú asociované s rovnakým kľúčom (v medzivýsledku)
- Map**:** vstupom je súbor dát, výstupom asociatívne pole (prvky nie sú indexované podľa postupnosti celých čísel, ale pomocou kľúčov) typu: kľúč/hodnota
- Reduce: vstupom je výstup z funkcie *map* a výstupom je asociatívne pole typu: kľúč/hodnota, pričom platí, že polia s rovnakými kľúčmi sú spracovávané na jednom uzle

Transformácie párov kľúč/hodnota

- MapReduce sa dá zapísať ako transformácia párov kľúč/hodnota:

{K1,V1} -> {K2, List<V2>} -> {K3,V3}

- Kde *K1* a *V1* sú vstupné páry do *map* funkcie MapReduce úlohy, čítajú sa obvykle z HDFS
- Výstupom z *map* metódy (a teda vstupom do metódy *reduce*) bude množina párov kľúčov a asociovaných zoznamov hodnôt, ktoré sú označené *K2* a *V2*
- Každý *mapper* vyprodukuje výstup vo forme jednotlivých dvojíc kľúč/hodnota, tieto sú potom skombinované do podľa kľúča do formy kľúč/zoznam hodnôt. Túto operáciu vykoná napr. metóda *shuffle*
- Finálny výstup MapReduce úlohy je ďalšia množina párov kľúč/hodnota - *K3*/*V3*, ktoré už predstavujú výsledok úlohy a sú zapísané do HDFS

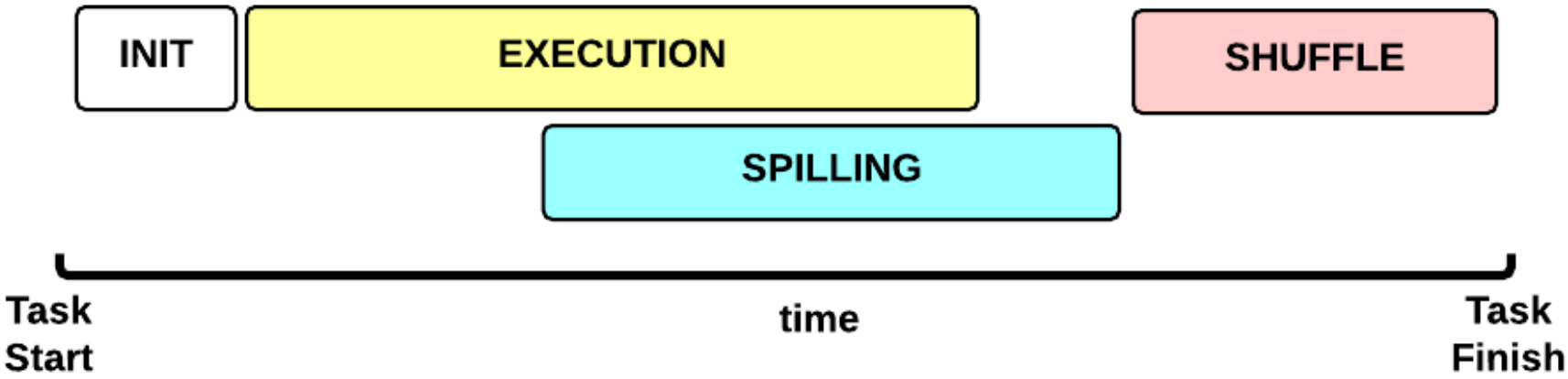
MapReduce job – fázy

- Inicializácia
- Map
- Shuffle/Sort
- Reduce

MapReduce job - inicializácia

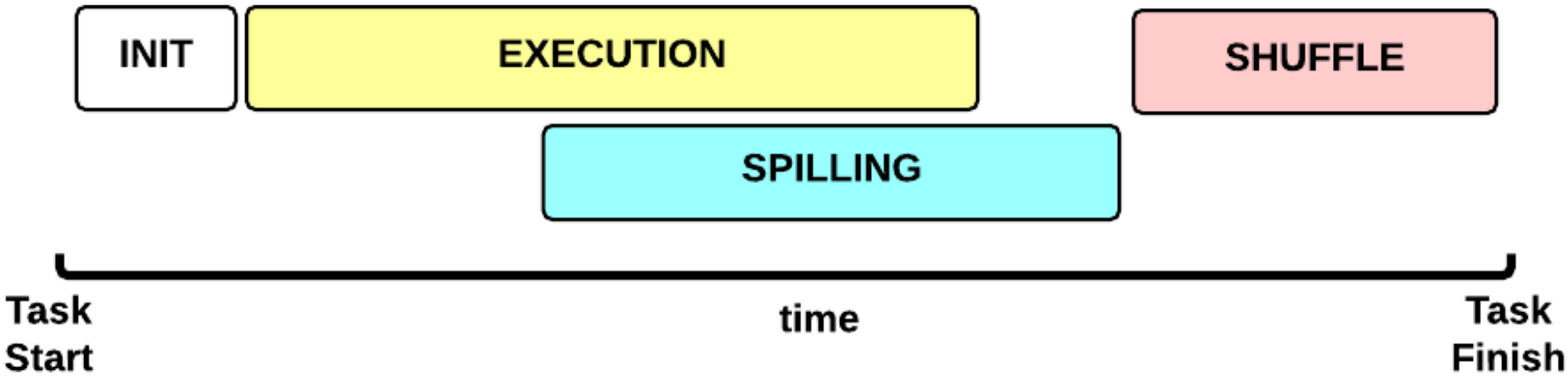
- Inicializácia
  - Hadoop rozdelí vstupné dáta, najčastejšie na podmnožiny o veľkosti HDFS blokov (*splits*)
  - Priradí každý *split* jedému mapperu, snaží sa prideliť *split* na mapper, kde sa daný *split* nachádza
- Map
- Shuffle/Sort
- Reduce

MapReduce job - Map



INIT: Inicializácia map tasku
EXECUTION: Hadoop číta daný split riadok po riadku, volá metódu *map* na každý riadok, spracúva ho ďalej ako dvojicu (kľúč, hodnota), aplikuje logiku *map* metódy a emituje kľúč/hodnotu medzivýsledku
SPILLING: výsledok mappera je uložený v pamäti, keď je buffer skoro plný, zapíše sa na disk (uvoľňuje pamäť) – lokálny, nie HDFS

MapReduce job - Shuffle



SHUFFLE: po zapísaní všetkých dát z mappera spojí všetky výstupy a ”zabalí” ich pre reducer. Po skončení *map* fázy, je možné medzivýsledky zoradiť podľa kľúča (**Sorting**) – triedenie šetrí čas reduceru

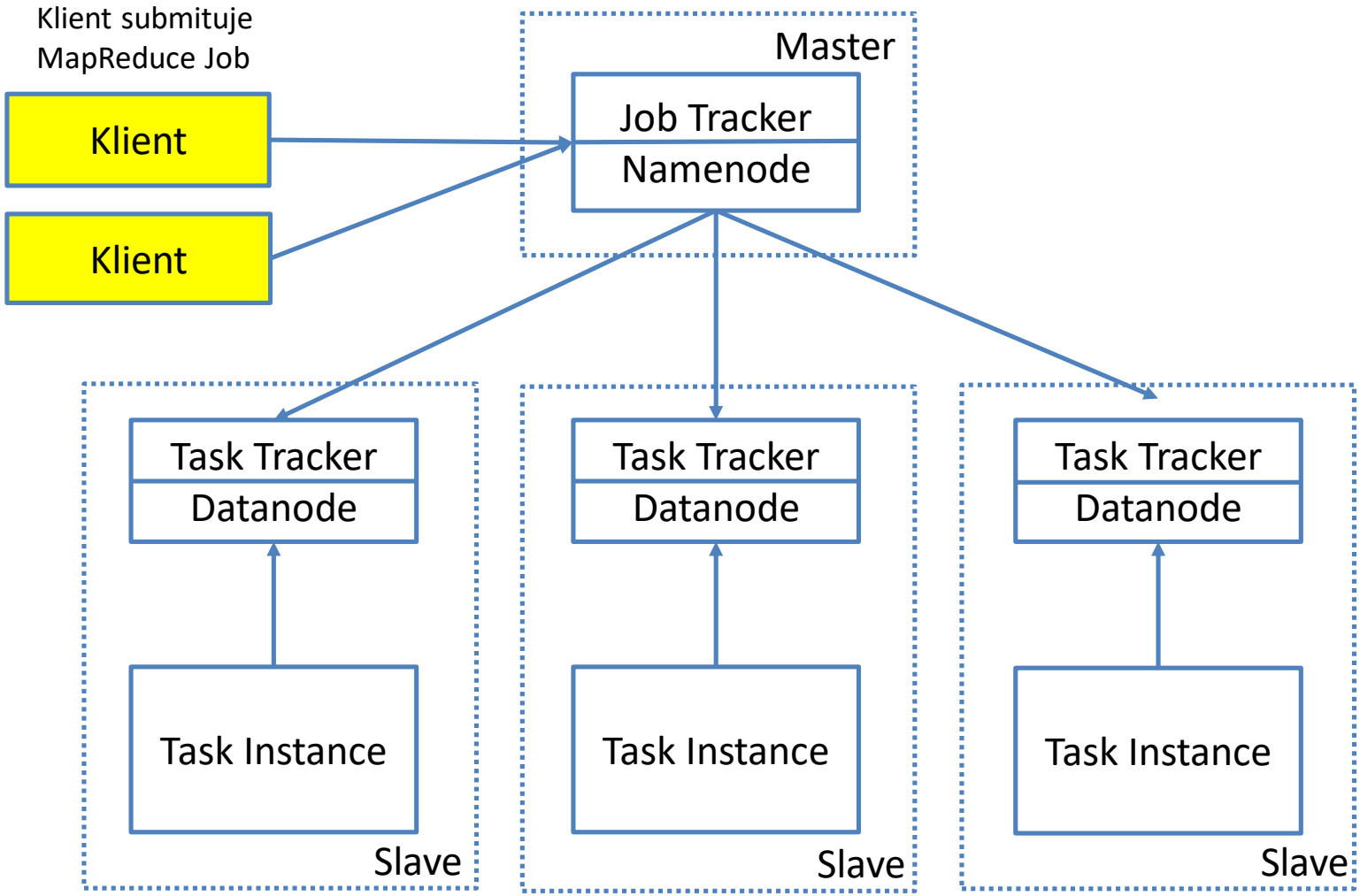
Partitioning - počas shufflingu – ak existuje viacero reducerov, rozdelí zoradené medzivýsledky daného mappera podľa kľúča a rozpošle reducerom

Combiner - tvári sa ako lokálny reducer a agreguje dáta podľa kľúča, kým sú v pamäti

MapReduce job - Reduce

- Lokálne Hadoop číta agregované a rozdelené medzivýsledky
- Aplikuje metódu *reduce* na tieto vstupy
- Reducer aplikuje logiku implementácie *reduce* metódy a vyprodukuje výsledné páry kľúč/hodnota, ktoré zapisuje do výstupného súborového systému (HDFS)
- Na urýchlenie MapReduce – dáta sa hneď presunú z mappera na reducer (aby sa nepresúvali všetky naraz po skončení posledného mappera)
- Speculative execution
  - reduce fáza môže začať až po skončení všetkých map fáz -> niekedy je celý job brzdený jedným mapper taskom, ktorý neprebieha korektne

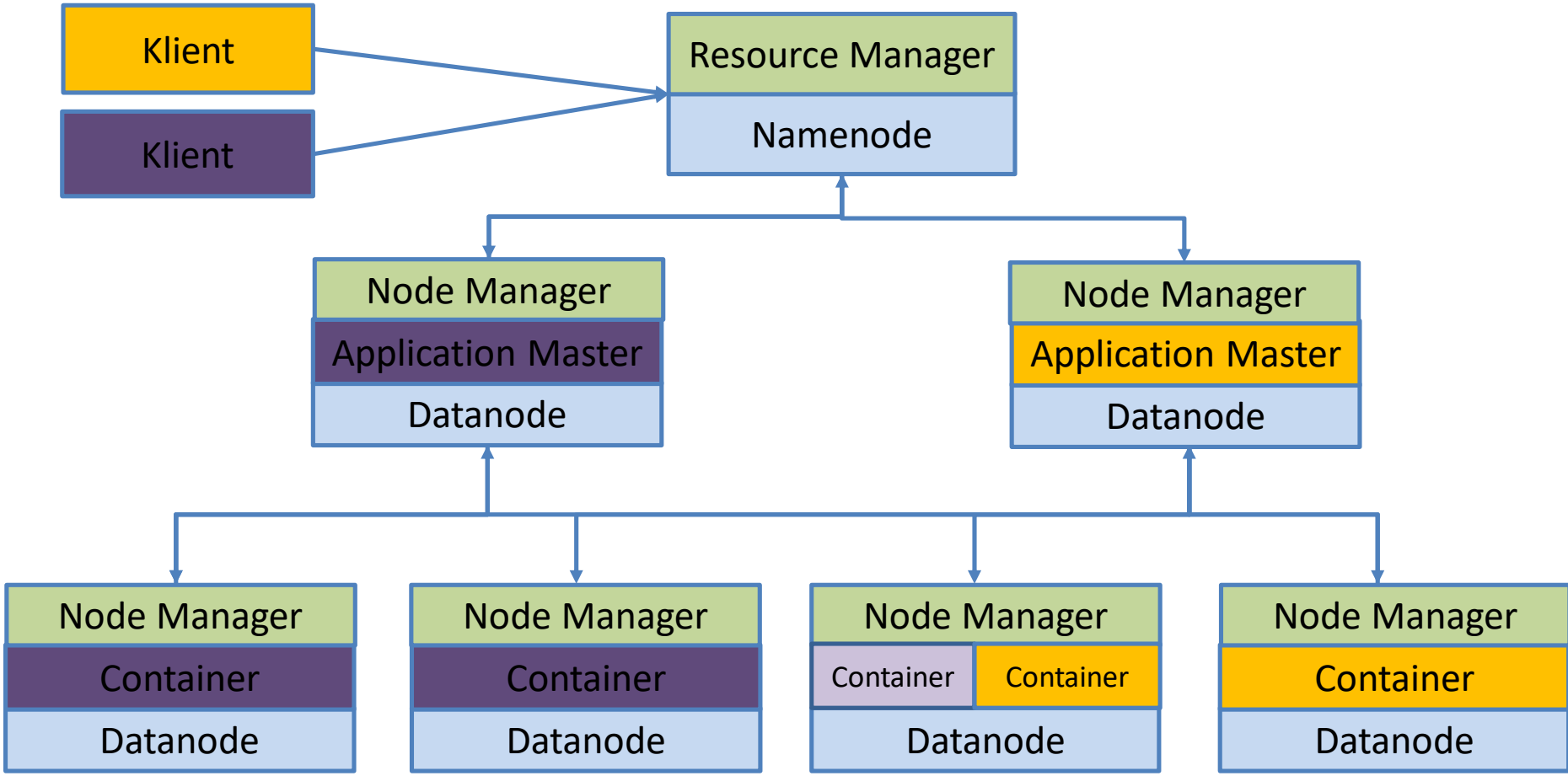
MapReduce v1 architektúra



MapReduce v1

- Previazanie medzi Cluster Resource Management (manažmentom zdrojov) a MapReduce modelom vykonávania
- Obmedzená škálovateľnosť MR1
  - JobTracker spustený na jednom zariadení, určený na viacero úloh
  - Resource management
  - Job a task scheduling
  - Monitoring
- JobTracker – SPOF (Single point of failure)

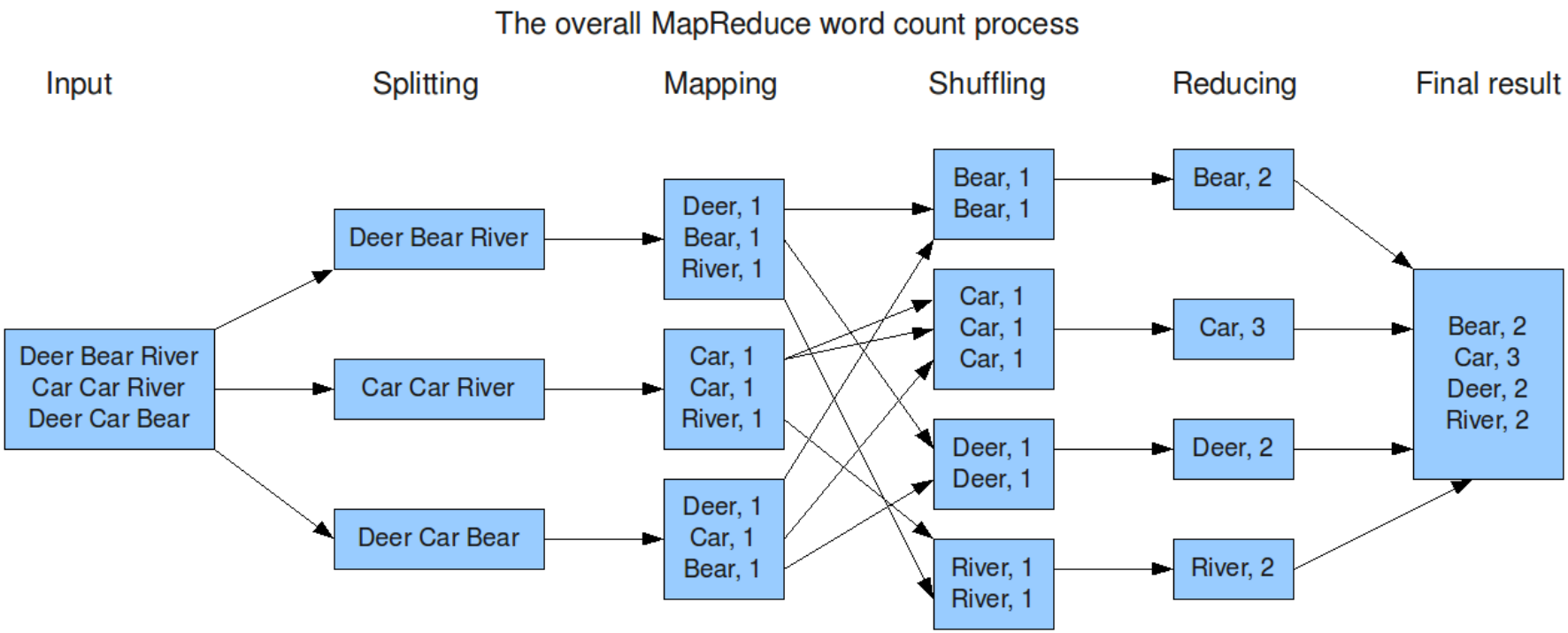
YARN architektúra



MapReduce v2 - YARN

- Traja aktéri v YARN (Yet Another Resource Negotiator):
  - Job Submitter (klient)
  - Resource Manager (master)
  - Node Manager (slave/worker)
- Resource Manager (jeden na klaster) plní rolu mastra
  - Nesie informácie o workeroch a ich umiestnení v rackoch (Rack Awareness) a o zdrojoch, ktorými disponujú
  - Beží na ňom niekoľko služieb, najpodstatnejšia (v porovnaní s MR1) je Resource Scheduler (plánovač zdrojov), ktorý rozhoduje o prideľovaní zdrojov z Datanodov
- Node Manager (viacero na klastri)
  - Predstavuje worker uzol v infraštruktúre,
  - Po naštartovaní a pripojení do klastra sa hlási Resource Managerovi
  - Posiela mu pravidelne heartbeat (identicky ako MR1) a poskytuje zdroje (výpočtové, dátové)
  - Jeho kapacitu špecifikuje množstvo operačnej pamäte a počet jadier
  - Počas behu Resource Scheduler rozhoduje o využívaní tejto kapacity
  - Container je časť kapacity workeru, ktorá je používaná klientom pre beh aplikácie
- Application Master
  - Dohliada nad jobmi (aplikáciami), ak niektoré z úloh pre daný job bežia pomalšie ako je predpokladané, Application Master vygeneruje duplicitné úlohy (ak jedna z duplicitných úloh prebehne, zapíše výsledky na disk a zastaví druhú duplicitnú úlohu)

MapReduce – príklad Wordcount



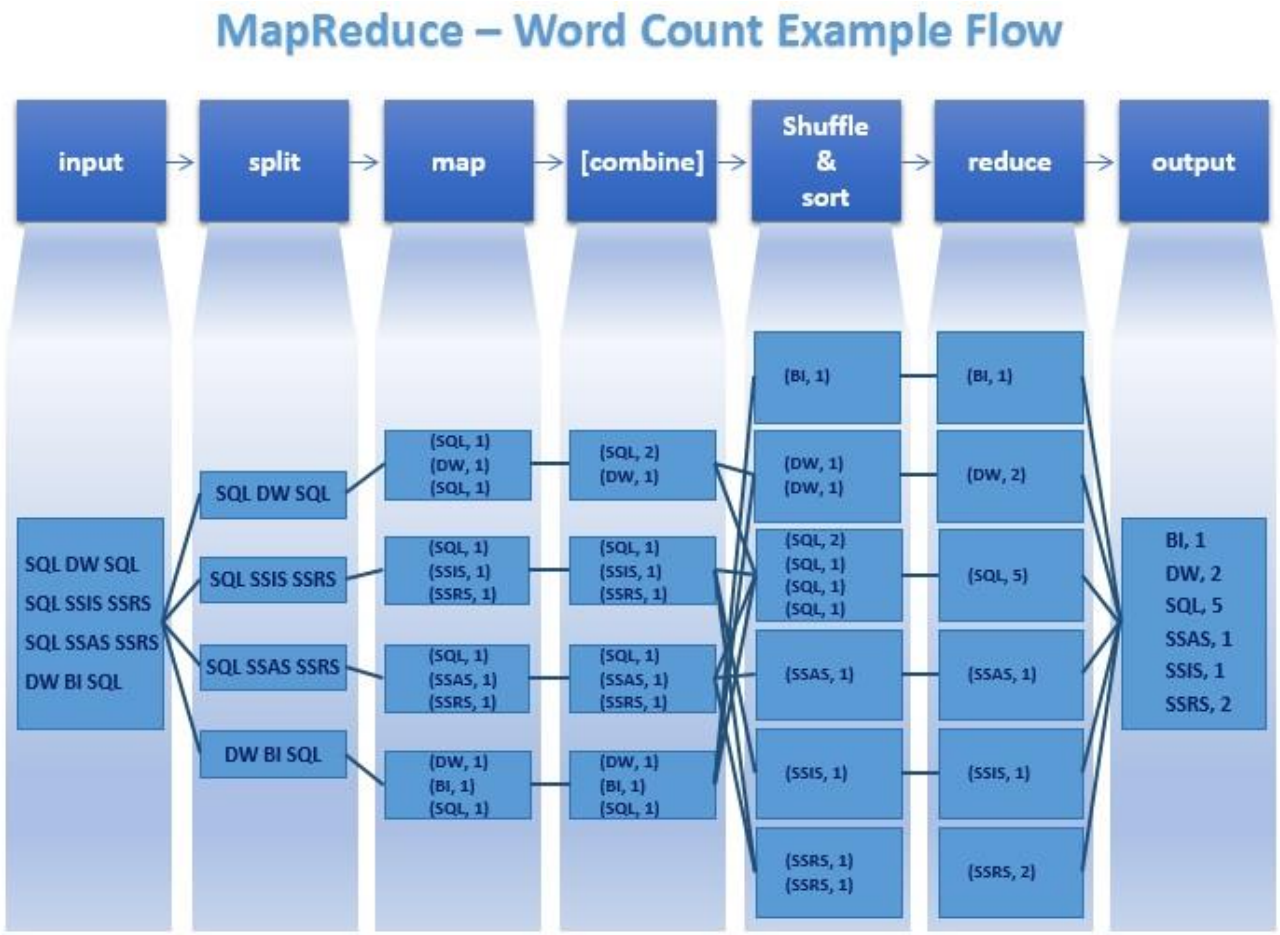
Wordcount - Mapper

```
map(String key, String value)
// key: dokument ID
// value: obsah
for each word w in
   value: EmitIntermediate(w, "1");
```

Wordcount - Reducer

```
reduce(String key, Iterator
values):
// key: slovo
// values: zoznam výskytov slova
int result = 0;
for each v in values:
    result += ParseInt(v);
  Emit(AsString(result));
```

MapReduce – príklad Wordcout II



MapReduce – príklad distrib. dotazu

- SELECT * WHERE id % n = s – map funkcia (s je číslo výpočtového uzla 0..n - 1)
- SELECT položka, sum(počet) GROUP BY položka SORT BY položka – reduce funkcia

MapReduce - príklad