TSVD 10

Technológie spracovania veľkých dát

Peter Bednár, Martin Sarnovský

Obsah

- Distribuované metódy strojového učenia
- Spôsoby distribúcie/paralelizácie modelov
- Distribuované klasifikačné modely
  - Stromový klasifikátor (+ bagging, Random Forest klasifikátor)
- Distribuované zhlukovacie modely
  - k-Means
- Distribuované gradientové metódy
  - Support vector machine
- Knižnica Mahout
- Knižnica MLlib

Distribuované strojové učenie

- Rovnaké problémy ako pri spracovaní veľkých dát (veľkosť dát, náročnosť na spracovanie štandardnými prostriedkami), aplikovaná na oblasť modelovania
- Dáta v distribuovanom prostredí
  - dáta príliš veľké na uloženie na jednom disku (nutnosť ich distribuovať, napr. na klastri)
  - dáta vznikajú a sú ukladané na rozdielnych miestach (aj geograficky)
- Spracovanie samotných dát
  - použitie distribuovanej architektúry pri tvorbe samotných modelov

Vlastnosti distribuovaných systémov

- Paralelné načítavanie dát
  - Pri načítavaní veľkých množstiev dát z disku
- Odolnosť voči zlyhaniam
  - Využitie vlastností distribuovaných architektúr, vrátane replikácie dát atď.
- Distribuované prostredie – ďalšia vrstva v hierarchii pamäte, náklady na komunikáciu v rámci tohto prostredia
- Loading time vs. Running time
  - Rôzne aspekty náročnosti pri spracovávaní veľkých dát
  - Ak dominuje čas samotnej konštrukcie modelu, väčší dôraz klásť na distribuovanie/paralelizáciu operácií
  - Ak dominuje prístup k dátam/načítavanie dát – redukovať počet prístupov k dátam

Spôsoby distribúcie/paralelizmu modelov

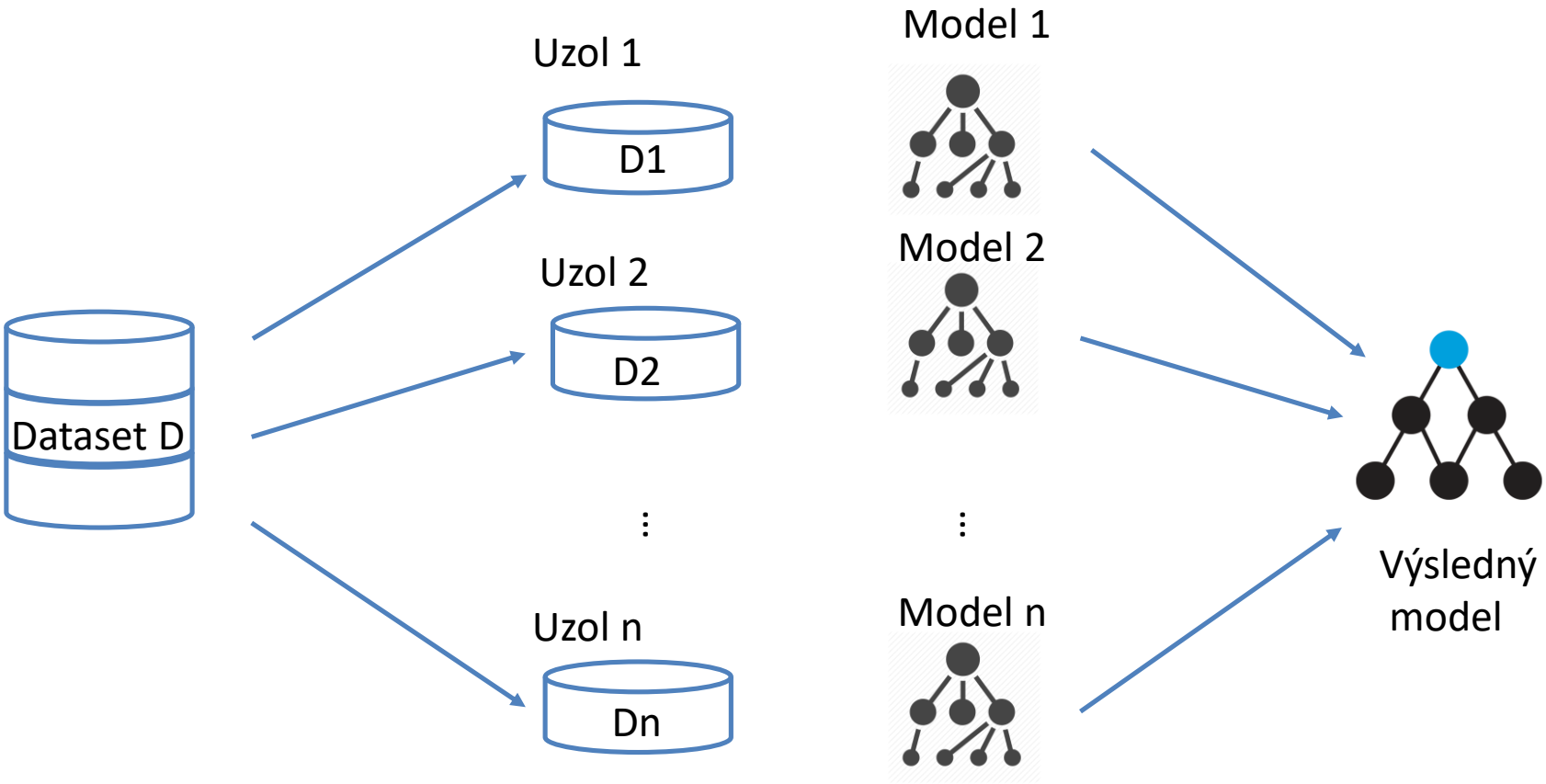
- Založená na dátach
  - Dátová množina rozdelená naprieč prostredím (pracovné uzly)

- Založená na modeli
  - Paralelizuje/distribuuje sa samotná tvorba modelu
- Možné kombinovanie oboch prístupov – tvorba hybridných techník distribúcie modelov strojového učenia

Distribúcia založená na dátach

- Dátová množina sa rozdelí medzi výpočtové kapacity (uzly)
- Tzv. „additive updates“, techniky pre paralelizáciu výpočtov rozdelením dátovej množiny sú napr. micro-batches
- Kľúč k tvorbe valídnych modelov týmto spôsobom je rozdelenie dát na navzájom nezávislé podmnožiny
- Tvorba finálneho modelu – agregácia výsledkov z jednotlivých podmodelov

Distribúcia založená na dátach

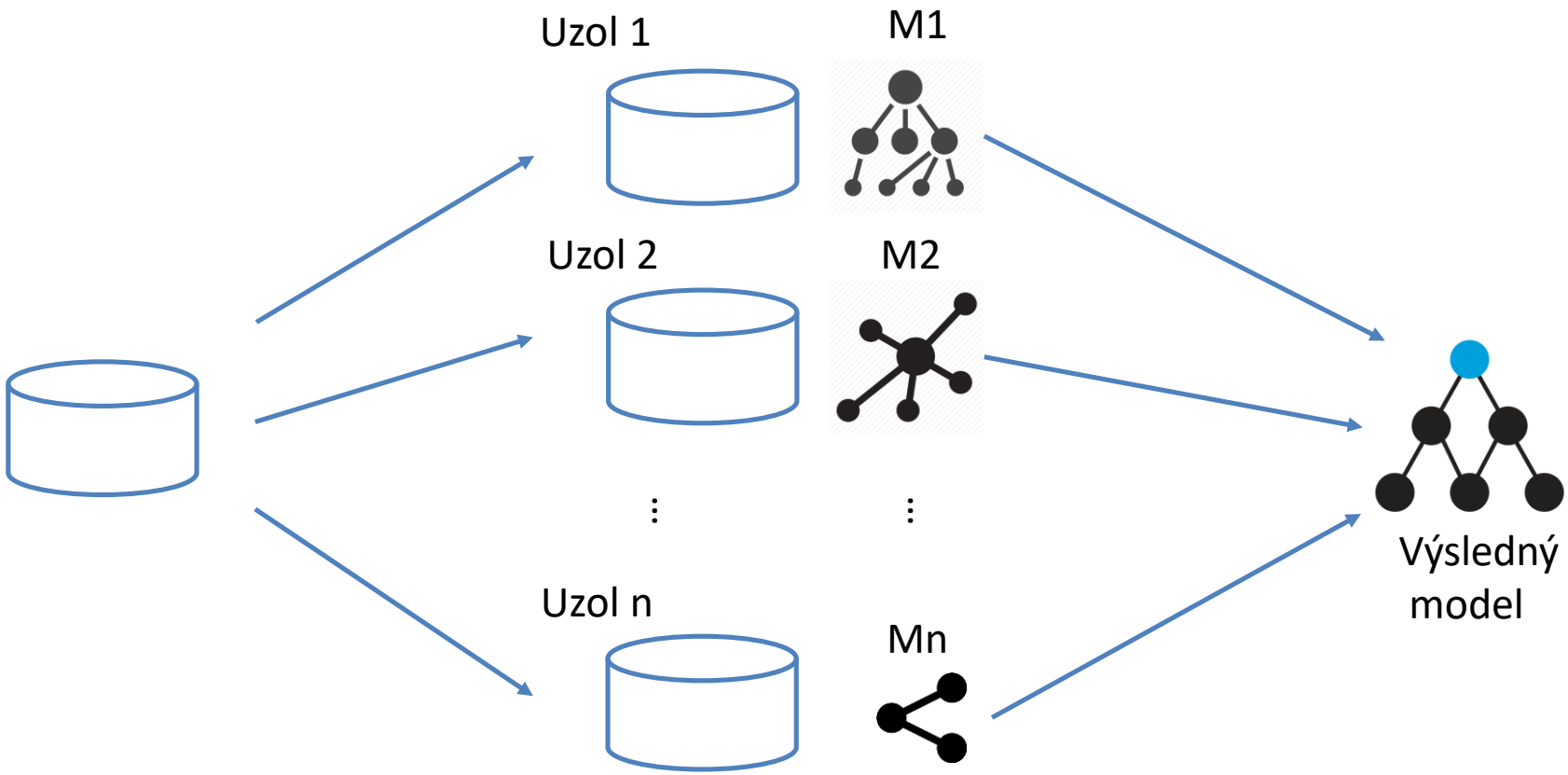


$D = D_1 + D_2 + ... + D_n$

Distribúcia založená na modeli

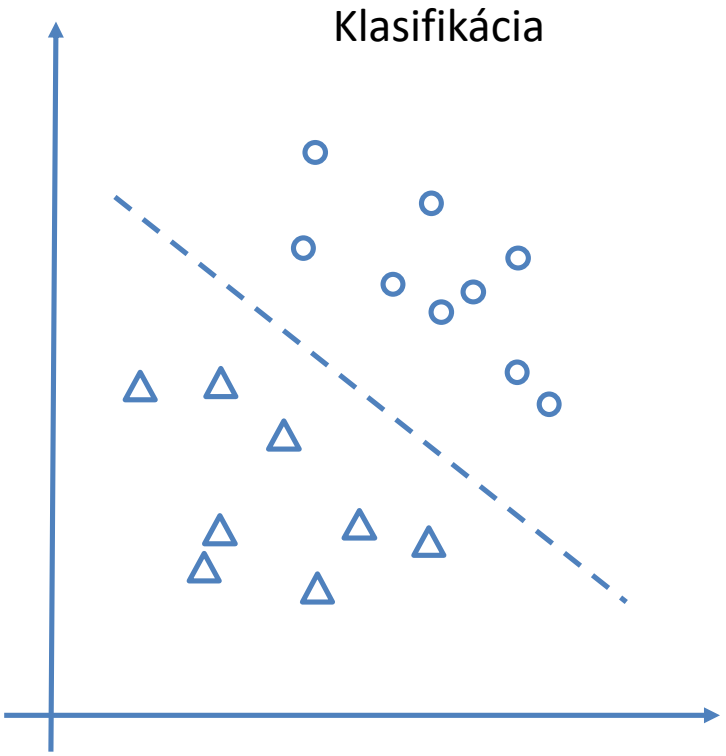
- Samotná tvorba modelu je rozdelená na pracovné uzly, kde sa počítajú paralelne
- Dôležité je navrhnúť dekompozíciu modelu tak, aby jednotlivé časti boli navzájom na sebe nezávislé
- Funkcia distribúcie – závisí na konkrétnom modeli
  - môže byť funkciou parametrov modelu
  - môže distribúciu riadiť fixne, alebo s určitou mierou náhodnosti, alebo dynamicky vzhľadom na potreby celej vykonávanej úlohy
- Finálny model – špecifikácia funkcie, ktorá vykoná agregáciu čiastkových modelov do finálneho

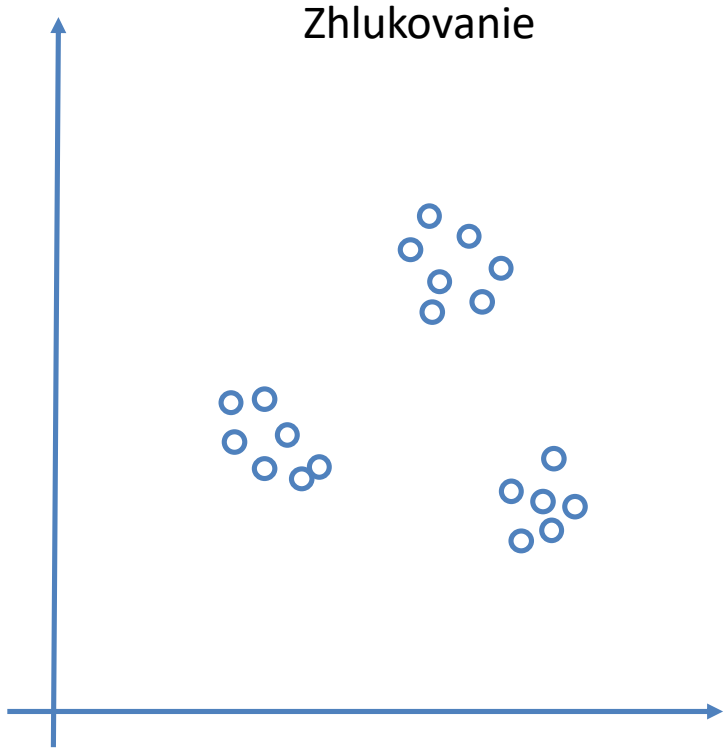
Distribúcia založená na modeli



Distribuované strojové učenie

- Základné typy metód strojového učenia





Vhodnosť metód pre paralelizáciu

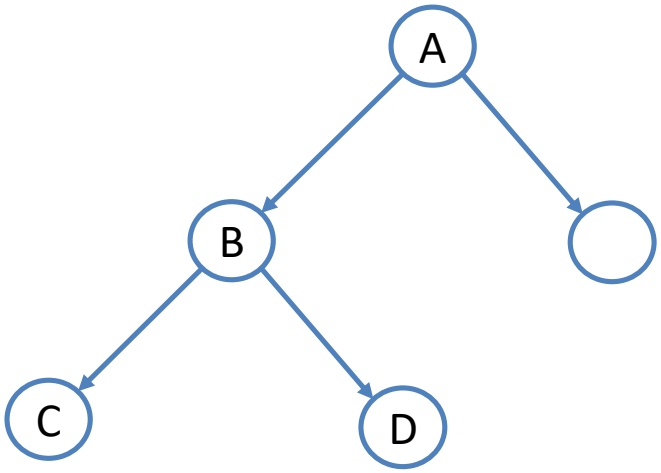
- Pre ktoré typy metód je myšlienka paralelizácie vhodnejšia
- Pre niektoré metódy niektoré technológie menej vhodné (napr. MapReduce pre iteratívne úlohy)

Distribuované stromové algoritmy

- Viacero možností vzhľadom na použitie stromového klasifikátora:
  - Učenie jedného stromu
  - Zložené prístupy
    - Random Forests
    - Bagging
    - Boosting

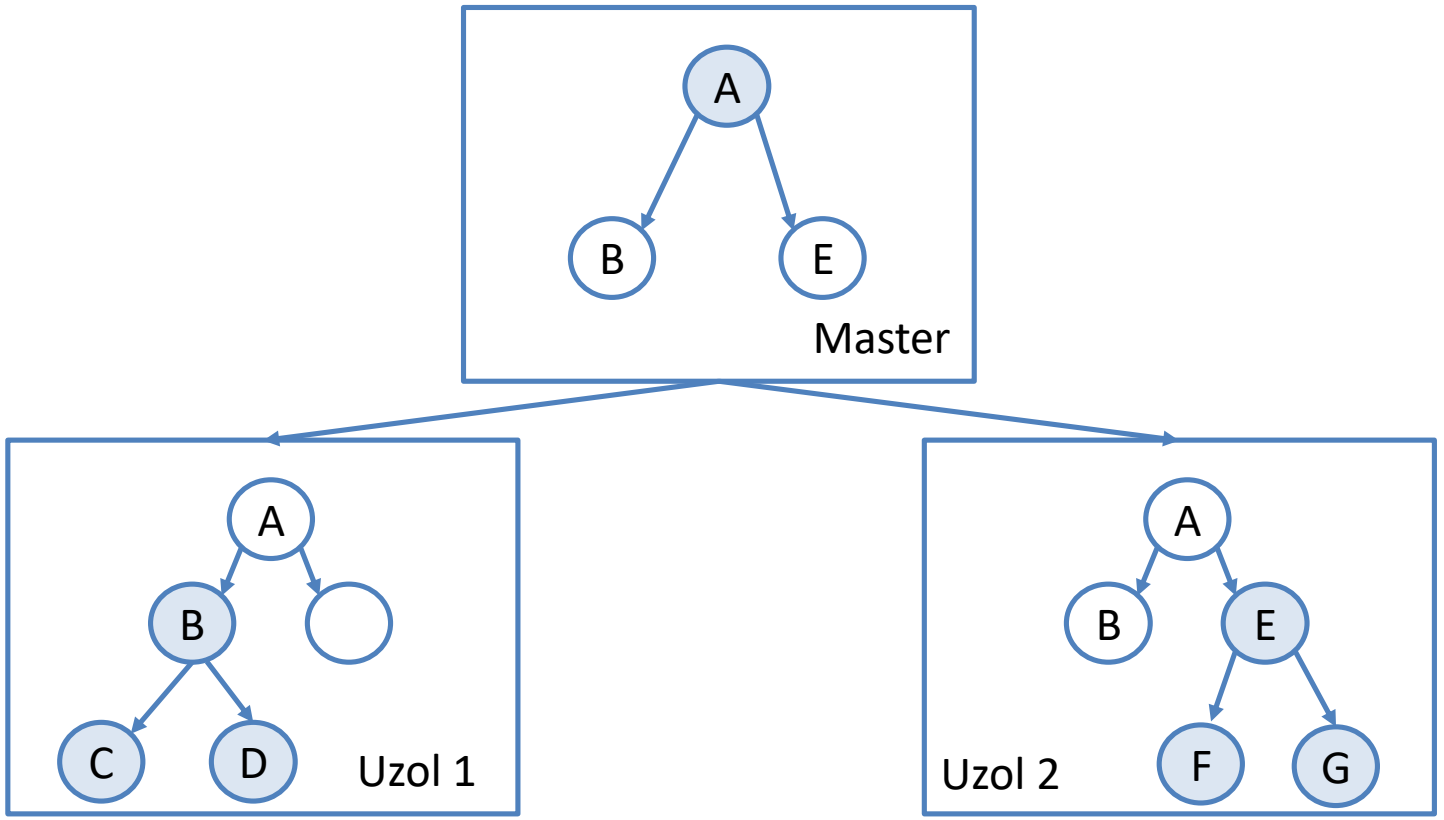
Distribuovaná tvorba stromov

- Hlavná myšlienka pri učení stromu – rozhodnúť ako vetviť uzly (algoritmy C4.5, ID3 atď.)
- Hlavná myšlienka paralelizácie



Distribuovaná tvorba stromov

- Ak vetvenie A je ukončené, môžeme vygenerovať časť stromu (vetvenie uzlov B a E) paralelne



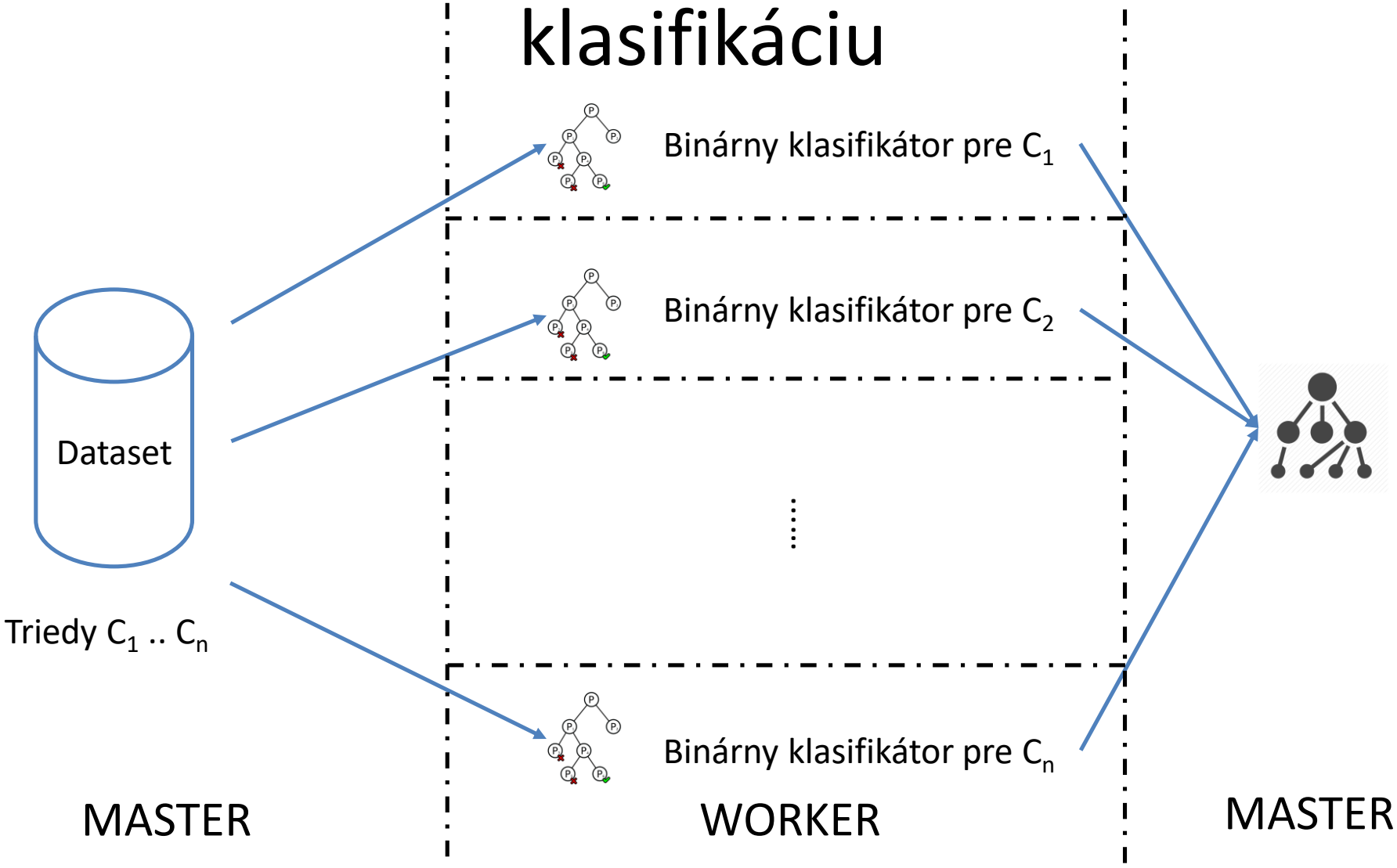
Distribuovaná tvorba stromov

- Využitie pamäte – ak dáta pre C sa zmestia do pamäti uzla, ktorý túto úlohu spracováva – je výhodnejšie generovať všetky nasledujúce podstromy na tomto uzle
- Všeobecne – keď sa „blížime“ k listovým uzlom, je výhodnejšie sa vyhnúť paralelizácii/distribúcii, resp. sa paralelizácia modelu kombinuje s paralelizáciou na úrovni dát
- Obvykle už tieto uzly nespracovávajú tak veľké množstvo dát; paralelná/distribuovaná implementácia môže byť menej efektívna ako sekvenčné spracovanie (vysoká cena komunikácie)

Distribuované stromy pre multi-label klasifikáciu

- Multi-label klasifikačný problém – inštancia (príklad) môže patriť do viac ako jednej triedy
- Viacero prístupov k riešeniu – v prípade modelov, ktoré nepodporujú tento typ klasifikácie – považovanie klasifikáciu do každej triedy ako separátny binárny klasifikačný problém
- Tzn. v prípade klasifikácie do *n* tried skonštruujeme *n* binárnych stromových klasifikátorov, ktorých výsledky potom agregujeme

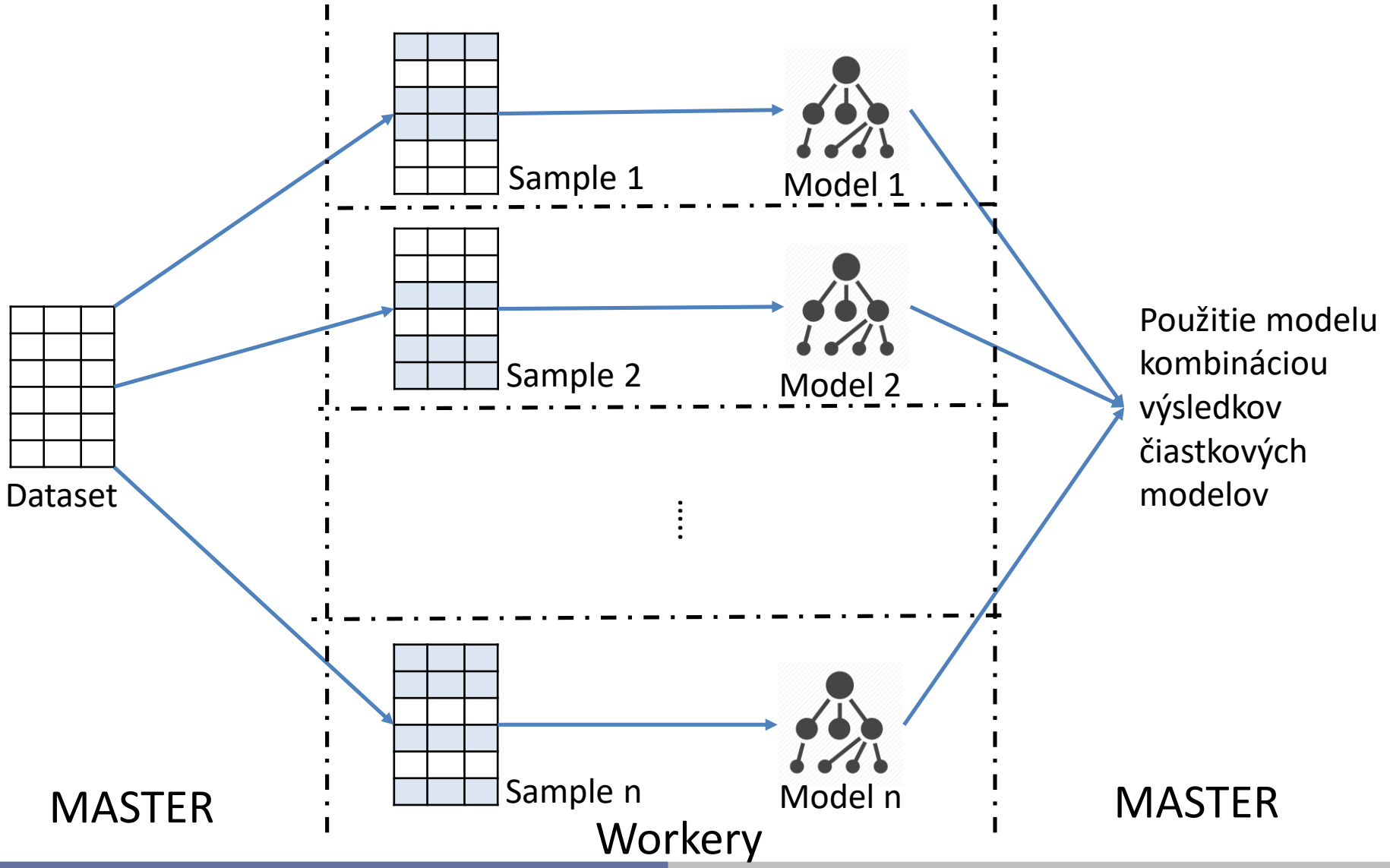
Distribuované stromy pre multi-label klasifikáciu



Bagging

- Bootstrap aggregation
- Trénovacia množina s *n* inštanciami
- Vytvoríme *k* podmnožín trénovacích dát (náhodným vzorkovaním s opakovaním) s veľkosťou $n' \leq n$
- Na týchto podmnožinách prebieha konštrukcia stromových klasifikátorov
- Klasifikácia nového príkladu –
  - vypočíta sa výsledok každého stromu, rozhodne sa väčšinovým hlasovaním jednotlivých stromov s rovnakými váhami alebo ako priemer pravdepodobnosti
  - pri regresných úlohách - aritmetickým priemerom čiastkových regresných funkcií
- Ideálne pre použitie na distribuovaných architektúrach (paralelné trénovanie čiastkových klasifikátorov)
- Hlavný cieľ – urýchlenie spracovania paralelizáciou načítavania a výpočtu

Bagging


Sample 1
Model 1
Sample 2
Model 2
Dataset
Použitie modelu kombináciou výsledkov čiastkových modelov
Sample n
Model n
MASTER
Workery
MASTER

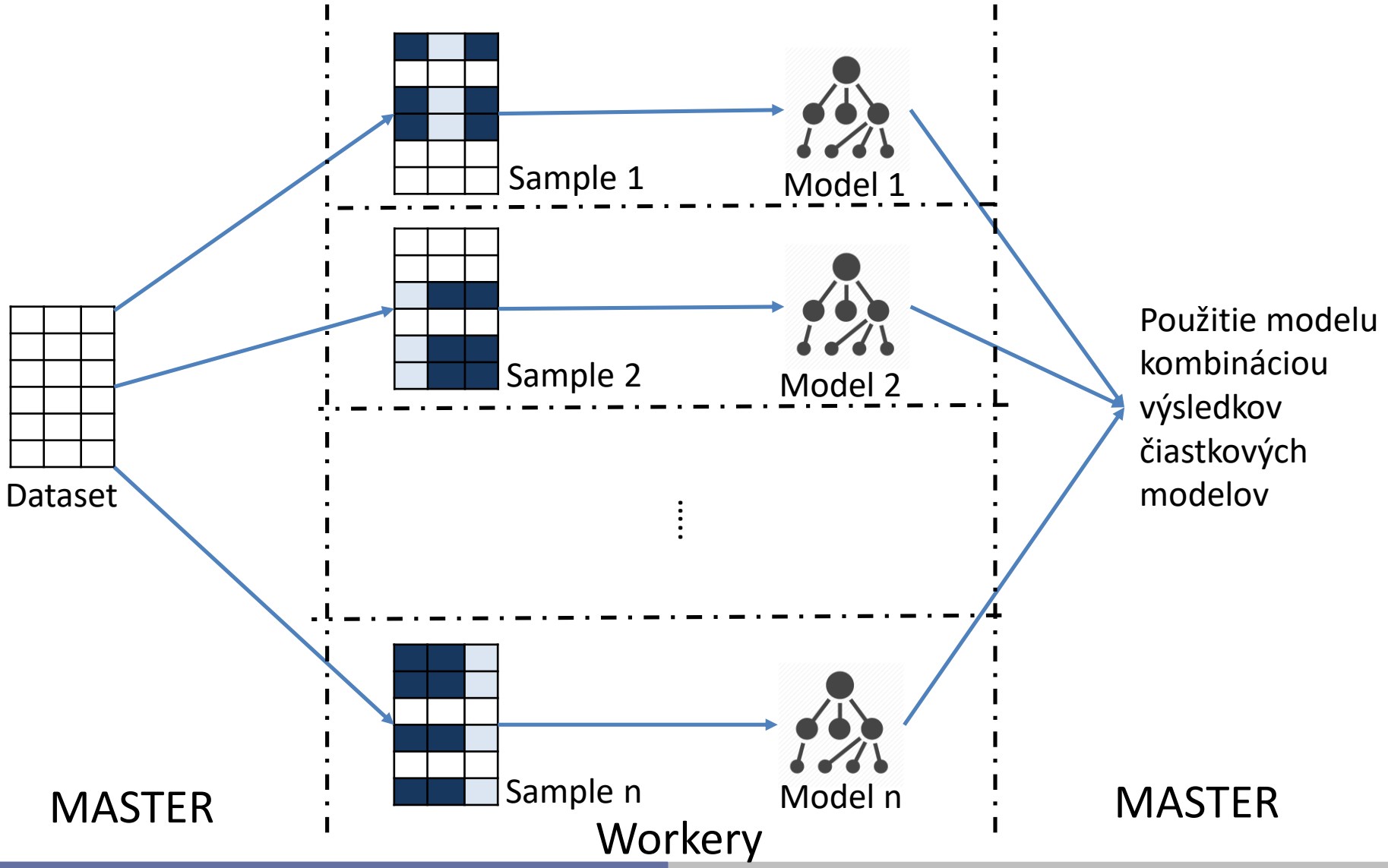
Bagging implementácia

- 1. Skopírovať dáta do HDFS
- 2. Map:
  - ak je *m* – počet prvkov v trénovacej množine a p – počet uzlov, tak pre náhodných *m/p* inštancií z trénovacej množiny určíme rovnaký kľúč (hodnota potom bude samotná inštancia)
  - prakticky v HDFS – dáta už distribuované v blokoch (replikách)
- 3. Reduce:
  - Skolektuje všetky inštancie s rovnakým kľúčom na jednom uzle, natrénovať na tejto podmnožine dát klasifikátor

Random Forests

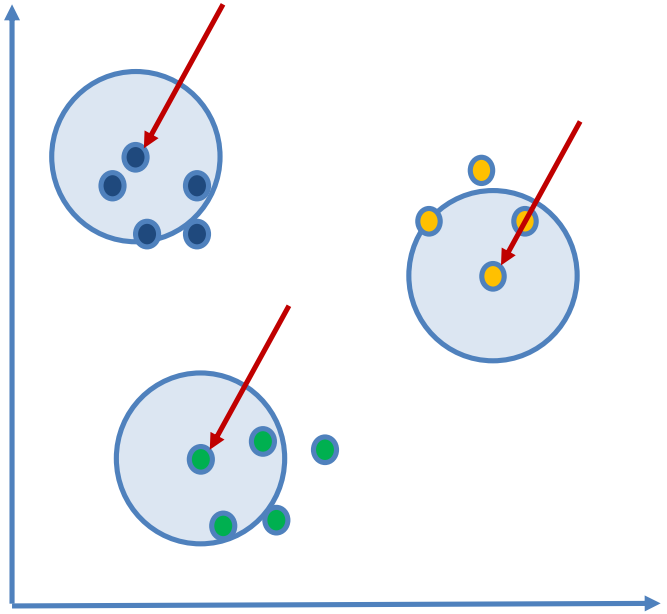
- RandomForest – základ je bagging, rozšírený o náhodný výber prediktorov
  - *m* je celkový počet príznakov v trénovacej množine
  - *n* je celkový počet inštancií v trénovacej množine
  - Náhodne sa vyberie m_0 príznakov; štandardne ako $m_0 = \sqrt{m}$
  - Príklady do trénovacej množiny sa vyberú n-násobným výberom s opakovaním z N (bootstraping, ako v baggingu)
  - Nechá sa narásť celý rozhodovací strom (neorezáva sa)
  - Pri hodnotení nového príkladu sa vypočíta výsledok každého stromu a rozhodne sa hlasovaním jednotlivých stromov (rovnako ako v prípade baggingu)
- Jeden z najpresnejších algoritmov v súčasnosti, použiteľný na veľmi veľké datasety
- Nepotrebujú testovaciu množinu ani krížovú validáciu, samé pri učení vypočítavajú odhad skutočnej chyby (OOB)
- Nie je to black-box; významnosť atribútov sa vypočítava permutáciou hodnôt atribútu a otestovaním, ako veľmi sa zmenili výsledky
- Náhrada chýbajúcich hodnôt: medián triedy, modus triedy
- Nevýhodou je na prvý pohľad zložitosť a niekedy aj náchylnosť na preučenie
- Distribuované implementácie – podobné ako pri baggingu

Random Forests



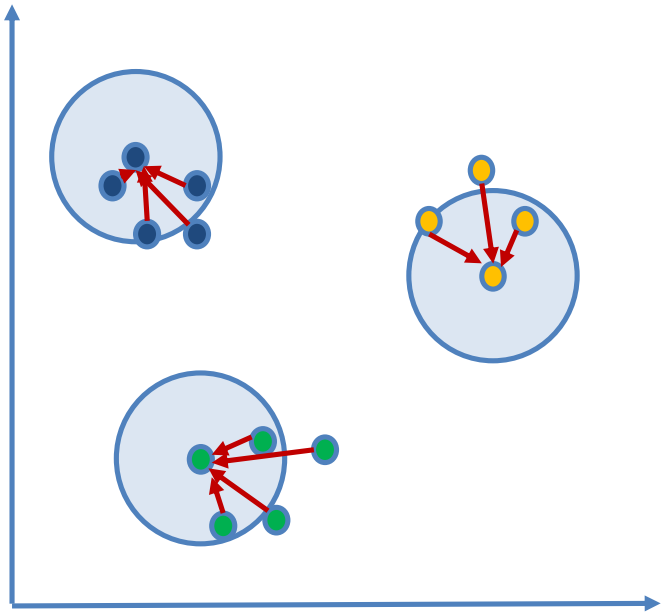
K-means zhlukovanie

- Podobnosť jednotlivých objektov a zhlukov sa meria ako ich vzdialenosť vzhľadom na priemernú hodnotu zhluku
- Štandardný algoritmus pre *k*-means:
- 1. Inicializácia *K* centier zhlukov (centroidov)
- 2. Priradenie každej inštancie (vektora) k najbližšiemu centroidu
- 3. Prepočítanie centroidov ako priemerných hodnôt inštancií zo zhluku



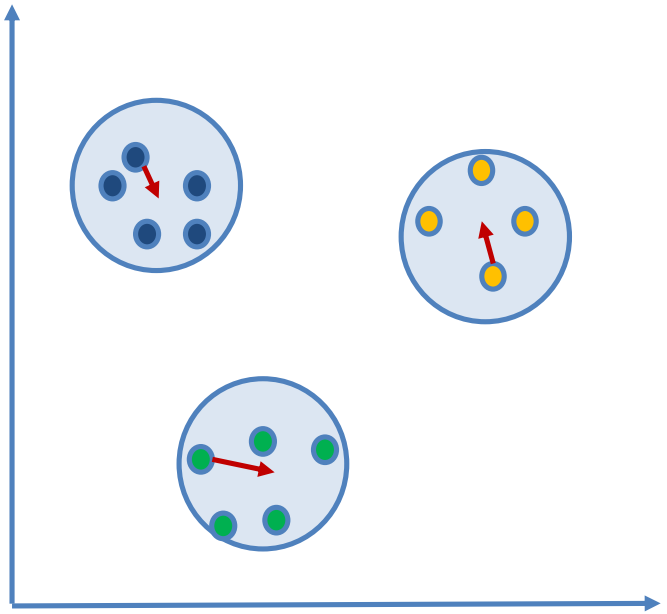
K-means zhlukovanie

- Podobnosť jednotlivých objektov a zhlukov sa meria ako ich vzdialenosť vzhľadom na priemernú hodnotu zhluku
- Štandardný algoritmus pre *k*-means:
- 1. Inicializácia *K* centier zhlukov (centroidov)
- 2. Priradenie každej inštancie (vektora) k najbližšiemu centroidu
- 3. Prepočítanie centroidov ako priemerných hodnôt inštancií zo zhluku



K-means zhlukovanie

- Podobnosť jednotlivých objektov a zhlukov sa meria ako ich vzdialenosť vzhľadom na priemernú hodnotu zhluku
- Štandardný algoritmus pre *k*-means:
- 1. Inicializácia *K* centier zhlukov (centroidov)
- 2. Priradenie každej inštancie (vektora) k najbližšiemu centroidu
- 3. Prepočítanie centroidov ako priemerných hodnôt inštancií zo zhluku



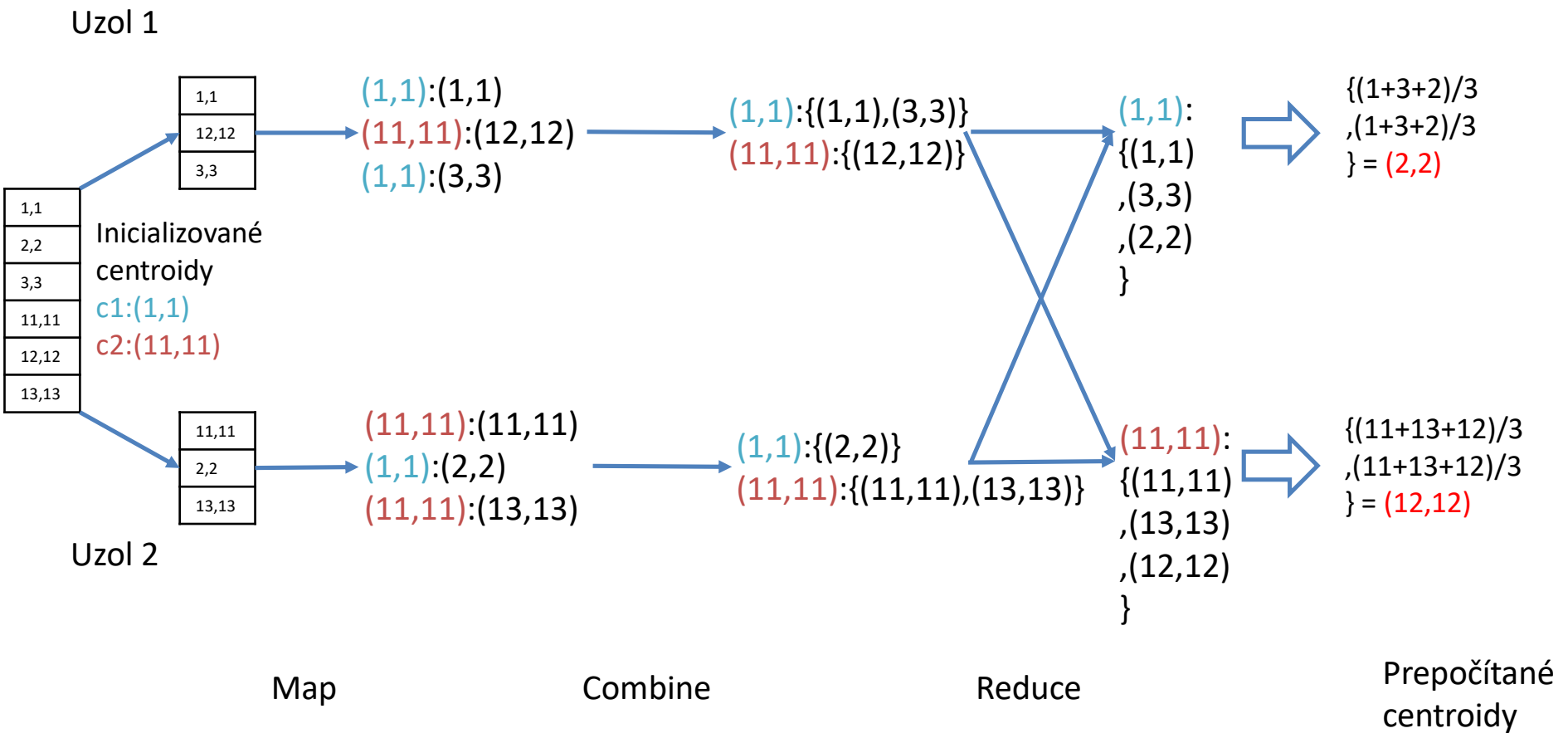
Paralelizácia *k*-means prístupu

- Výpočtovo najnáročnejší krok v k-means – počítanie vzdialeností inštancií od centroidov

MapReduce implementácia

- Každá inštancia – pár (kľúč/hodnota), kde
  - Kľúč: zhluk (resp. centroid, ktorý ho reprezentuje), do ktorého daná inštancia patrí
  - Hodnota: samotná inštancia (napr. vektor)
- Map:
  - Na každej podmnožine (split), pre všetky inštancie nájdi a priraď najbližší centroid; updatuj podľa toho hodnotu kľúča
  - Mapper emituje dvojice kľúč/hodnota (centroid/inštancia)
- Combine:
  - Combiner môže byť použitý pre spájanie inštancií s rovnakým kľúčom (už v map fáze, resp. medzi map a reduce fázami)
- Reduce:
  - Pre všetky inštancie s rovnakým kľúčom (teda ktoré patria do spoločného zhluku), prepočítaj nový centroidy

Paralelizácia *k*-means prístupu



MapReduce implementácia

- Nepriradzujeme špecificky dáta k jednotlivým uzlom
- Dáta uložené v HDFS blokoch – programátor nepozná ich presné umiestnenie
- Tzn. – nemáme kontrolu nad tým, kde jednotlivé vektory budú spracovávané
- Dôsledok – nepoznáme cenu komunikácie a načítavania dát