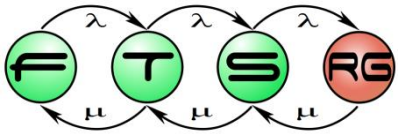
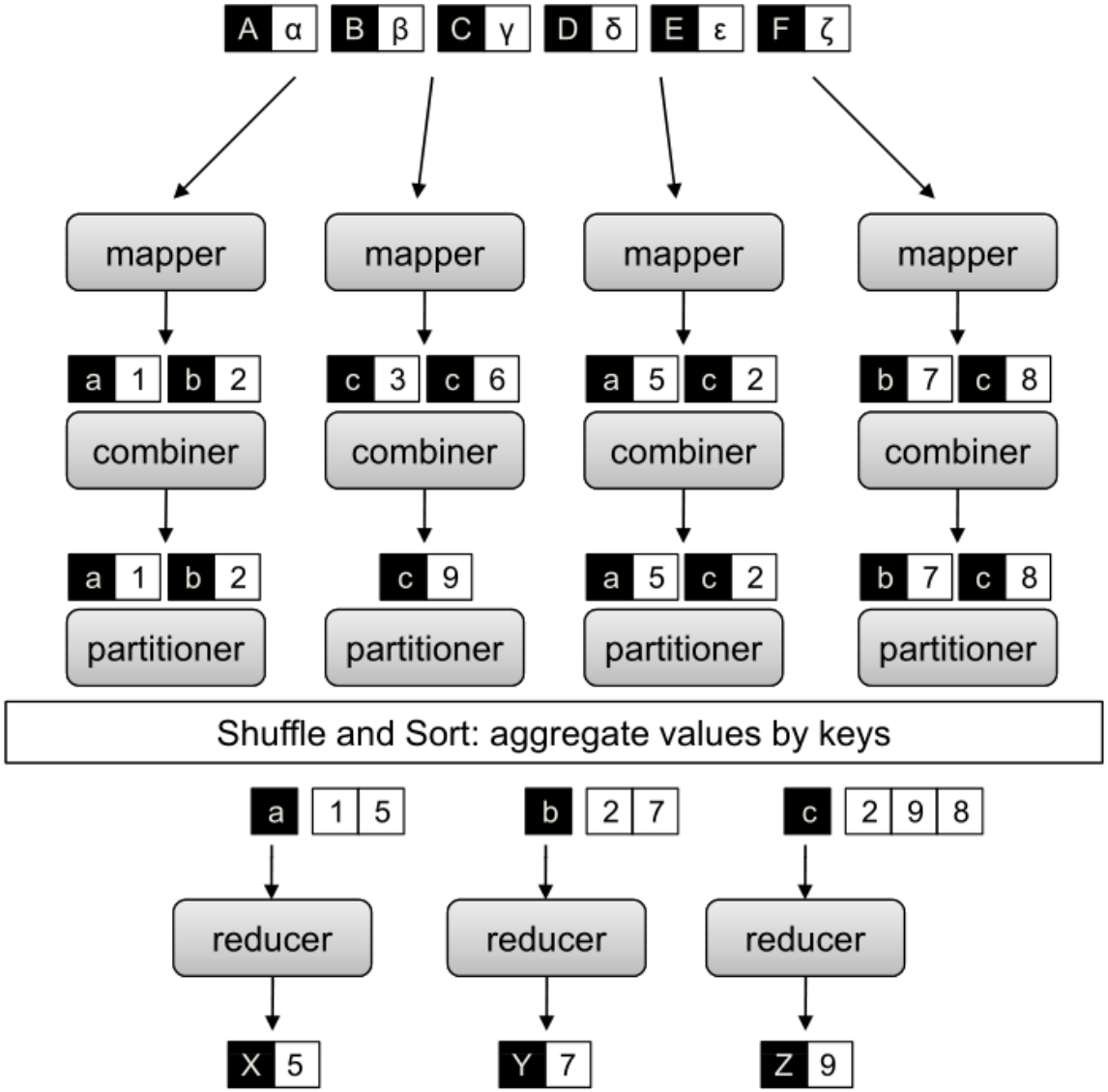
További MapReduce szemelvények: gráfproblémák

„Big Data” elemzési módszerek

Kocsis Imre
ikocsis@mit.bme.hu
2015. 10. 28.



MapReduce: a teljes kép



Forrás: [1], p 30

Partíciók és Kombinálás

- Legegszerűbb part.: hash(kulcs) mod #reducer
  - f: kulcs → reducer
- Kombinálás: lokális aggregálás
  - „mini-reducer”
  - Hálózati kommunikáció redukálása
  - Általánosságban nem „csereszabatos” a reducer-rel
  - Word countnál?

PageRank

The **anatomy** of a **large-scale hypertextual Web search engine**

S Brin, L Page - Computer networks and ISDN systems, 1998 - Elsevier

In this paper, we present Google, a prototype of a **large-scale search engine** which makes heavy use of the structure present in **hypertext**. Google is designed to crawl and index the **Web** efficiently and produce much more satisfying **search** results than existing systems. **...**

Idézetek száma: 11052 Kapcsolódó cikkek Mind a(z) 349 változat Idézés

PageRank

- Keresés a weben: találatok sorrendezése?
  - „spam farm”, „term spam”, „spider traps”, …
- „random surfer” modell
  - látogatás valószínűsége
  - *d* csillapító paraméter
- Egy oldal PageRank-je magas, ha
  - sok oldal linkeli vagy
  - magas PageRank-ű oldalak linkelik

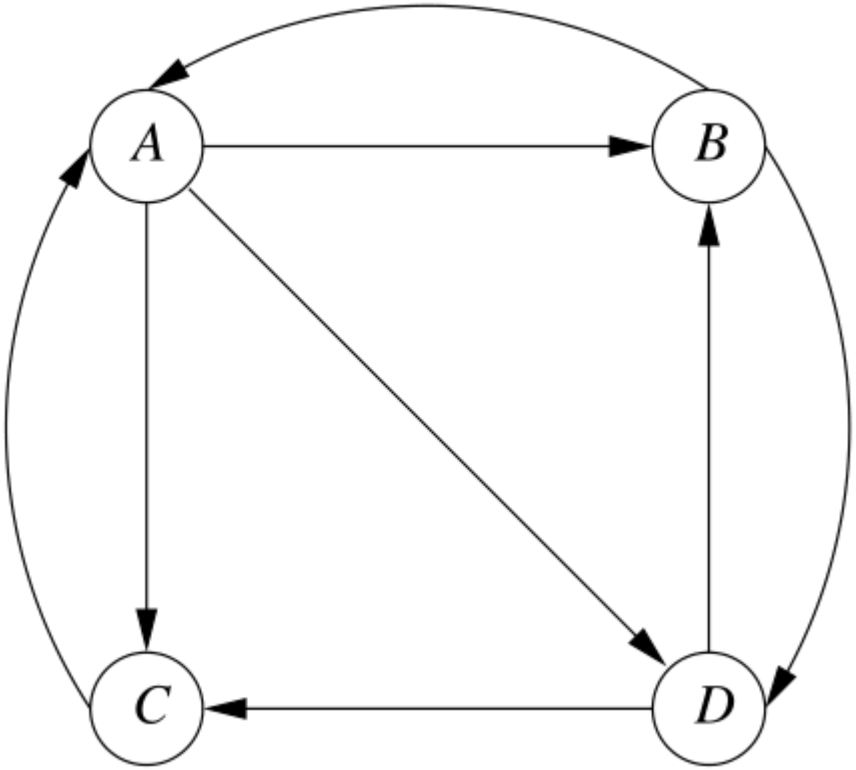
PageRank

- A oldalt linkeljék $T_1, T_2, \ldots, T_n$ oldalak.
- $0 < d < 1$.
  - Eredeti, publikált default: 0.85
- $C(A)$: kimenő fokszám
- Definíció:

$$PR(A) = (1 - d) + d\left(\frac{PR(T_1)}{C(T_1)} + \cdots + \frac{PR(T_n)}{C(T_n)}\right)$$

- ~ random látogató helyének val. eloszlása

Tranzíciós mátrix



$$M = \begin{bmatrix} 0 & 1/2 & 1 & 0 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{bmatrix}$$

PageRank számítása

- $\boldsymbol{v_0}$ kezdeti eloszlás
- Egy lépés után: $M\boldsymbol{v_0}$
- Két lépés után: $M(M\boldsymbol{v_0}) = M^2\boldsymbol{v_0}$

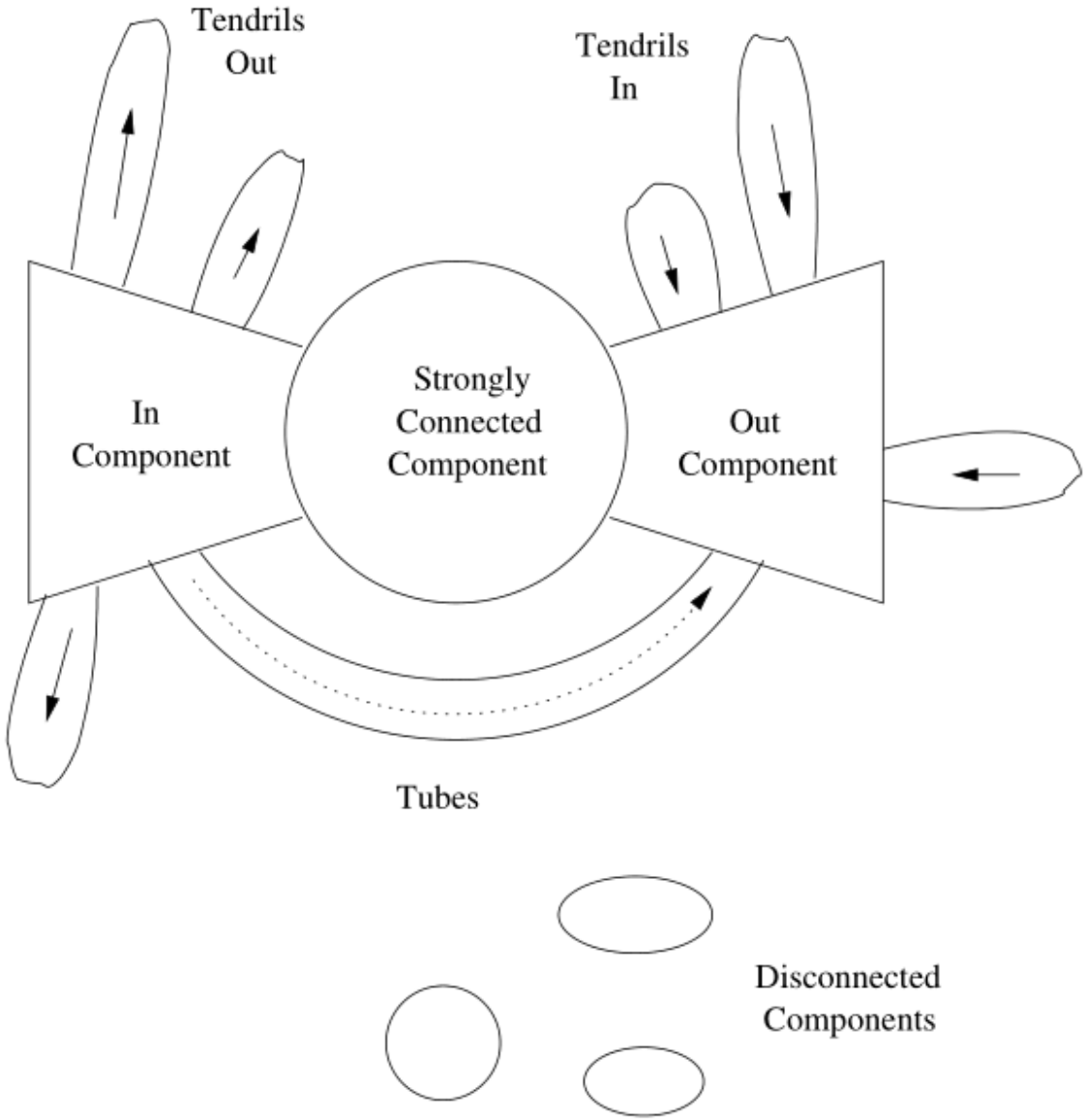
- De ez egy Markov folyamat!
  - Van $\boldsymbol{v}$ stacionárius eloszlása,
  - Ha a gráf erősen összefüggő és nyelőmentes

PageRank számítása

- Gyakorlatban: iterálás
  - de max. a normál lebegőpontos aritmetika határáig (50-75 iteráció a webre)

- Ez is négyzetes minden lépésben…
- … de M ritka volta kihasználható

„teleportálás”: „spider traps” és „dead ends”



MapReduce?

- t db mátrix-vektor szorzás

- N.B. a külső ciklus lehet probléma
  - Job indítási overhead!
  - Mátrix újra felolvasása minden iterációban
  - …

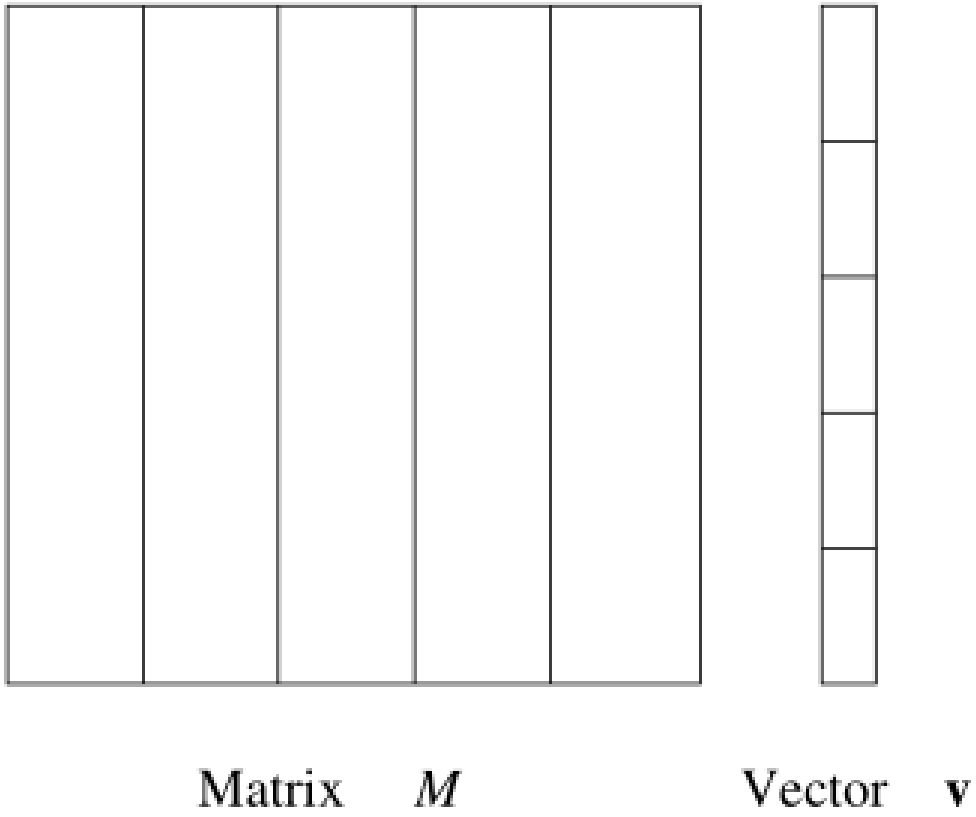
- M legyen $n \times n$-es mátrix

$$x_i = \sum_{j=1}^{n} m_{ij} v_j$$

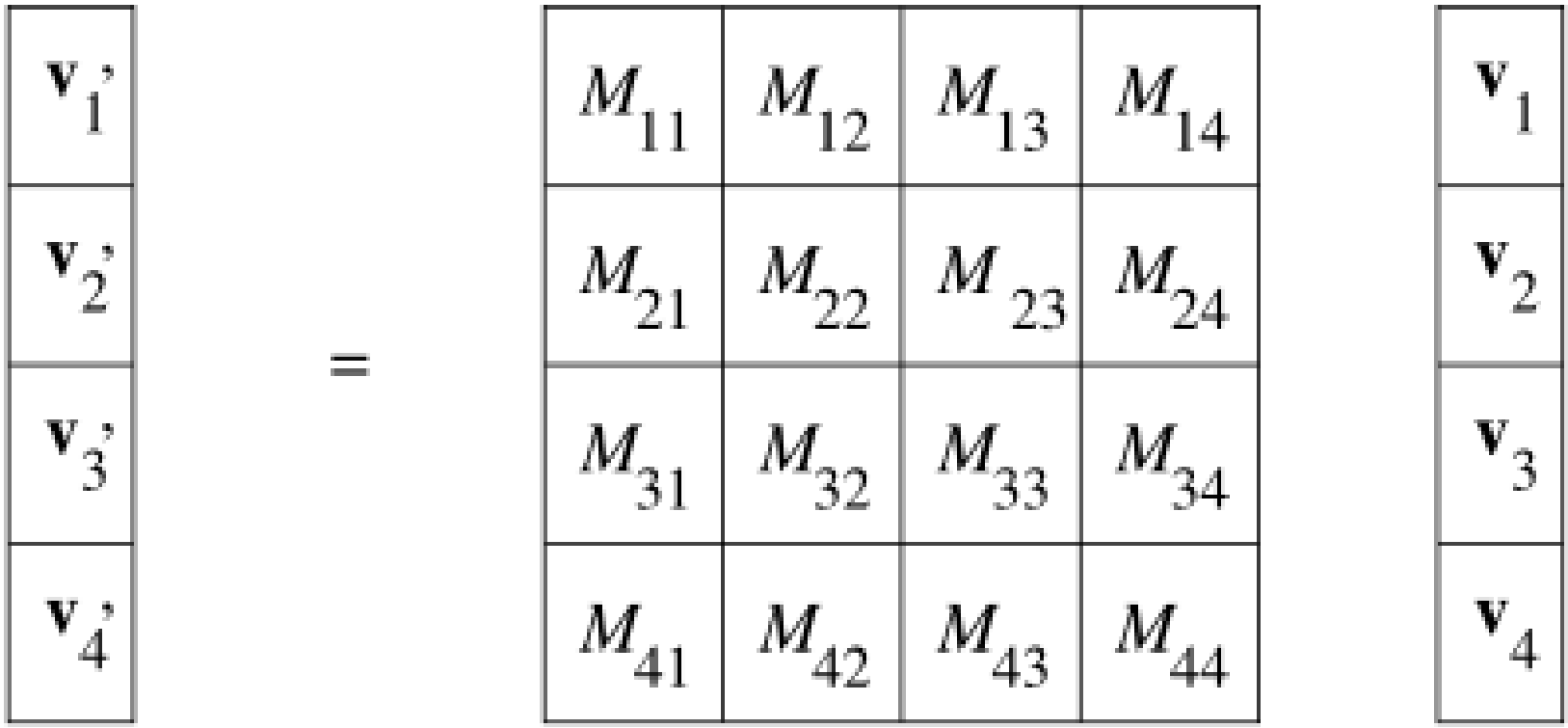
- $n = 100$ nem Big Data…
- Mapper, ha a vektor befér a memóriába: $(i, m_{ij} v_j)$

MapReduce mátrix-vektor szorzás

- Ha a vektor nem fér el a memóriában: „csíkozás” (*striping*)



PageRank k^2 mapperrel



Általánosítás: a GIM-V primitív

- „Generalized Iterative Matrix-Vector multiplication”
- A szokásos feladat:

$$M \times v = v', \text{ ahol } v'_i = \sum_{j=1}^{n} m_{ij} v_j$$

- Valójában három operátor:
  - `combine2`: a szorzás
  - `combineAll`: a szumma
  - `assign`: vektorelemek frissítése

Általánosítás: a GIM-V primitív

- „Generalized Iterative Matrix-Vector multiplication”
- A szokásos feladat:

$$M \times v = v', \text{ ahol } v'_i = \sum_{j=1}^{n} m_{ij} v_j$$

- Valójában három operátor:
  - `combine2`: a szorzás
  - `combineAll`: n szorzás-eredmény szummája i-re
  - `assign`: vektorelemek frissítése

GIM-V

$$v' = M \times_G v$$

$$v_i' = assign(v_i, combineAll_i(\{x_j | j = 1 \dots n, x_j = combine2(m_{ij}, v_j)\}))$$

- Iteratív: $\times_G$-t konvergenciakritériumig alkalmazzuk
- BTW feltételezve egy `E(sid, did, val)` és `V(id,val)` táblát $\times_G$ SQL-ben:

```
SELECT E.sid, combineAll_{E.sid}(combine2(E.val,V.val))
  FROM E, V
  WHERE E.did=V.id
  GROUP BY E.sid
```

GIM-V: PageRank

- M: oszlopnormalizált szomszédossági mátrix
- `combine2`: $c \times m_{ij} \times v_j$
- `combineAll`: $\frac{1-c}{n} + \sum_{j=1}^{n} x_j$
- `assign`: v_{new}

GIM-V: Random Walk with Restart

- Csomópontok közelségének mérése
- A k-adik csomópontból induló vektor egyenlete (M ua):

$$r_k = cMr_k + (1 - c)e_k$$

- Ahol e_k nullvektor a k-adik, 1 értékű pozíciótól eltekintve
- `combine2`: $c \times m_{ij} \times v_j$
- `combineAll`: $(1 - c)I(i \neq k) + \sum_{j=1}^{n} x_j$
- `assign`: v_{new}

GIM-V: összefüggő komponensek

- Minden v_i-hez „komponens ID” c_i^h:
- Minimum csomópont-ID h hopon belül
- M: szomszédossági mátrix

- Minden iterációban:
  - Minden csomópont „elküldi” szomszédjainak c_i^h-ját
  - Iteráció: minden csomópontban minimumképzés

- `combine2`: $m_{ij} \times v_j$

- `combineAll`:$MIN\{x_j | j = 1 \dots n\}$

- `assign`: $MIN(v_i, v_{new})$

GIM-V: összefüggő komponensek

- Megállás: nem változik a vektor

- Tétel: legfeljebb gráfátmérő számú lépés kell
  - *Diameter*: pontpárok közötti legrövidebb utak hosszának maximuma

GIM-V: BASE algoritmus

Algorithm 1: GIM-V BASE Stage 1.

Input : Matrix $M = \{(id_{src}, (id_{dst}, mval))\}$,
Vector $V = \{(id, vval)\}$
Output: Partial vector
$V' = \{(id_{src}, \texttt{combine2}(mval, vval)\}$

```
1  Stage1-Map(Key k, Value v) ;
2  begin
3      if (k, v) is of type V then
4          Output(k, v);                          // (k: id, v: vval)
5      else if (k, v) is of type M then
6          (id_dst, mval) ← v;
7          Output(id_dst, (k, mval));             // (k: id_src)
8  end
9  Stage1-Reduce(Key k, Value v[1..m]) ;
10 begin
11     saved_kv ←[ ];
12     saved_v ←[ ];
13     foreach v ∈ v[1..m] do
14         if (k, v) is of type V then
15             saved_v ← v;
16             Output(k, ("self", saved_v));
17         else if (k, v) is of type M then
18             Add v to saved_kv      // (v: (id_src, mval))
19     end
20     foreach (id'_src, mval') ∈ saved_kv do
21         Output(id'_src, ("others",combine2(mval', saved_v)
22     end
23 end
```

GIM-V: BASE algoritmus

Algorithm 2: GIM-V BASE Stage 2.

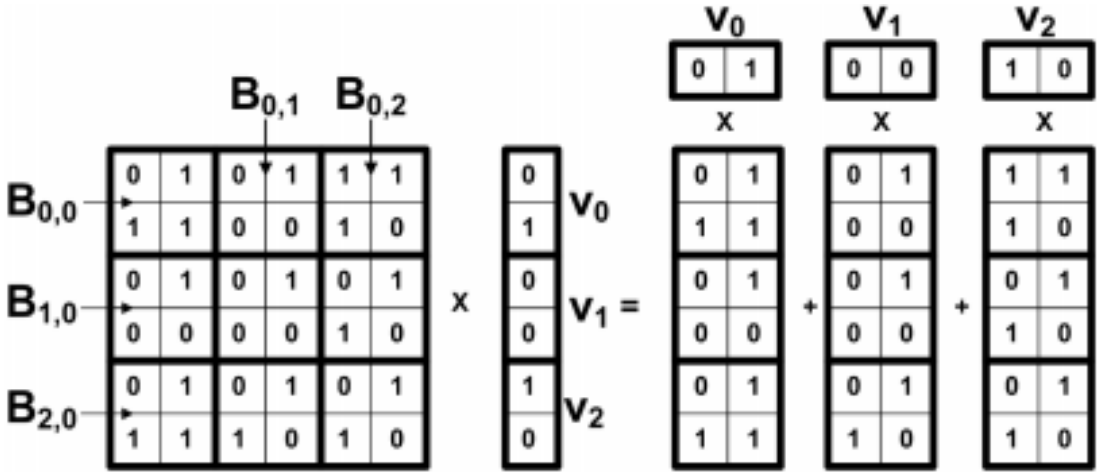
Input : Partial vector $V' = \{(id_{src}, vval')\}$
Output: Result Vector $V = \{(id_{src}, vval)\}$

```
1  Stage2-Map(Key k, Value v) ;
2  begin
3      Output(k, v);
4  end
5  Stage2-Reduce(Key k, Value v[1..m]) ;
6  begin
7      others_v ←[ ];
8      self_v ←[ ];
9      foreach v ∈ v[1..m] do
10         (tag, v') ← v;
11         if tag == "same" then
12             self_v ← v';
13         else if tag == "others" then
14             Add v' to others_v;
15     end
16     Output(k,assign(self_v,combineAll_k(others_v)));
17 end
```

GIM-V: algoritmusok

- Naive multiplication, **block multiplication**, clustered edges, diagonal block iteration, …

- Blokk szorzás: vektorokat és mátrixot is blokkosítva tárolunk (és csak ha nem nulla)



Hivatkozások

- [1] Lin, J., & Dyer, C. (2010). Data-Intensive Text Processing with MapReduce. *Synthesis Lectures on Human Language Technologies*, *3*(1), 1–177. doi:10.2200/S00274ED1V01Y201006HLT007
- [2] http://www.slideshare.net/jeffreybreen/big-data-stepbystep-using-r-hadoop-with-rhadoops-rmr-package
- [3] http://www.cloudera.com/content/support/en/downloads.html
- [4] http://hortonworks.com/products/hortonworks-sandbox/
- [5] http://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html
- [6] http://home.mit.bme.hu/~ikocsis/notes/2013/10/23/(r)hadoop-sandbox-howto/
- [7] http://home.mit.bme.hu/~ikocsis/notes/2013/10/28/rhadoop-sandbox-with-the-cloudera-quickstart-vm/
- [8] Iványi, A. "Informatikai algoritmusok." *ELTE Eötvös Kiadó, Budapest* (2004).
- [9] Chu, C. T., Kim, S. K., Lin, Y. A., Yu, Y., Bradski, G. R., Ng, A. Y., & Olukotun, K. (2006). Map-reduce for machine learning on multicore. In B. Schölkopf, J. Platt, & T. Hofmann (Eds.), *Advances in Neural Information Processing Systems 19: Proceedings of the 2006 Conference* (pp. 281–288). MIT Press.