An introduction to MapReduce and MapReduce algorithms for the Web

Mauro Sozio

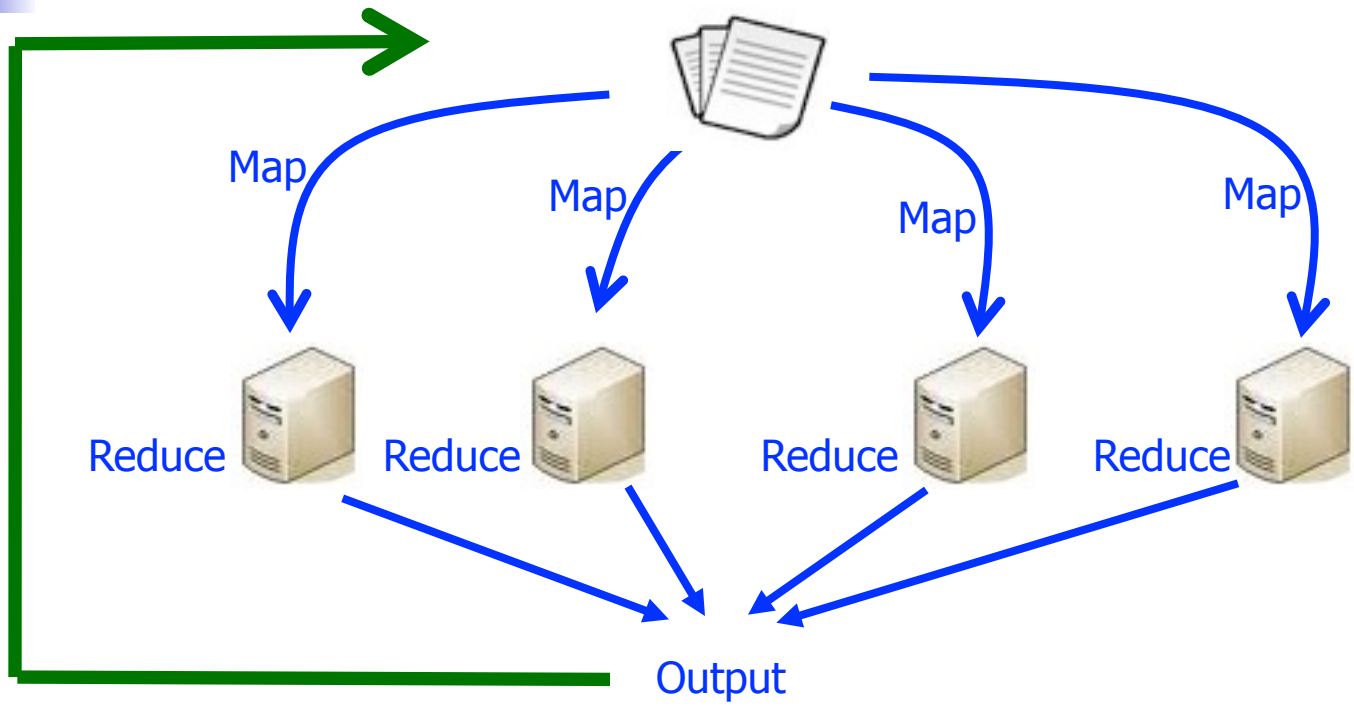
Massive amount of data generated daily

Some facts:

- Facebook >800M users, 900M objects (pages, groups, events).
- Flickr >50 million users, 6 billion images!
- Twitter > 300M users, 1.6 billion search queries per day
- Google > 3 billion search queries per day

How to make sense of all these data?

MapReduce



Initially developed by Google, it is nowadays used by several companies (Yahoo!, IBM) and universities (Cornell, CMU...).

Many sequential algorithms have been adapted to MapReduce.

MapReduce Algorithms

- Matrix-vector iterative algorithms efficient in MapReduce:
  - PageRank;
  - Linear and logistic regression, naive Bayes, k-means clust., SVM;
  - Pair-wise document similarity, language modeling.

MapReduce in Brief

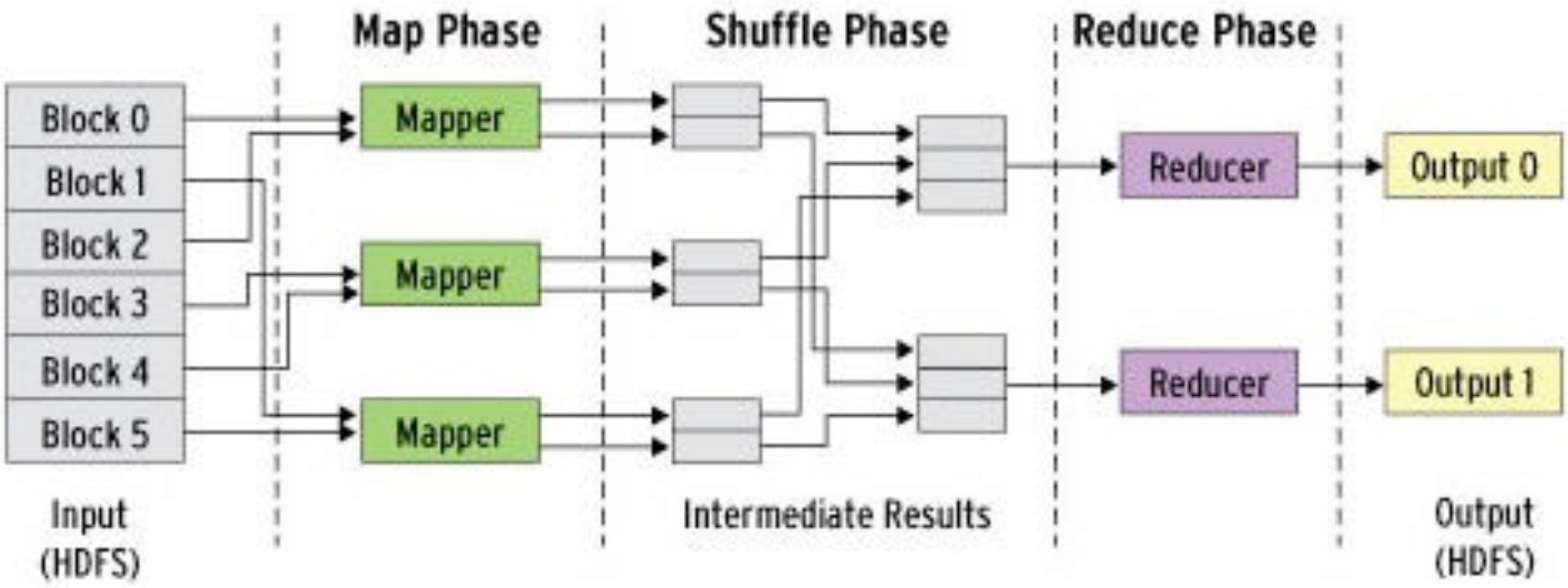
- The programmer defines the program logic as two functions:
  - Map transforms the input into key-value pairs to process
  - Reduce aggregates the list of values for each key
- The MapReduce environment takes in charge distribution aspects.
- A complex program can be decomposed as a succession of Map and Reduce tasks.
- Higher-level languages (Pig, Hive, etc.) help with writing distributed applications.

Three operations on key-value pairs

- User-defined: *map* : (key,value) → List(key',value')
  e.g. *map (docId, document) → ((jaguar,1),(mac,1),(jaguar,1),...)*
- Built-in function: *shuffle*. Group pairs with a same key and assign them to a reducer. Seamless to the user.
- User-defined: *reduce*: (key,List(values)) → List(key',value')
  e.g. reduce: (jaguar,(1,2,1)) → ((jaguar,4))

- Mapping operations are independent from each other!
- The output from a reduce could be the input for another MapReduce iteration.

Map,Reduce,Shuffle



MR: Counting Words

```
void map(String docId, String document):
  // docId: document name
  // document: document content
  for each word w in document:
    Output(w, "1");

void reduce(String word, Iterator partialSums):
  // word: a word
  // partialCounts: a list of partial sums
  int sum = 0;
  for each pc in partialCounts:
    sum += ParseInt(pc);
  Output(word, AsString(sum));
```

Example

DocID	Content
d1	the jaguar is a new world mammal of the felidae family.
d2	for jaguar, atari was keen to use a 68k family device.
d3	mac os x jaguar is available at a price of us $199 for apple's
d4	one such ruling family to incorporate the jaguar into their
d5	It is a big cat.

Example

(d1,content),,...,(d5,content) —*Map*→

jaguar 1
mammal 1
family 1
jaguar 1
jaguar 2
...

—*Shuffle*→

jaguar 1,1,2
mammal 1
family 1
...

—*Reduce*→

jaguar 5
mammal 1
family 1
...

A MapReduce cluster

- Nodes inside a MapReduce cluster are decomposed as follows:
  - A jobtracker acts as a master node; MapReduce jobs are submitted to it.
  - Several tasktrackers run the computation itself, i.e., map and reduce tasks
  - A given tasktracker may run several tasks in parallel
  - Tasktrackers usually also act as data nodes of a distributed filesystem (e.g., GFS, HDFS)

+ a client node where the application is launched.

PageRank in MR

PageRank: importance score for nodes in a graph, used for ranking query results of Web search engines. Fixpoint computation, as follows:

Compute A_G. Make sure lines sum to 1.

Let u be the uniform vector of sum 1, $v = u$.

Repeat N times:
Set $v=(1-d)A_G{}^T v + du$ (say,d=1/4).

Note: Only matrix-vector multiplications!

Recommendations in Social Networks

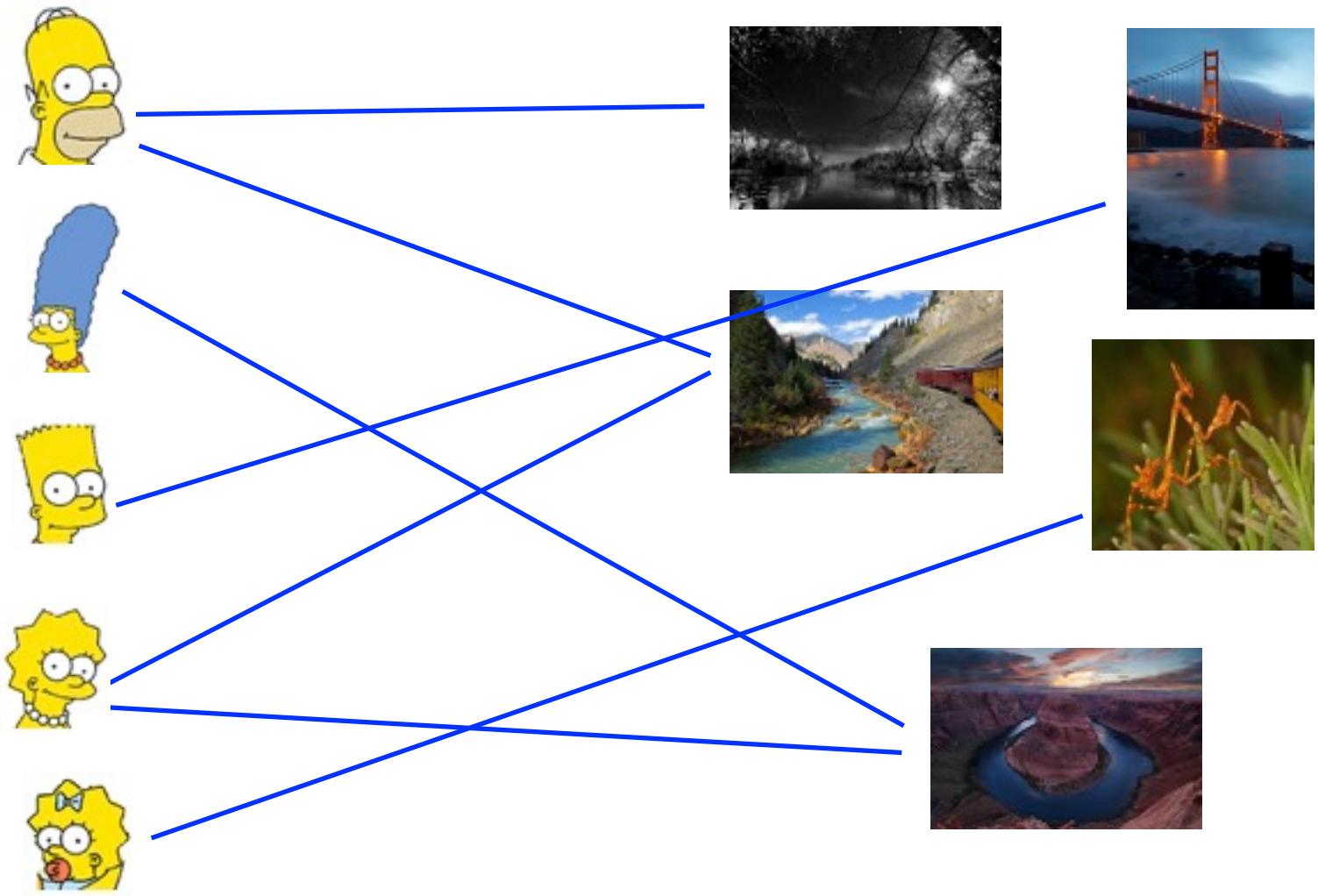




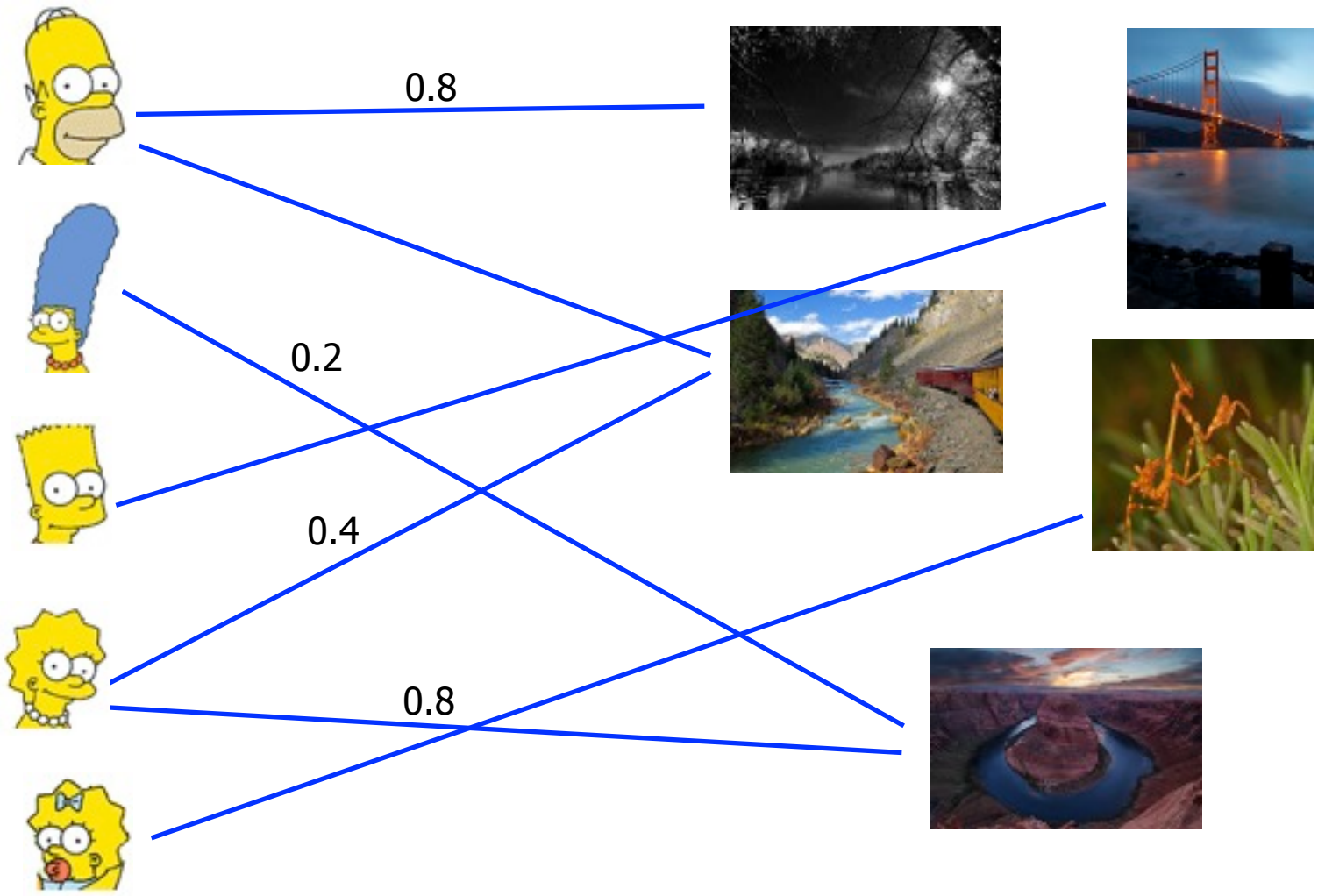




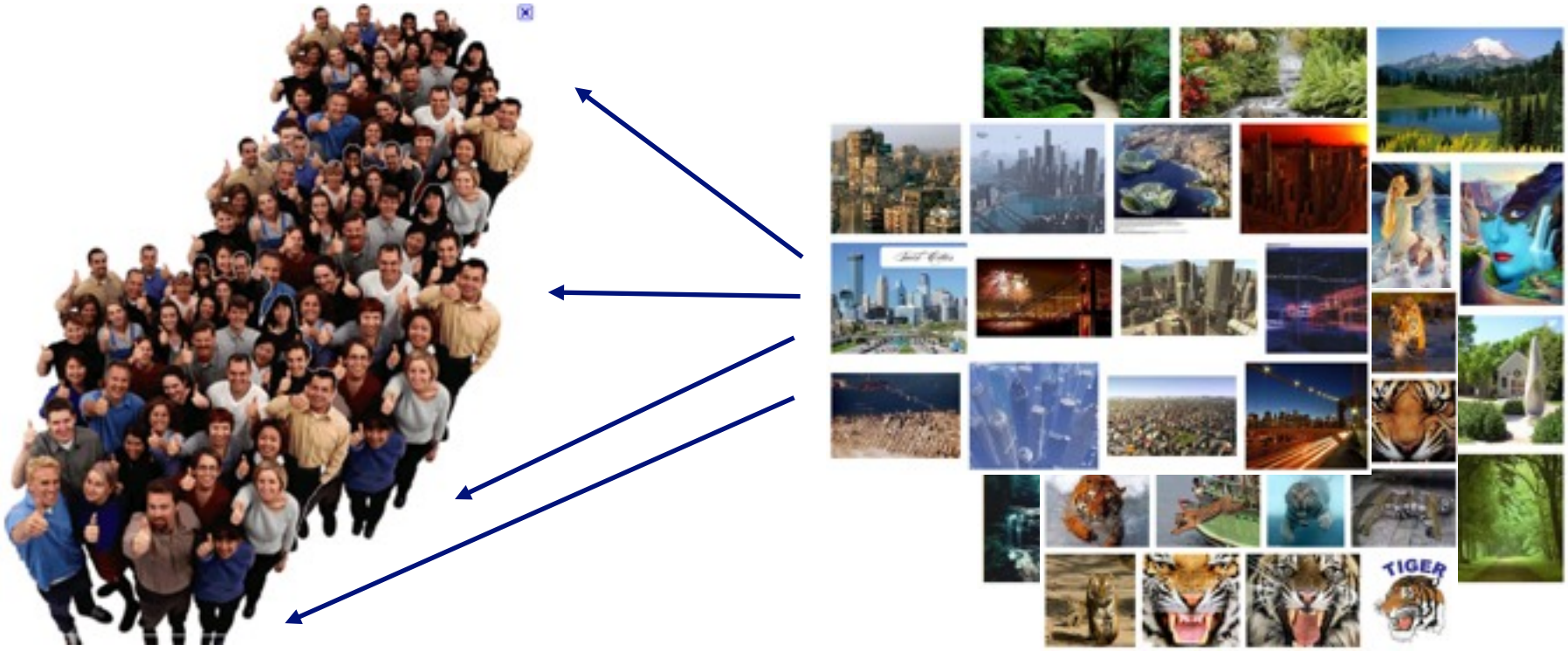
Recommendations in Social Networks



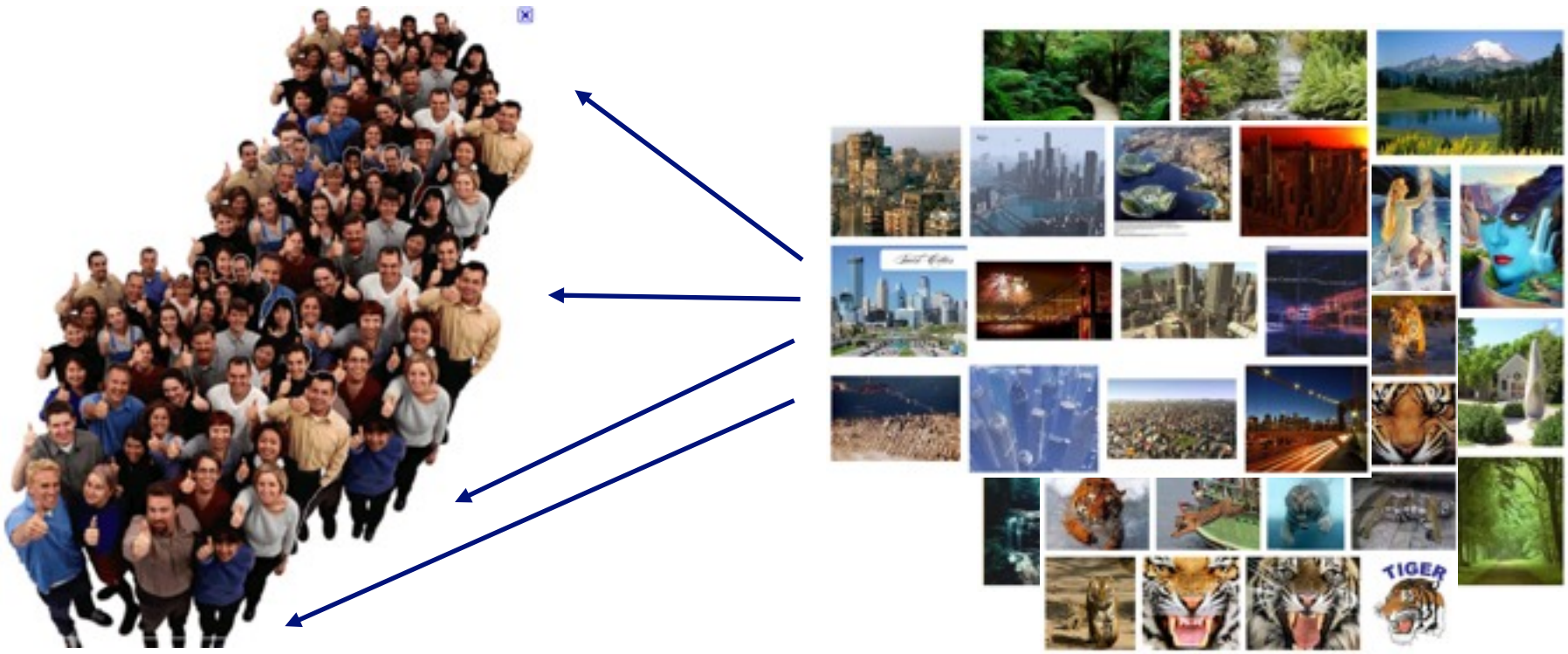
Recommendations in Social Networks



Matching billion of pics to billion of users



Matching billion of pics to billion of users



How to deal with massive data?
Parallel computation (MapReduce)

Problem Formulation

We wish to:

- maximize users' satisfaction $\longrightarrow$ max. the sum of the edge weights
- assign only "a few" items to each user $\longrightarrow$ upper bound on the number of images a user could be matched to.
- avoid that only a few items (e.g. the most popular ones) participate to the matching $\longrightarrow$ upper bound on the number of users an item is assigned to.

Problem Formulation

We wish to:

- maximize users' satisfaction ⟶ max. the sum of the edge weights
- assign only "a few" items to each user ⟶ upper bound on the number of images a user could be matched to.
- avoid that only a few items (e.g. the most popular ones) participate to the matching ⟶ upper bound on the number of users an item is assigned to.

Problem Formulation (b-matching) Given a bipartite graph G=(U,C,E) where U is a set of users and C is a set of content items, and a function $b: U \cup C \to N$ (upper bound) find an assignment of items to users s.t.:

- at most b(u) items are assigned to each user;
- each item c is assigned to at most b(c) users;
- the total weight is maximized.

Our algorithms

	Approx. Guarantee	Max. Number of MR jobs
StackMR	$\frac{1}{6+\epsilon}$	$O(\frac{\log^3 n}{\epsilon^2} \cdot \log \frac{w_{\max}}{w_{\min}})$
GreedyMR	$\frac{1}{2}$	m
StackGreedyMR	-	m

Main contributions:

- One of a few practical algorithms with provable guarantees on the number of MR jobs (sublinear).
- Extensive experimental evaluation on real data.

Experiments

3 Datasets $|T|$: number of items; $|C|$: number of users; $|E|$: total number of item-user pairs with non zero similarity.

Dataset	$\|T\|$	$\|C\|$	$\|E\|$
flickr-small	2 817	526	550 667
flickr-large	373 373	32 707	1 995 123 827
yahoo-answers	4 852 689	1 149 714	18 847 281 236

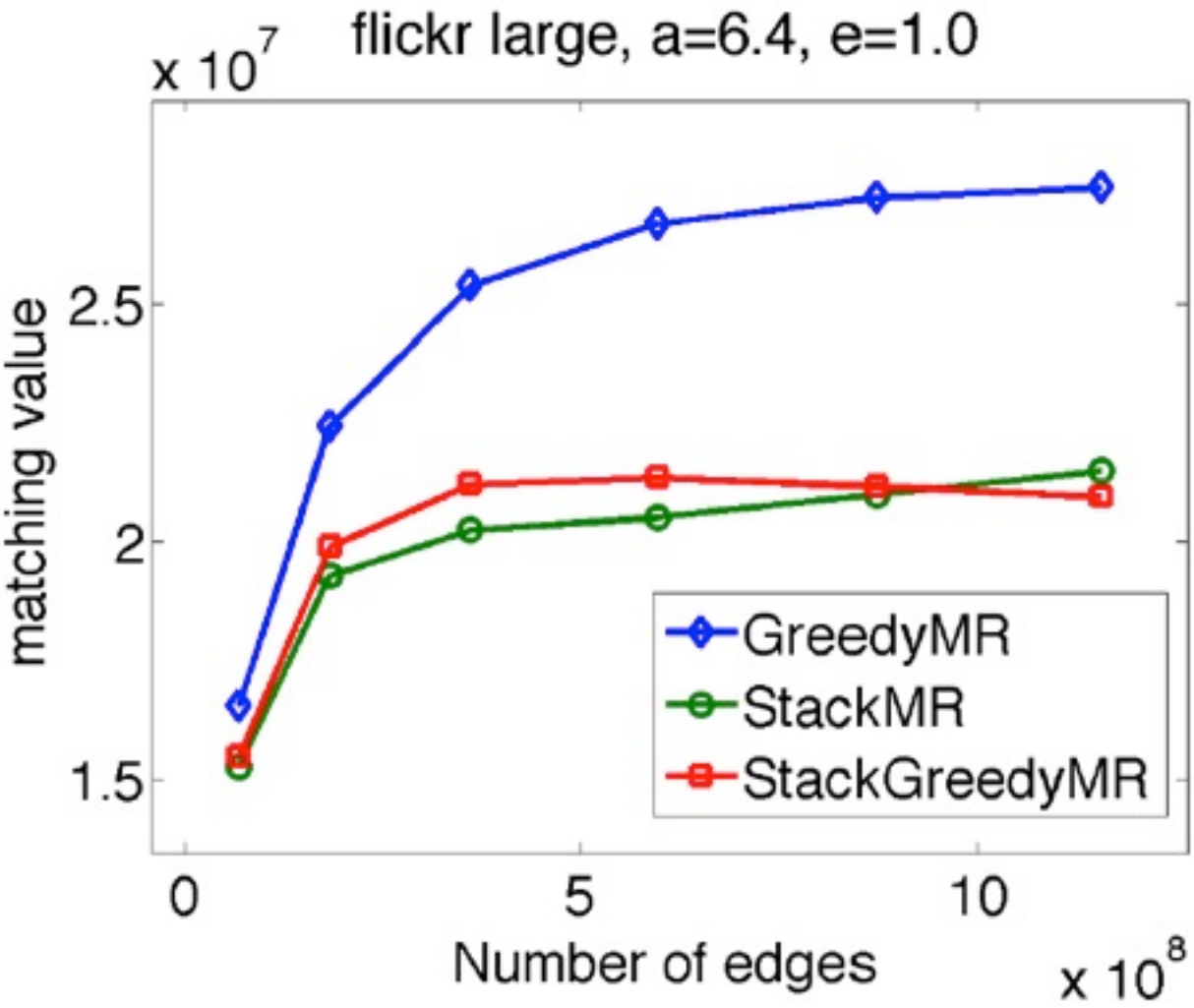
Edge weights:

- cosine similarity between tag vectors in Flickr
- tfidf in Yahoo! answers

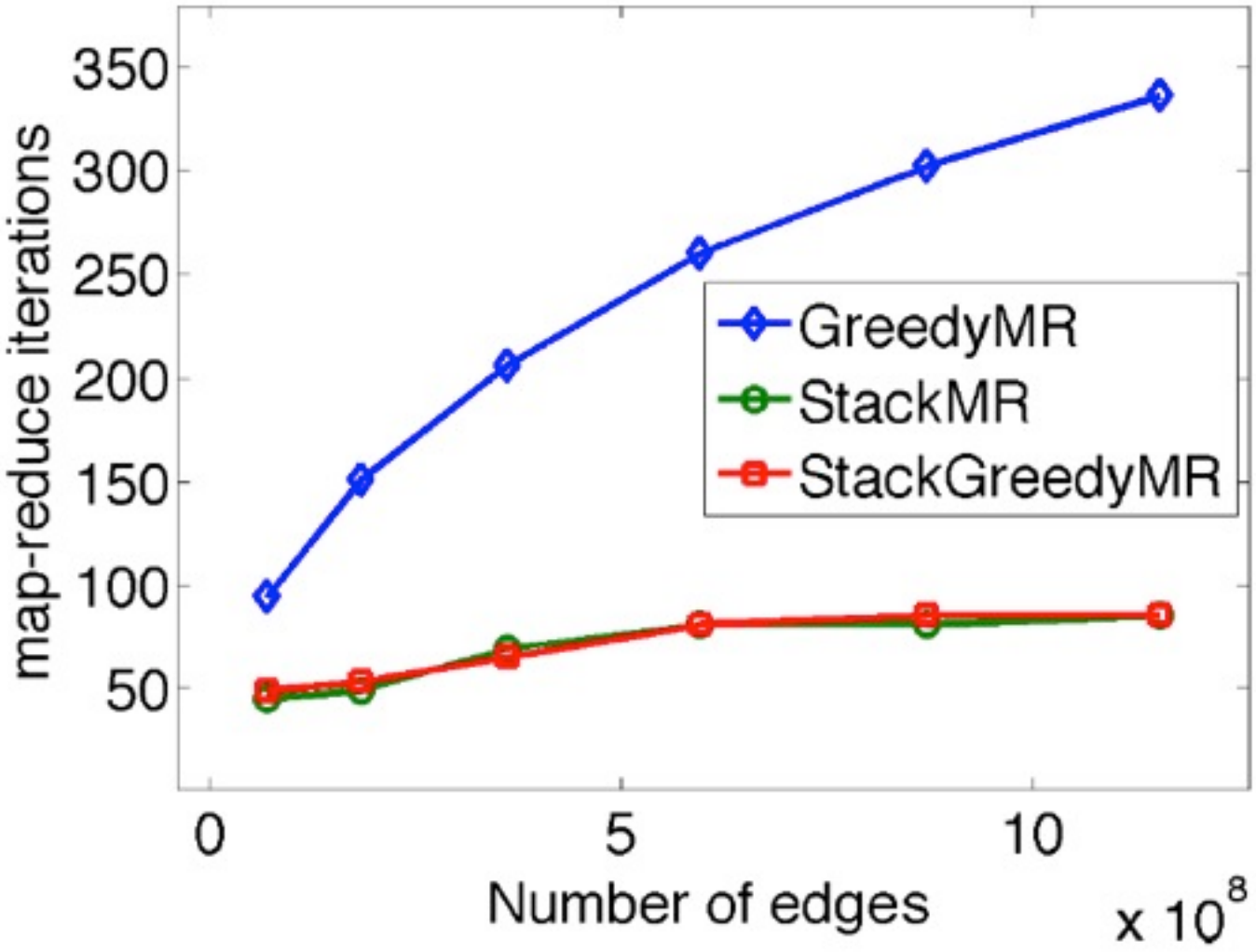
We measure:

- accuracy;
- running time.

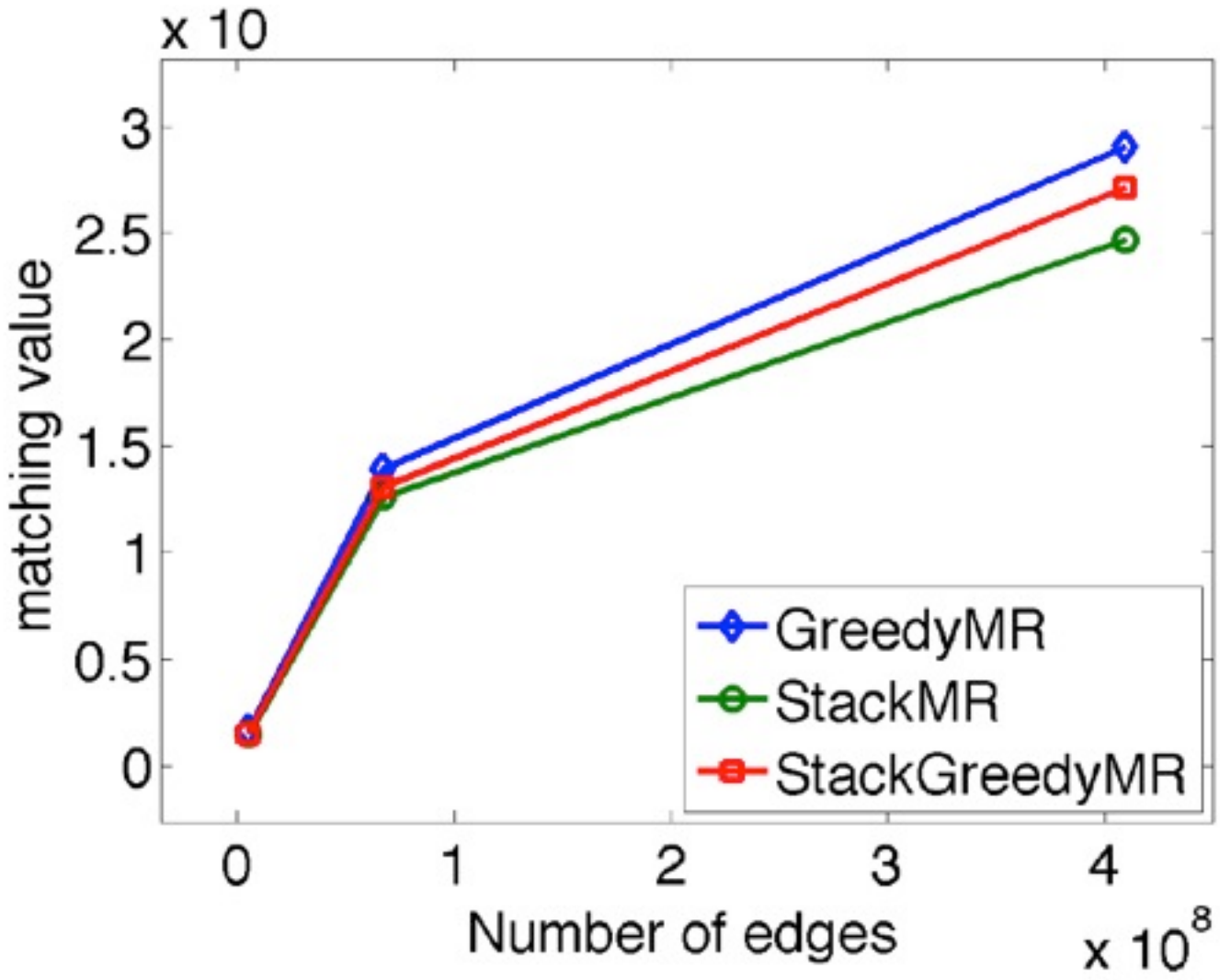
Flickr: Accuracy



Flickr: Running Time



Yahoo! Answers: Accuracy



Yahoo! Answers: running time