Space-Round Tradeoffs for MapReduce Computations

A. Pietracaprina, G. Pucci, F. Silvestri

M. Riondato, E. Upfal

ICS 2012

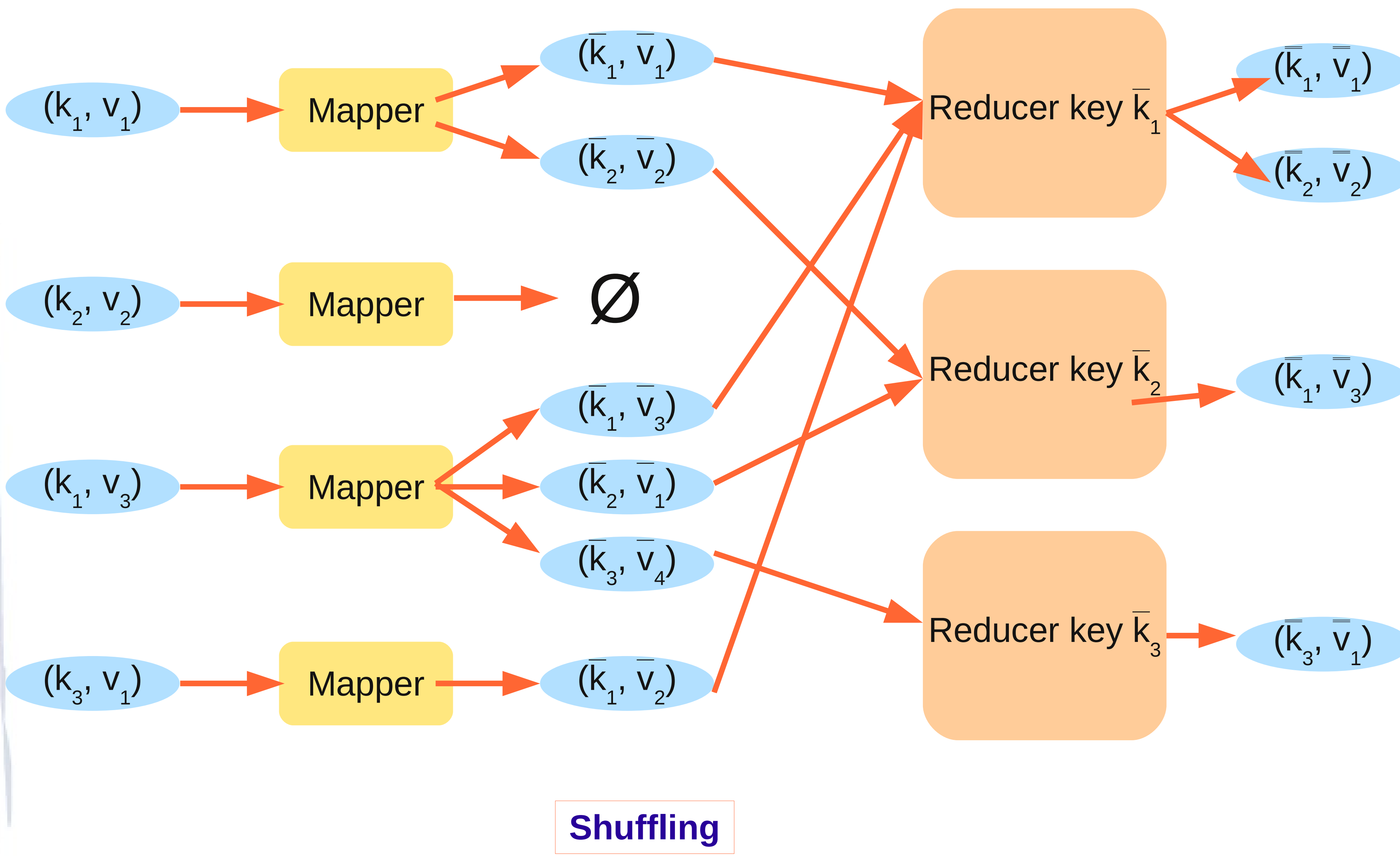
MapReduce

- Introduced in [Dean & Ghemawat, OSDI 2004]
- Programming paradigm for large data sets
- Typically used on clusters of commodity computers
- Widely used in many scenarios: log processing, data-mining, scientific computations,...

MapReduce (2)

- Eases programmer tasks
  - The runtime system manages low-level details
  - Focus on the problem, not on the platform
- Inspired by functional programming
- Algorithm is a sequence of rounds
  - Map/Reduce functions
  -

A MapReduce round



Previous work

- Modeling efforts
  - [Feldman et al, SODA 2008]
  - [Karloff et al, SODA 2010]
  - [Goodrich et al, ISAAC 2011]

- Algorithms
  - Graph problems, e.g. [Suri et al, WWW 2011][Lattanzi et al, SPAA 2011]
  - Clustering, e.g. [Ene et al, KDD 2011]

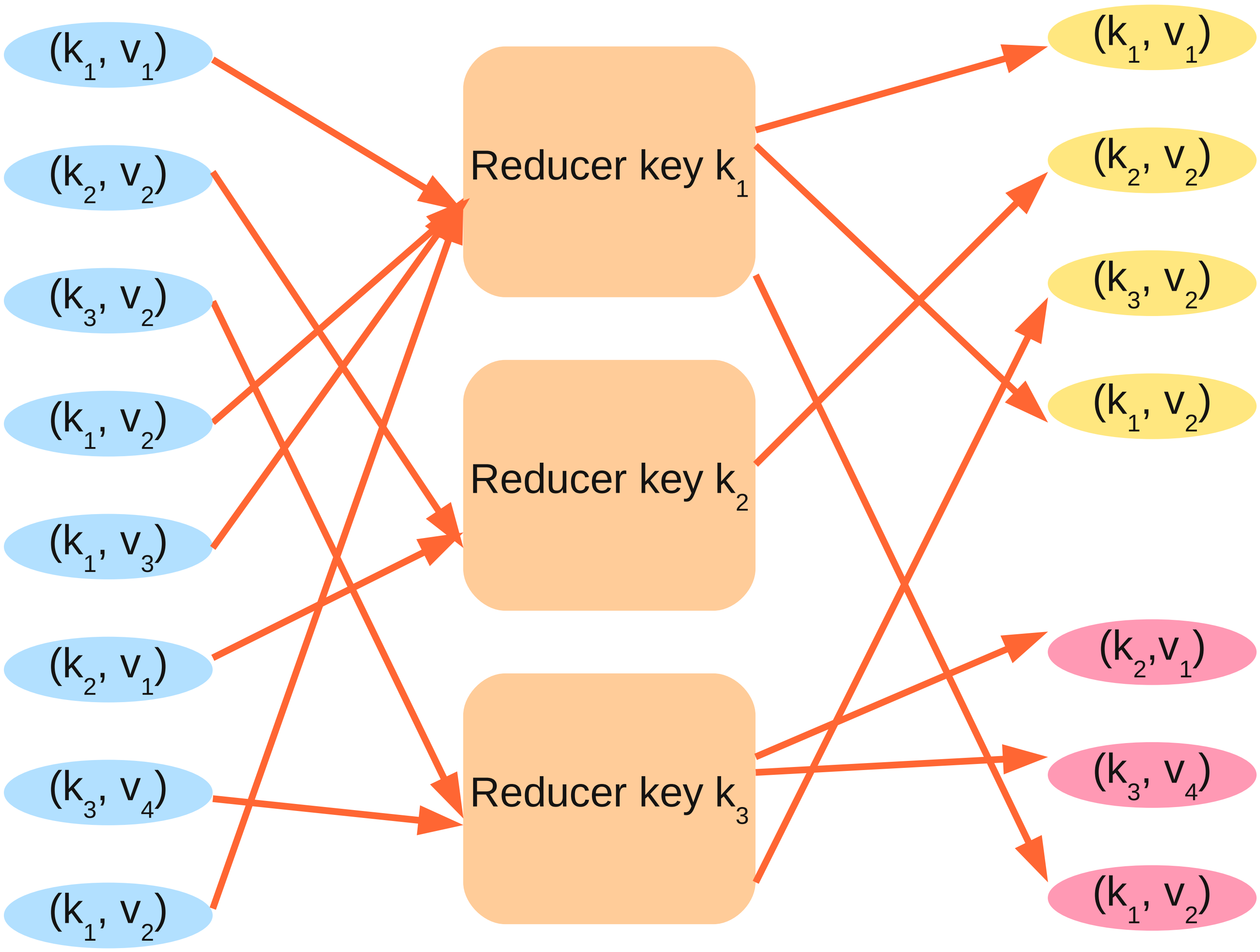
Our results

1. Computational model for MapReduce
   - Overcomes some limitations of previous models
   - Two parameters describing the local and aggregate space constraints
2. Algorithms for sparse/dense matrix multiplication
   - Tradeoffs between performance and space parameters
3. Applications based on matrix multiplication
   - Matrix inversion and matching

The MR(m,M) model

- Based on [Karloff et al, SODA 2010]
- Clear separation between model and underlying infrastructure
- Maintains functional flavor
- No need to distinguish between mappers and reducers
- An MR algorithm is a sequence of rounds

An MR round



(k1, v1)
(k2, v2)
(k3, v2)
(k1, v2)
(k1, v3)
(k2, v1)
(k3, v4)
(k1, v2)
Reducer key k1
Reducer key k2
Reducer key k3
(k1, v1)
(k2, v2)
(k3, v2)
(k1, v2)
(k2,v1)
(k3, v4)
(k1, v2)

Tradeoffs

- Complexity measure: number of rounds
  - Rationale: shuffling is the expensive operation
- Parameters *m* and *M*:
  - ***m:*** max reducer size (limits the number of pairs received by a reducer)
  - ***M:*** max amount of total space (max number of pairs in a round)
  - Allow for a flexible use of parallelism: e.g., *M/m* reducers of size *m*, or *M* reducers of size *O(1)*
- We aim at deriving tradeoffs between space and number of rounds

Matrix multiplication on MR

- Lower and upper bounds for
  - Dense-dense matrix multiplication
  - Spare-sparse matrix multiplication
    - three variants (D1, D2, R1)
    - Estimating density of product matrix
  - Sparse-dense matrix multiplication

- Optimal space-round tradeoffs in many cases

Notation

- A, B, C=AxB: matrices of size $\sqrt{n} \times \sqrt{n}$
- Divide into submatrices of size $\sqrt{m} \times \sqrt{m}$
  - Partition the $(n/m)^{3/2}$ multiplications into $(n/m)^{1/2}$ groups
  - Each submatrix appears once in each group
- $\overline{n}$: number of nonzero entries in A and B
- $\overline{o}$: number of nonzero entries in C (not known!)

Dense-dense case

- Each group requires space $3n$
- In each round: compute multiplications within *M/3n* groups
- Number of rounds

$$O\left(\frac{n^{3/2}}{M\sqrt{m}}+\log_m n\right)$$

- Constant number of rounds if *m=poly(n)* and $M=\Omega(n^{3/2}/\sqrt{m})$

Sparse-sparse: Deterministic D1

- Column-row product: compute all nonzero products between the *i*-th column of A and *i*-th row of B (nonzero products could be < n)
- Compute the $\sqrt{n}$ column-row products into phases
- In each phase:
  - number of column-row products in the phase computed via prefix-sum
  - no more than M nonzero products

Sparse-sparse: Deterministic D1 (2)

- Number of rounds

$$O\left(\frac{\bar{n}\min(\bar{n},\sqrt{n})}{M}\log_m n\right)$$

- Constant number of rounds if *m=poly(n) and M sufficiently* large
- Extends to the sparse-dense case
- Inefficient use of reducer space *m*

Sparse-sparse: Deterministic D2

- Clever implementation of dense-dense algorithm leveraging on the sparsity
- Number of groups in each phase computed through a prefix sum based on the space requirements of involved submatrices
- Number of rounds

$$O\left(\frac{(\bar{n}+\bar{o})\sqrt{n}}{M\sqrt{m}}\log_m n\right)$$

- Constant round complexity if *m=poly(n), M* sufficiently large

Sparse-sparse: Randomized R3

- D2 can be improved if $\bar{o}$ is known
  - Avoid prefix sums by processing $M/(\bar{n}+\bar{o})$ groups per phase
- An approximation to $\bar{o}$ is given by a randomized algorithm
- Number of rounds $O\left(\frac{(\bar{n}+\bar{o})\sqrt{n}}{M\sqrt{m}}+\log_m n\right)$

Density of product matrix

- We use streaming sketches [Bar-Yossef, RANDOM 2002]
  - Data-structure for computing number of distinct values in a stream with small space
- Size of output matrix:
  - For each nonzero product, assign to pair (a_{ik},b_{kj}) the value (i,j)
  - Number of nonzero entries in C = number of distinct values (using sketches)

Lower bounds

- Only semiring operations (no Strassen)
- Matrices of size $\sqrt{n} \, x \sqrt{n}$
- $\overline{n}$ nonzero entries per matrix
- Number of rounds (based on [Hong & Kung, STOC 81])

$$\Omega\left(\frac{\bar{n}\min(\bar{n}, \sqrt{n})}{M\sqrt{m}} + \log_m n\right)$$

- Constant rounds → data replication

Applications

- We use dense-dense matrix multiplication for:
  - Inverse of a triangular matrix in constant rounds
  - Inverse of a general matrix in O(log n) rounds
  - Approximate inverse of a general matrix in O(log n) rounds (and less space)
  - Perfect matching in O(log n) rounds

Conclusion

- Our results provide evidence that nontrivial tradeoffs can be exercised between space requirement and performance
- Future work:
  - Tradeoffs for other problems, e.g. graphs, data-mining
  - Experimental evaluation of the model and algorithms

Thank you!