

Designing Efficient Map-Reduce Algorithms

Review of Map-Reduce

A Common Mistake

Size/Communication Trade-Off

Specific Tradeoffs

Jeffrey D. Ullman
Stanford University



Research Is Joint Work of

- Foto Afrati (NTUA)
- Anish Das Sarma (Google)
- Semih Salihoglu (Stanford)
- JU

Review of Map-Reduce

Mappers and Reducers

Key-Value Pairs

Example Application: Join

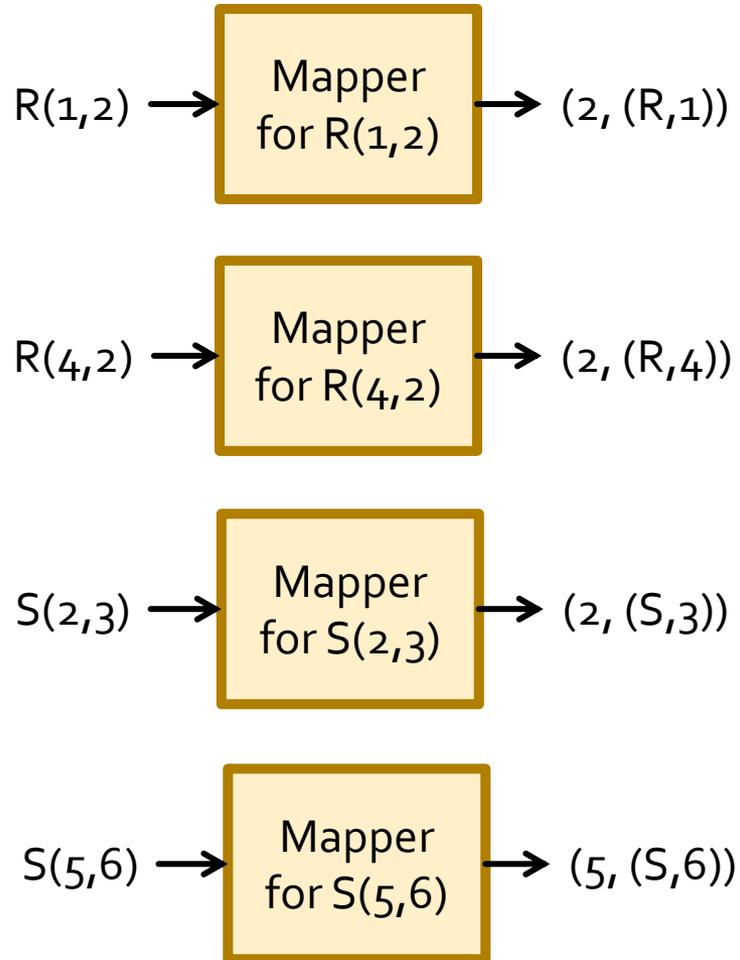
Mappers and Reducers

- Map-Reduce job =
 - Map function (inputs -> key-value pairs) +
 - Reduce function (key and list of values -> outputs).
- Map and Reduce Tasks apply Map or Reduce function to (typically) many of their inputs.
 - Unit of parallelism.
- *Mapper* = application of the Map function to a single input.
- *Reducer* = application of the Reduce function to a single key-(list of values) pair.

Example: Natural Join

- Join of $R(A,B)$ with $S(B,C)$ is the set of tuples (a,b,c) such that (a,b) is in R and (b,c) is in S .
- Mappers need to send $R(a,b)$ and $S(b,c)$ to the same reducer, so they can be joined there.
- **Mapper output:** key = B-value, value = relation and other component (A or C).
 - **Example:** $R(1,2) \rightarrow (2, (R,1))$
 $S(2,3) \rightarrow (2, (S,3))$

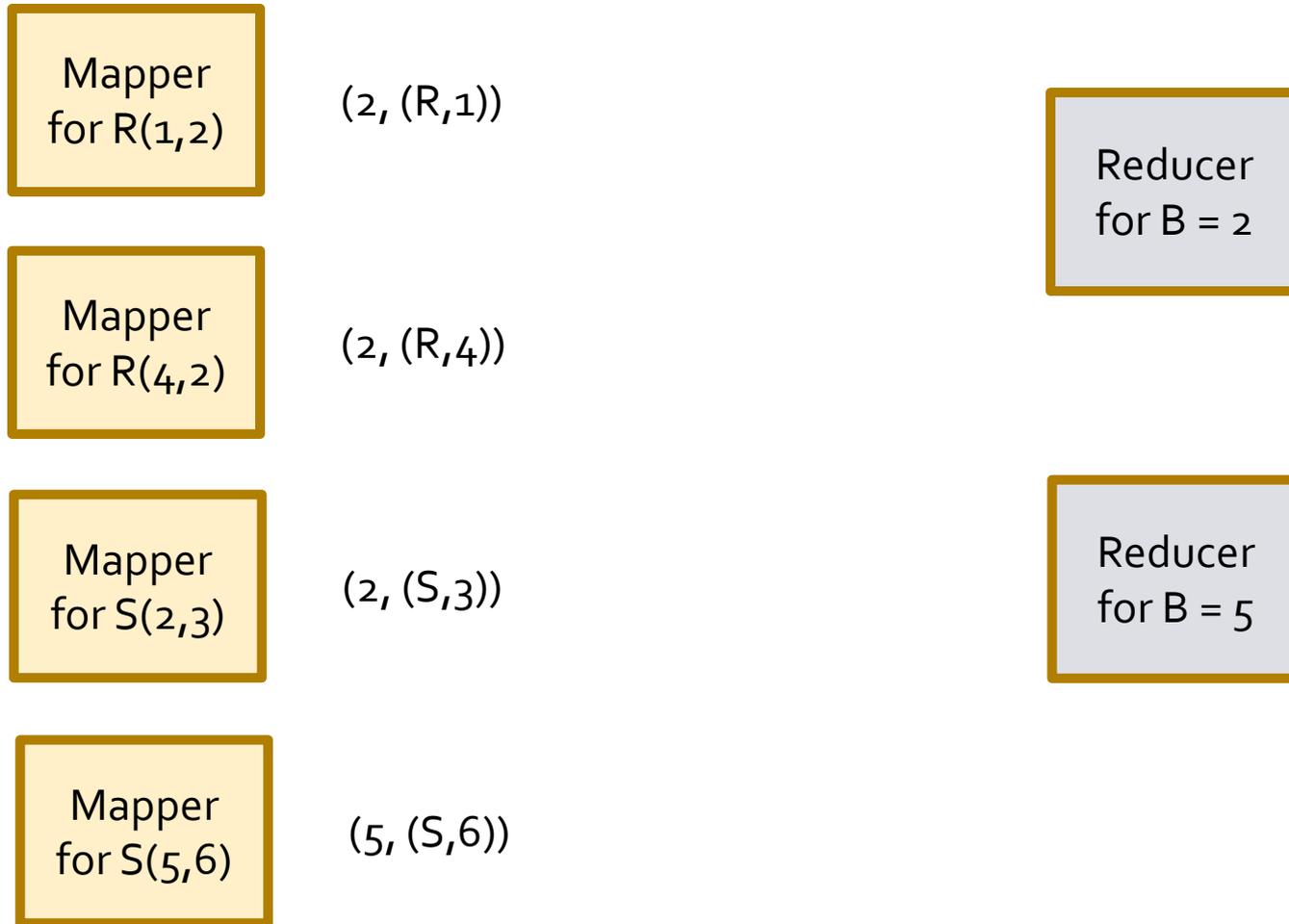
Mapping Tuples



Grouping Phase

- There is a reducer for each key.
- Every key-value pair generated by any mapper is sent to the reducer for its key.

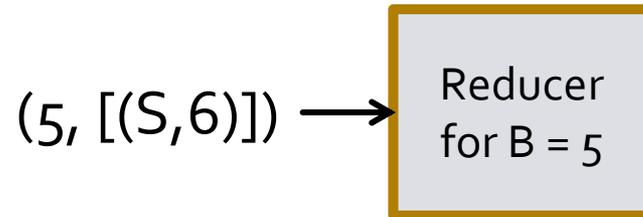
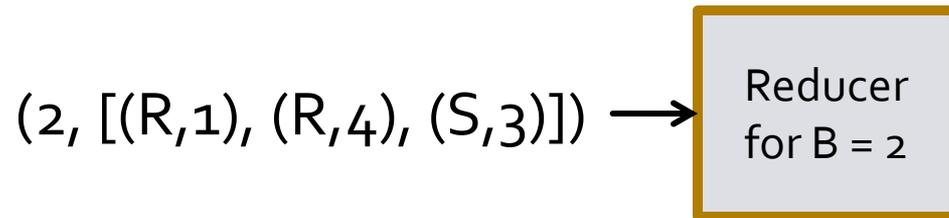
Mapping Tuples



Constructing Value-Lists

- The input to each reducer is organized by the system into a pair:
 - The key.
 - The list of values associated with that key.

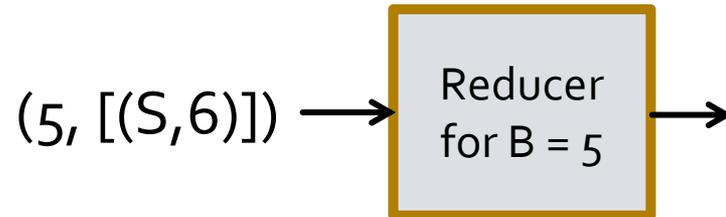
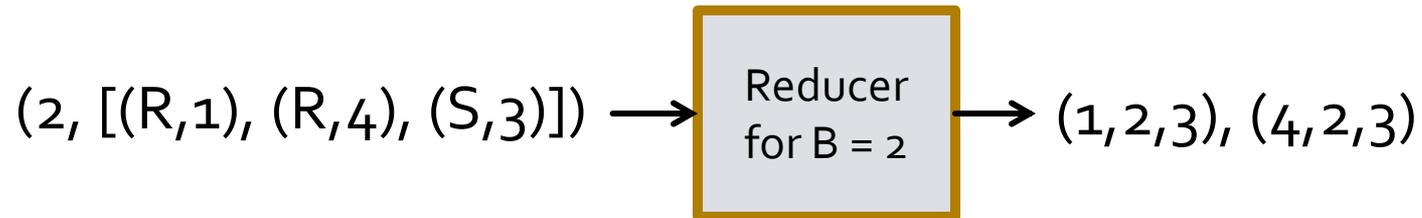
The Value-List Format



The Reduce Function for Join

- Given key b and a list of values that are either (R, a_i) or (S, c_j) , output each triple (a_i, b, c_j) .
 - Thus, the number of outputs made by a reducer is the product of the number of R 's on the list and the number of S 's on the list.

Output of the Reducers



Motivating Example

The Drug Interaction Problem
A Failed Attempt
Lowering the Communication

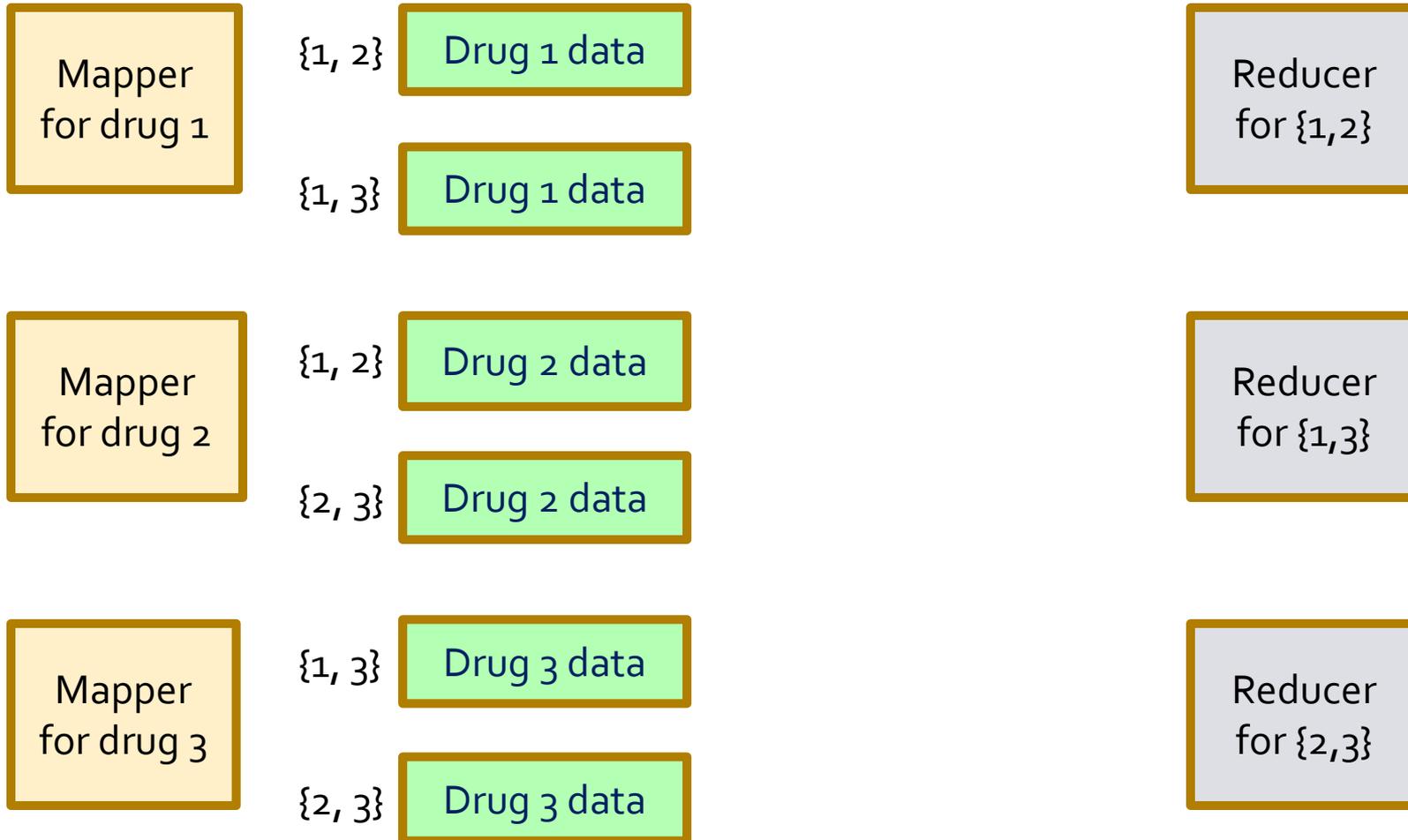
The Drug-Interaction Problem

- Data consists of records for 3000 drugs.
 - List of patients taking, dates, diagnoses.
 - About 1M of data per drug.
- Problem is to find drug interactions.
 - **Example**: two drugs that when taken together increase the risk of heart attack.
- Must examine each pair of drugs and compare their data.

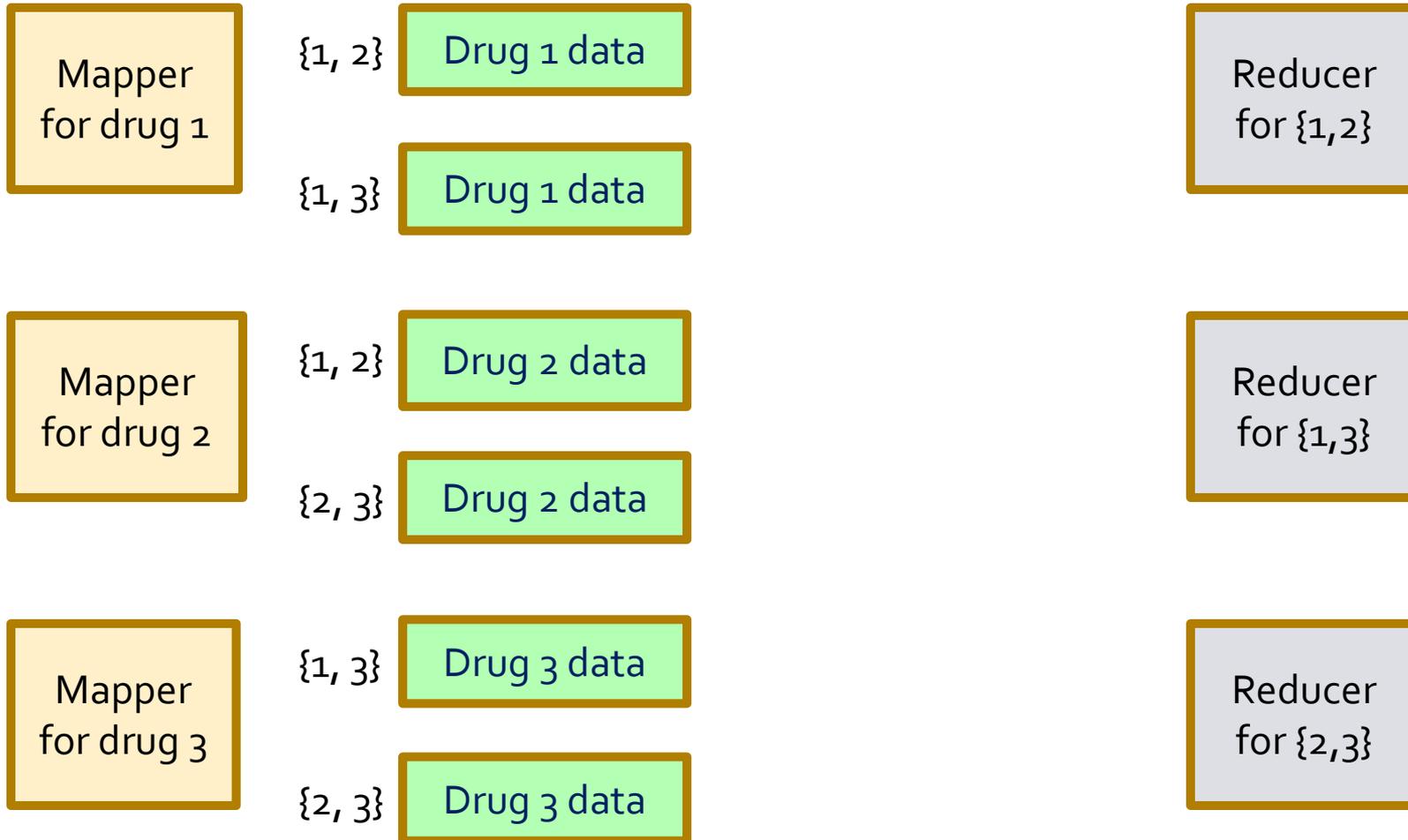
Initial Map-Reduce Algorithm

- The first attempt used the following plan:
 - Key = set of two drugs $\{i, j\}$.
 - Value = the record for one of these drugs.
- Given drug i and its record R_i , the mapper generates all key-value pairs $(\{i, j\}, R_i)$, where j is any other drug besides i .
- Each reducer receives its key and a list of the two records for that pair: $(\{i, j\}, [R_i, R_j])$.

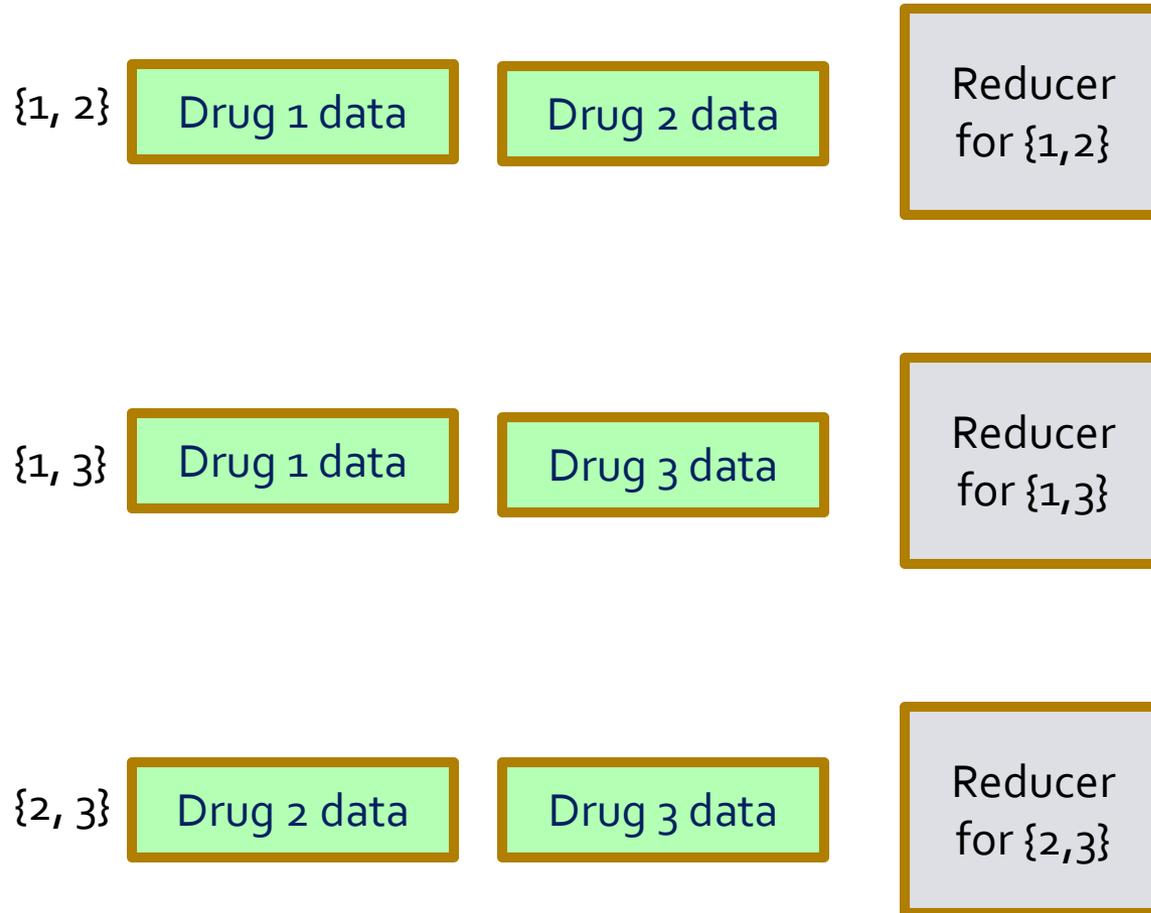
Example: Three Drugs



Example: Three Drugs



Example: Three Drugs



What Went Wrong?

- 3000 drugs
- times 2999 key-value pairs per drug
- times 1,000,000 bytes per key-value pair
- = 9 terabytes communicated over a 1Gb Ethernet
- = 90,000 seconds of network use.

A Better Approach

- Suppose we group the drugs into 30 groups of 100 drugs each.
 - Say $G_1 =$ drugs 1-100, $G_2 =$ drugs 101-200, ..., $G_{30} =$ drugs 2901-3000.
 - Let $g(i) =$ the number of the group into which drug i goes.

The Map Function

- A key is a set of two group numbers.
- The mapper for drug i produces 29 key-value pairs.
 - Each key is the set containing $g(i)$ and one of the other group numbers.
 - The value is a pair consisting of the drug number i and the megabyte-long record for drug i .

The Reduce Function

- The reducer for pair of groups $\{m, n\}$ gets that key and a list of 200 drug records – the drugs belonging to groups m and n .
- Its job is to compare each record from group m with each record from group n .
 - **Special case:** also compare records in group n with each other, if $m = n+1$ or if $n = 30$ and $m = 1$.
- Notice each pair of records is compared at exactly one reducer, so the total computation is not increased.

The New Communication Cost

- The big difference is in the communication requirement.
- Now, each of 3000 drugs' 1MB records is replicated 29 times.
 - Communication cost = 87GB, vs. 9TB.

Theory of Map-Reduce Algorithms

Reducer Size

Replication Rate

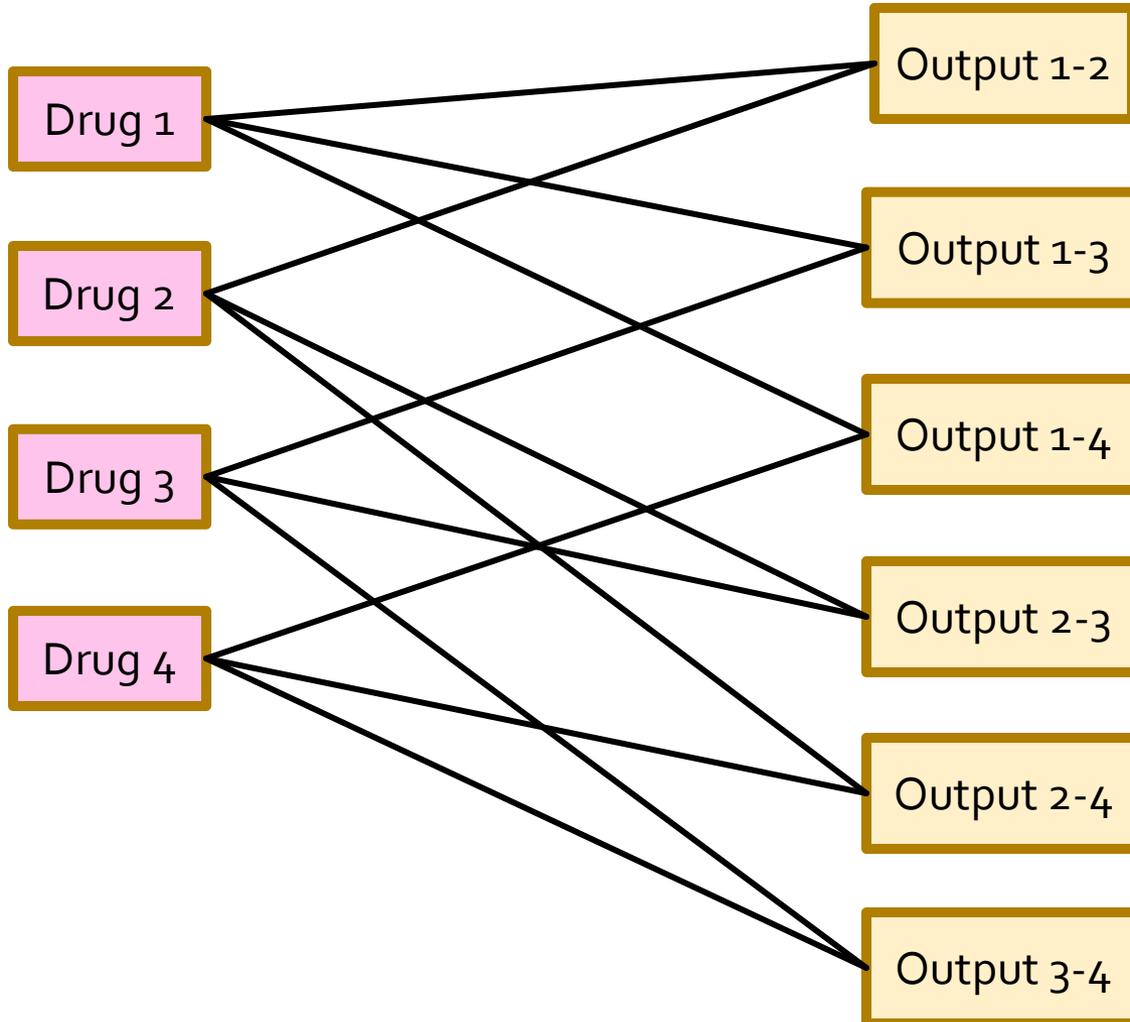
Mapping Schemas

Lower Bounds

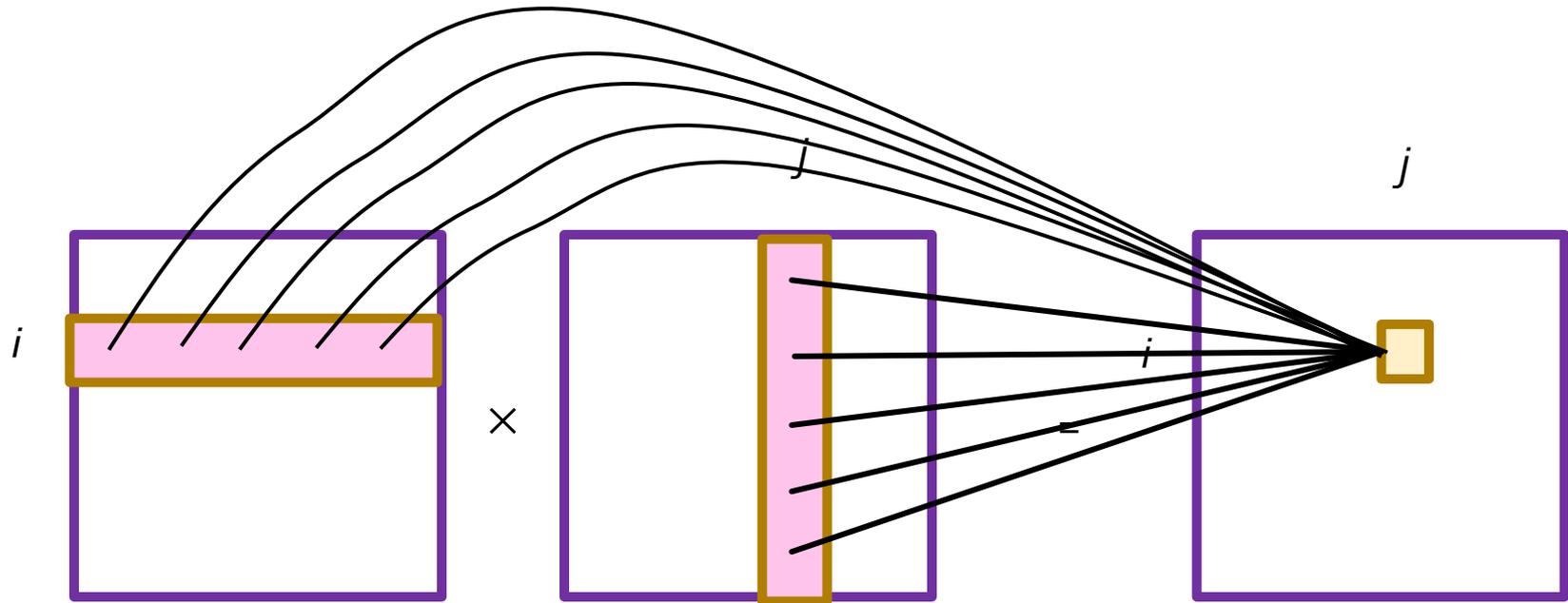
A Model for Map-Reduce Algorithms

1. A set of *inputs*.
 - **Example**: the drug records.
2. A set of *outputs*.
 - **Example**: One output for each pair of drugs.
3. A many-many relationship between each output and the inputs needed to compute it.
 - **Example**: The output for the pair of drugs $\{i, j\}$ is related to inputs i and j .

Example: Drug Inputs/Outputs



Example: Matrix Multiplication



Reducer Size

- *Reducer size*, denoted q , is the maximum number of inputs that a given reducer can have.
 - I.e., the length of the value list.
- Limit might be based on how many inputs can be handled in main memory.
- Or: make q low to force lots of parallelism.

Replication Rate

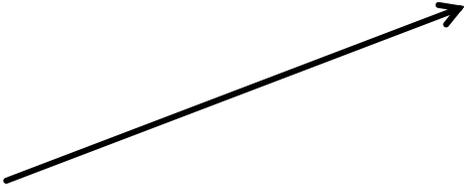
- The average number of key-value pairs created by each mapper is the *replication rate*.
 - Denoted r .
- Represents the communication cost per input.

Example: Drug Interaction

- Suppose we use g groups and d drugs.
- A reducer needs two groups, so $q = 2d/g$.
- Each of the d inputs is sent to $g-1$ reducers, or approximately $r = g$.
- Replace g by r in $q = 2d/g$ to get $r = 2d/q$.

Tradeoff!

The bigger the reducers,
the less communication.



Upper and Lower Bounds on r

- What we did gives an upper bound on r as a function of q .
- A solid investigation of map-reduce algorithms for a problem includes lower bounds.
 - Proofs that you cannot have lower r for a given q .

Proofs Need Mapping Schemas

- A *mapping schema* for a problem and a reducer size q is an assignment of inputs to sets of reducers, with two conditions:
 1. No reducer is assigned more than q inputs.
 2. For every output, there is some reducer that receives all of the inputs associated with that output.
 - Say the reducer *covers* the output.

Mapping Schemas – (2)

- Every map-reduce algorithm has a mapping schema.
- The requirement that there be a mapping schema is what distinguishes map-reduce algorithms from general parallel algorithms.

Example: Drug Interactions

- d drugs, reducer size q .
- No reducer can cover more than $q^2/2$ outputs.
- There are $d^2/2$ outputs that must be covered.
- Therefore, we need at least d^2/q^2 reducers.
- Each reducer gets q inputs, so replication r is at least $q(d^2/q^2)/d = d/q$.
- Half the r from the algorithm we described.

Inputs per
reducer

Number of
reducers

Divided by
number of
inputs

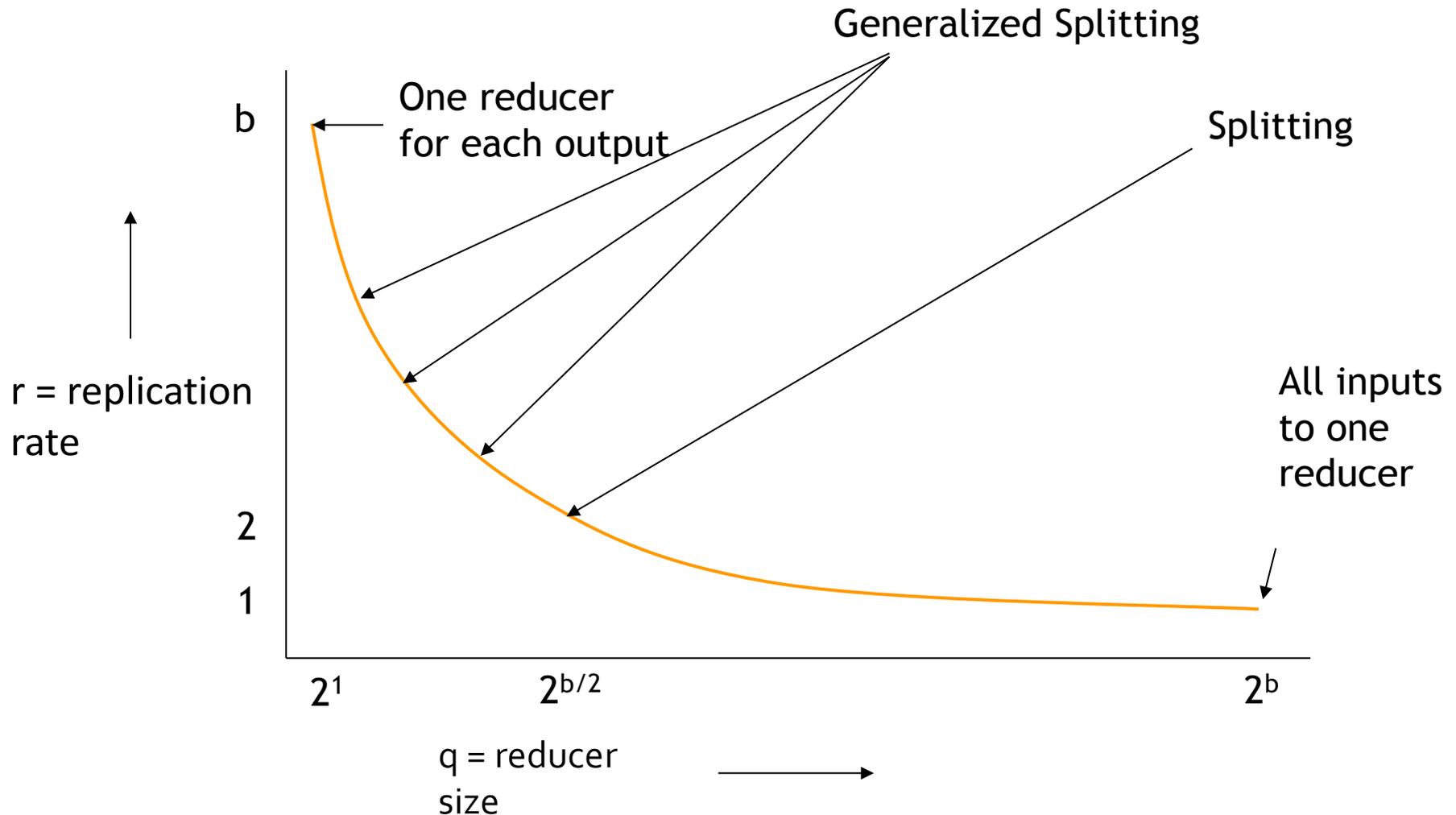
Specific Problems

Hamming Distance 1
Matrix Multiplication

Definition of HD₁ Problem

- Given a set of bit strings of length b , find all those that differ in exactly one bit.
- **Theorem:** $r \geq b/\log_2 q$.

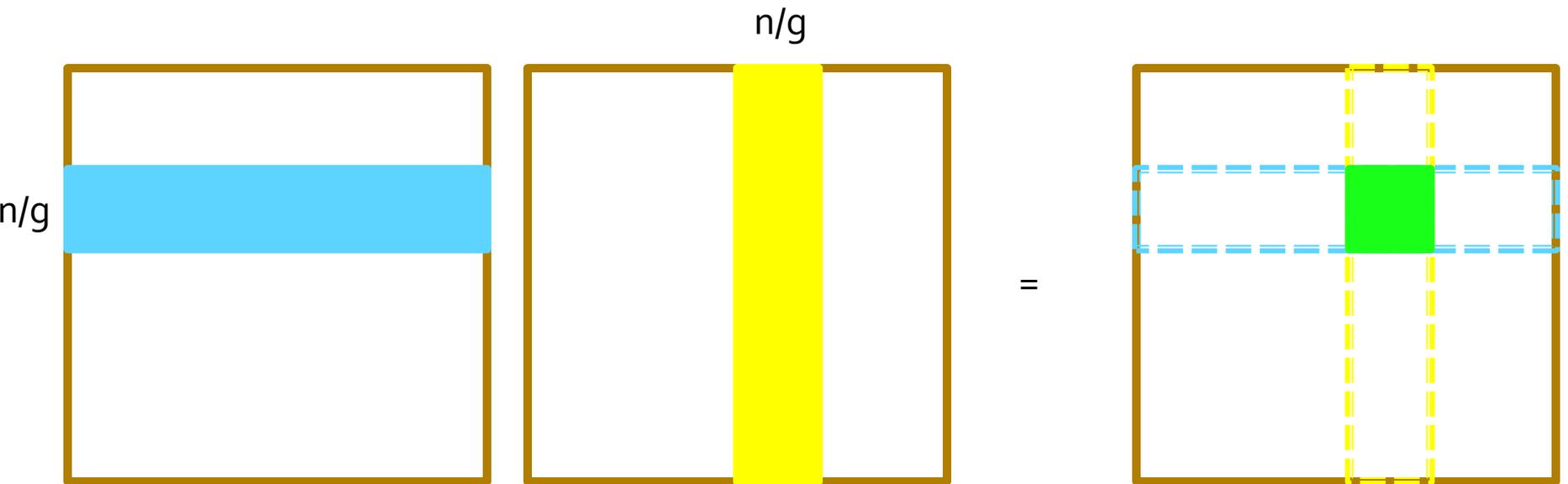
Algorithms Matching Lower Bound



Matrix Multiplication

- Assume $n \times n$ matrices $AB = C$.
- **Theorem:** For matrix multiplication, $r \geq 2n^2/q$.

Matching Algorithm

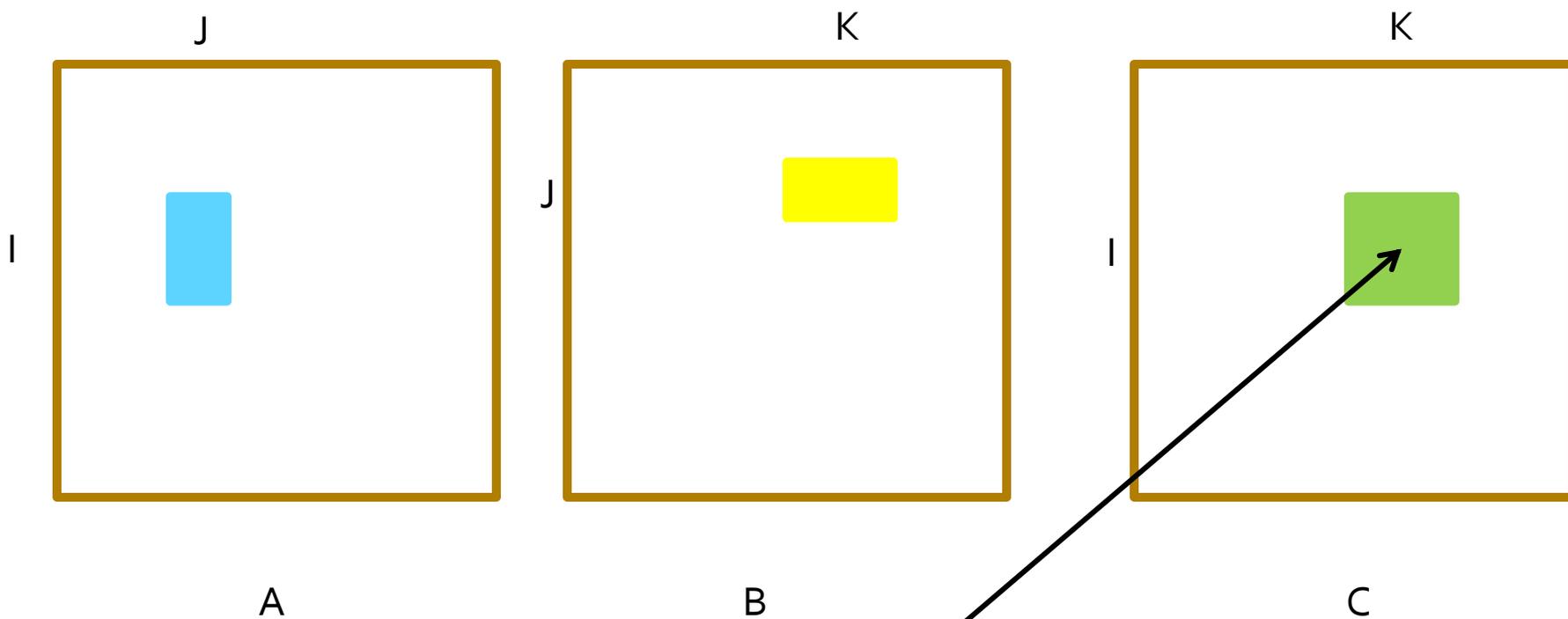


Divide rows of A and columns
of B into g groups gives
 $r = g = 2n^2/q$

Two-Job Map-Reduce Algorithm

- **A better way:** use two map-reduce jobs.
- **Job 1:** Divide both input matrices into rectangles.
 - Reducer takes two rectangles and produces partial sums of certain outputs.
- **Job 2:** Sum the partial sums.

Picture of First Job



For i in I and k in K , contribution
is $\sum_{j \text{ in } J} A_{ij} \times B_{jk}$

Comparison: Communication Cost

- **One-job method**: Total communication = $4n^4/q$.
- **Two-job method** Total communication = $4n^3/\sqrt{q}$.
 - Since $q < n^2$ (or we really have a serial implementation), two jobs wins!

Summary

- Represent problems by mapping schemas
- Get upper bounds on number of covered outputs as a function of reducer size.
- Turn these into lower bounds on replication rate as a function of reducer size.
- For $HD = 1$ problem: exact match between upper and lower bounds.
- 1-job matrix multiplication analyzed exactly.
- But 2-job MM yields better total communication.