COMP 322: Fundamentals of Parallel Programming

Lecture 8: Map Reduce

Vivek Sarkar, Eric Allen
Department of Computer Science, Rice University

Contact email: vsarkar@rice.edu

https://wiki.rice.edu/confluence/display/PARPROG/COMP322

Worksheet #7 Solution: Identifying Data Races

Identify as many “racy” statement pairs as you can in the pseudoccode shown on the right, e.g., (Si, Sj) if there is a potential data race between Si and Sj

(S2,S3)
(S2,S5)
(S2,S6)
(S2,S7)
(S2,S10)
(S2,S11)
(S2,S12)
(S3,S5)
(S3,S6)
(S3,S7)
(S3,S10)
(S3,S11)
(S3,S12)

<u>Example parallel program:</u>

```
1. p.x = 0; q = p;
2. async p.x = 1; // Task T1
3. async p.x = 2; // Task T2
4. async { // Task T3
5.   System.out.println("First read = " + p.x);
6.   System.out.println("Second read = " + p.x);
7.   System.out.println("Third read = " + p.x)
8. }
9. async { // Task T4
10.   System.out.println("First read = " + p.x);
11.   System.out.println("Second read = " + q.x);
12.   System.out.println("Third read = " + p.x);
13.}
```

Parallelism is the dominant technology trend in Cloud Computing

Software

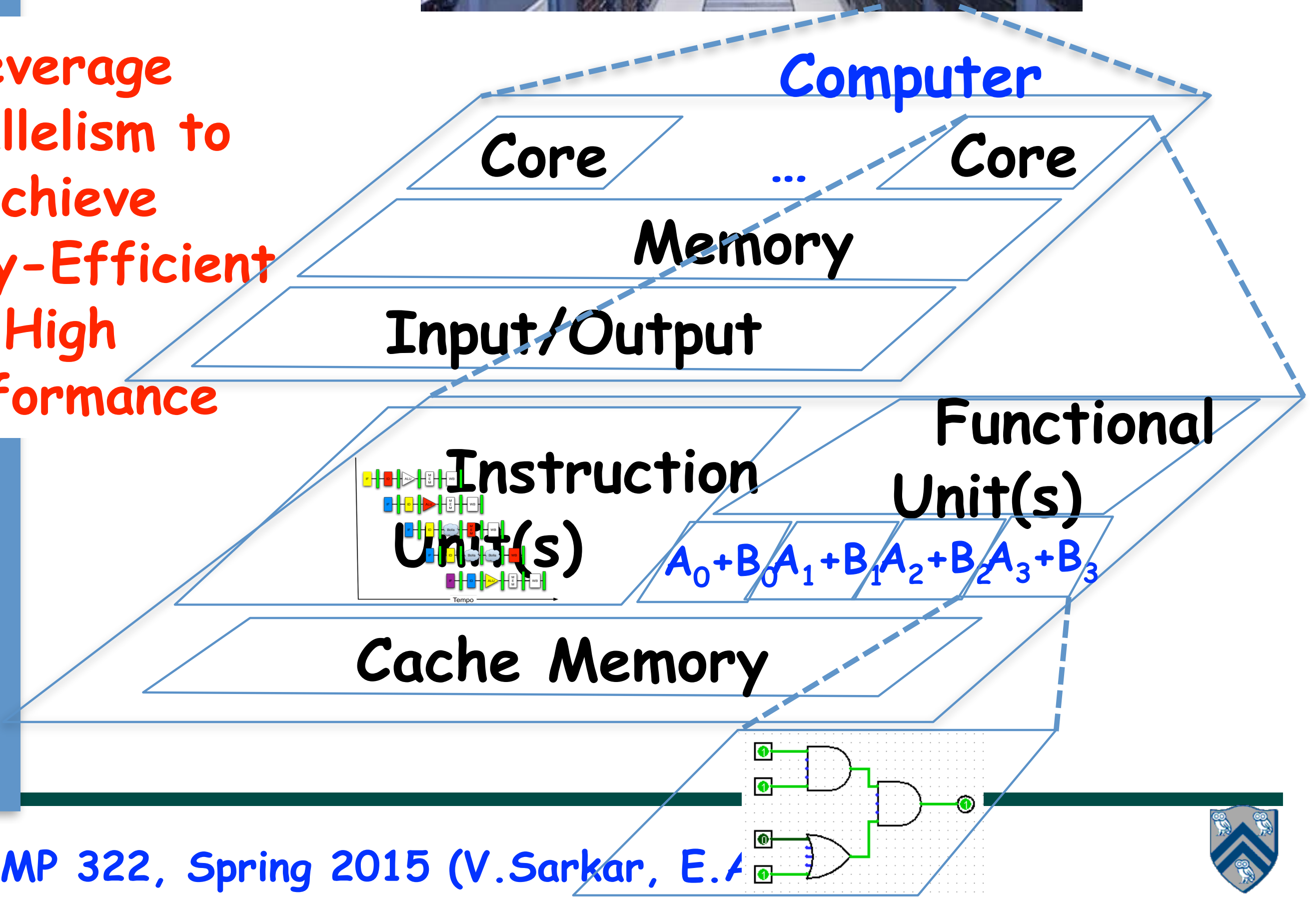
- **Parallel Requests**
  - Assigned to computer
  - e.g., Search "Rice Marching Owl Band"
- **Parallel Threads**
  - Assigned to core
  - e.g., Lookup, Ads
- **Parallel Instrs**
  - >1 instruction/cycle
  - e.g., 5 pipelined instructions
- **Parallel Data**
  - >1 data access/cycle
  - e.g., Load of 4 consecutive words

Hardware

Warehouse Scale Computer

Smart Phone

Leverage Parallelism to Achieve Energy-Efficient High Performance



Parallelism enables "Cloud Computing" as a Utility

- Offers computing, storage, communication at pennies per hour
- No premium to scale:

    1000 computers @ 1 hour
    = 1 computer @ 1000 hours

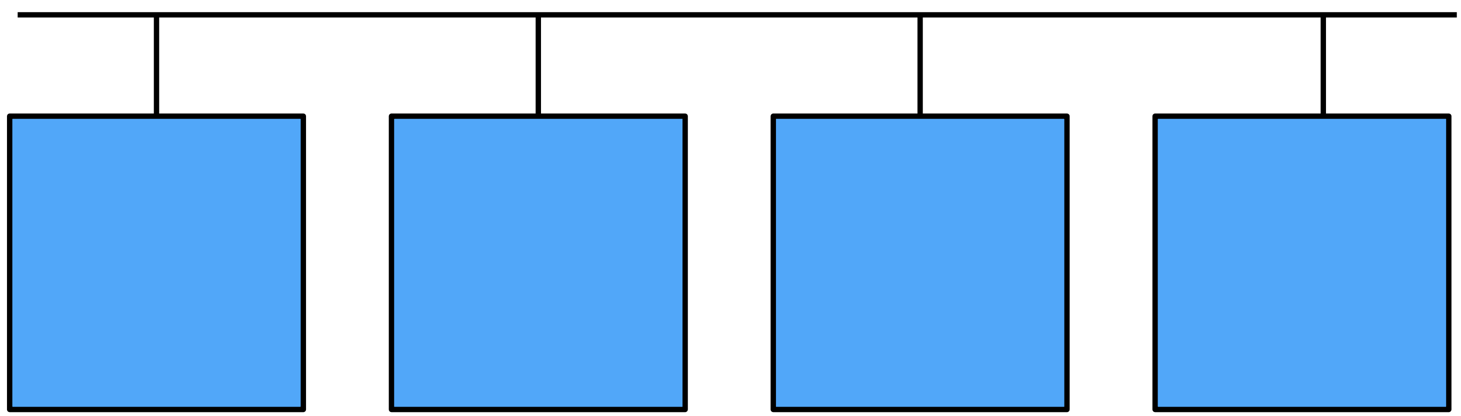
- Illusion of infinite scalability to cloud user
    - As many computers as you can afford
- Leading examples: Amazon Web Services (AWS), Google App Engine, Microsoft Azure
    - Economies of scale pushed down datacenter costs by factors of 3-8X
    - Traditional datacenters utilized 10% - 20%
    - Make profit offering pay-as-you-go use service at less than your costs for as many computers as you need
    - Strategic capability for company's needs
- Challenge: portable and scalable parallelism at cloud scale
    - One solution: leverage functional programming with Map-Reduce pattern

Streaming data requirements have skyrocketed

- AT&T processes roughly 30 petabytes per day through its telecommunications network
- Google processed roughly 24 petabytes per day in 2009
- Facebook, Amazon, Twitter, etc, have comparable throughputs
- Two Sigma maintains over 100 teraflops of private computing power, continuously computing over 11 petabytes of quantitative data
- (By comparison, the IBM Watson knowledge base stored roughly 4 terabytes of data when winning at Jeopardy)

Parallelism enables processing of big data

- Continuously streaming data needs to be processed at least as fast as it is accumulated, or we will never catch up
- The bottleneck in processing very large data sets is dominated by the speed of disk access
- More processors accessing more disks enables faster processing



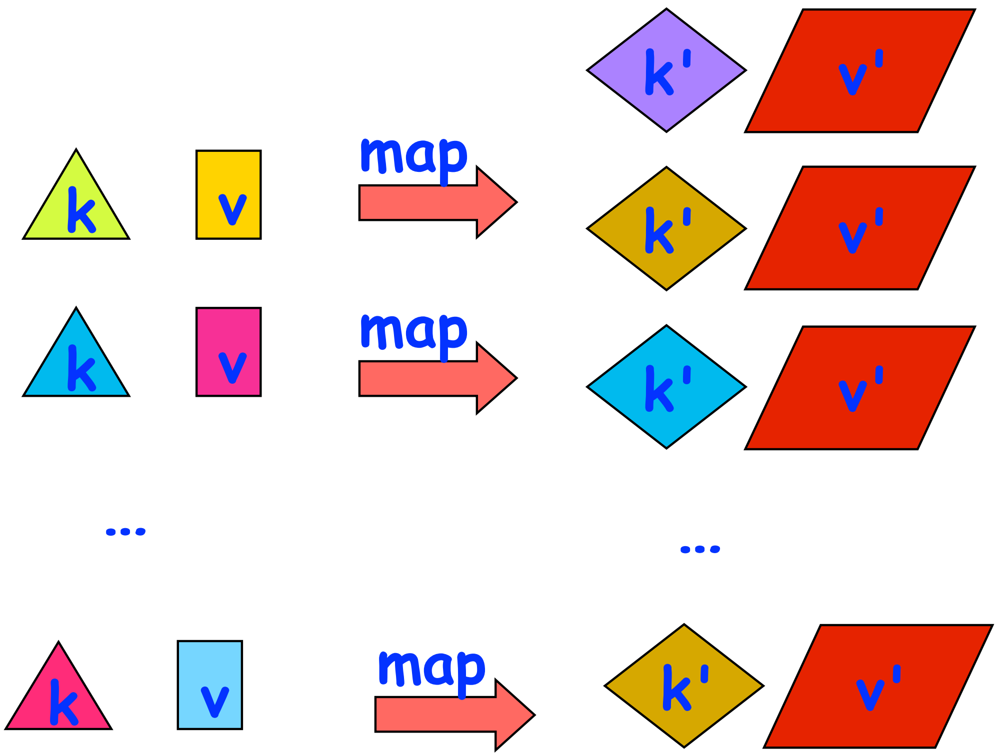
MapReduce Pattern

- Apply **Map** function **f** to user supplied record of key-value pairs
- Compute set of intermediate key/value pairs
- Apply **Reduce** operation **g** to all values that share same key to combine derived data properly
  - Often produces smaller set of values
- User supplies Map and Reduce operations in functional model so that the system can parallelize them, and also re-execute them for fault tolerance

MapReduce: The Map Step

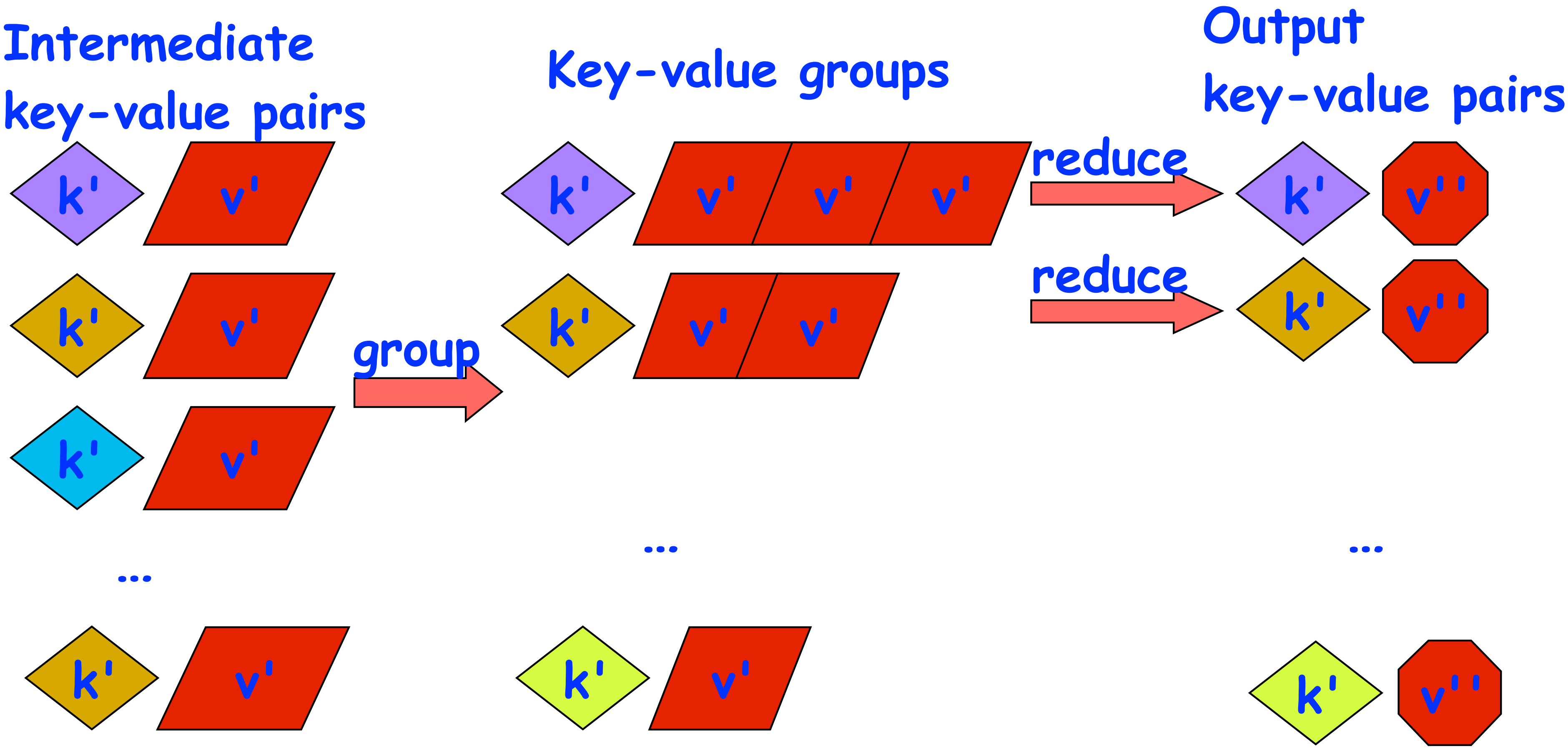
Input set of key-value pairs

Flattened intermediate set of key-value pairs



Source: http://infolab.stanford.edu/~ullman/mining/2009/mapreduce.ppt

MapReduce: The Reduce Step



Source: http://infolab.stanford.edu/~ullman/mining/2009/mapreduce.ppt

Map Reduce: Summary

- Input set is of the form {(k1, v1), . . . (kn, vn)}, where (ki, vi) consists of a key, ki, and a value, vi.
  - Assume that the key and value objects are immutable, and that equality comparison is well defined on all key objects.
- Map function f generates sets of intermediate key-value pairs, f(ki,vi) = {(k1' ,v1'),...(km',vm')}. The kj' keys can be different from ki key in the input of the map function.
  - Assume that a flatten operation is performed as a post-pass after the map operations, so as to avoid dealing with a set of sets.
- Reduce operation groups together intermediate key-value pairs, {(k', vj')} with the same k', and generates a reduced key-value pair, (k',v''), for each such k', using reduce function g

Google Uses MapReduce For …

- Web crawl: Find outgoing links from HTML documents, aggregate by target document
- Google Search: Generating inverted index files using a compression scheme
- Google Earth: Stitching overlapping satellite images to remove seams and to select high-quality imagery
- Google Maps: Processing all road segments on Earth and render map tile images that display segments
- More than 10,000 MR programs at Google in 4 years, run 100,000 MR jobs per day (2008)

Map/Reduce: State of Practice

- **Apache Hadoop now dominates use of the Map/Reduce framework**
- **Often, hadoop map/reduce functions are *no longer written directly***
  - —***Instead, a user writes a query in a very high level language and uses another tool to compile the query into map/reduce functions!***
    - ***Hive (another Apache project) compiles SQL queries into map/reduce***
    - ***Pig (yet another Apache project) compiles direct relational algebra into map/reduce***

Map/Reduce: State of Practice

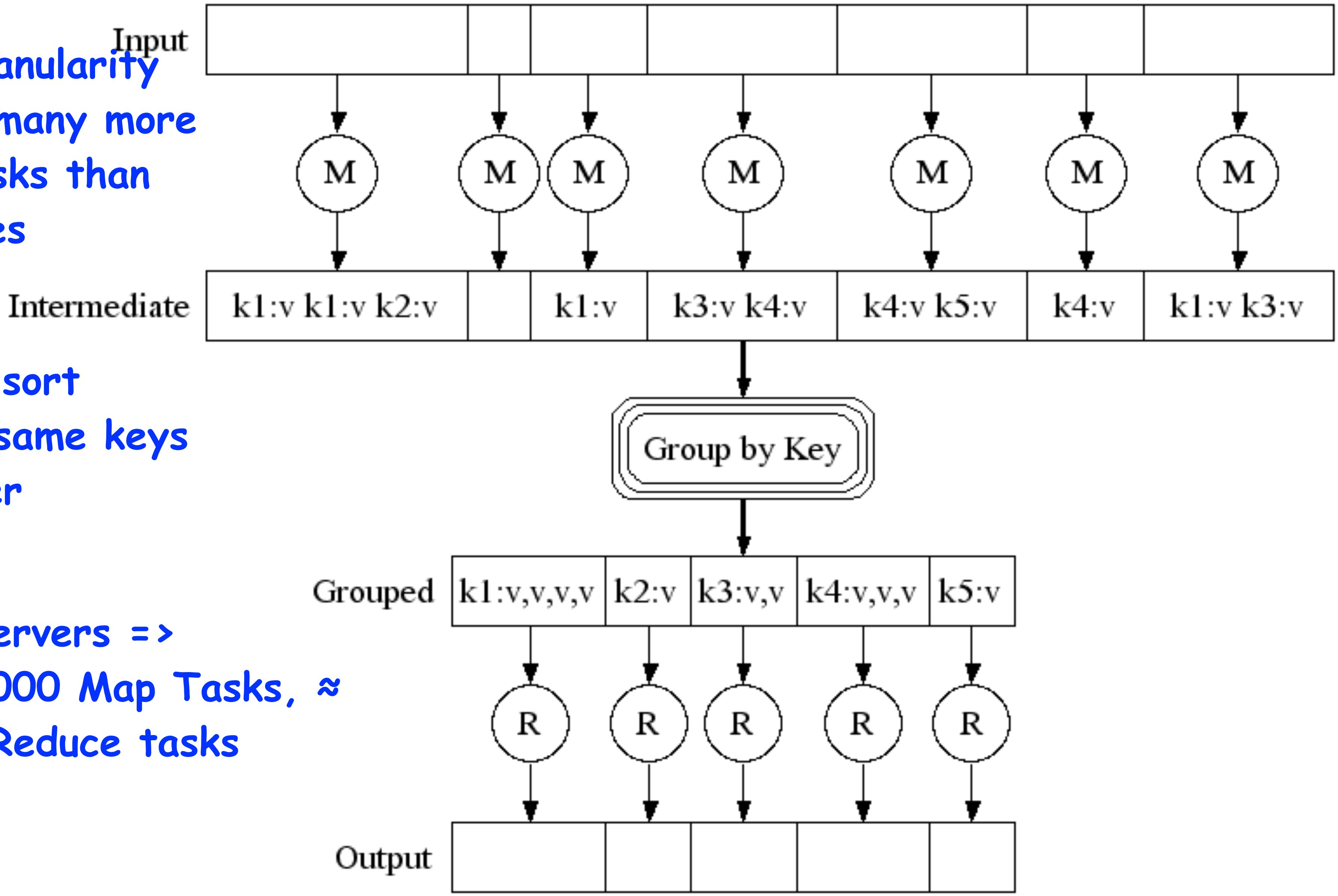
- Eventually, users started realizing that a much larger class of algorithms could be expressed as an iterative sequence of map/reduce operations
  - —Many machine learning algorithms fall into this category
- Tools started to emerge to enable easy expression of multiple map/reduce operations, along with smart scheduling
- Apache Spark: General purpose functional programming over a cluster
  - —Caches results of map/reduce operations in memory so they can be used on subsequent iterations
  - —Tends to be 10-100 times faster than Hadoop for many applications

MapReduce Execution

Fine granularity tasks: many more map tasks than machines

Bucket sort to get same keys together

2000 servers =>
≈ 200,000 Map Tasks, ≈ 5,000 Reduce tasks



WordCount example

Input: set of words

Output: set of (word,count) pairs

Algorithm:

1. For each input word W, emit (W, 1) as a key-value pair (map step).
2. Group together all key-value pairs with the same key (reduce step).
3. Perform a sum reduction on all values with the same key(reduce step).

- All map operations in step 1 can execute in parallel with only local data accesses
- Step 2 may involve a major reshuffle of data as all key-value pairs with the same key are grouped together.
- Step 3 performs a standard reduction algorithm for all values with the same key, and in parallel for different keys.

PseudoCode for WordCount

```
map(String input_key, String input_value):
  // input_key: document name
  // input_value: document contents
  for each word w in input_value:
    EmitIntermediate(w, "1"); // Produce count of words

reduce(String output_key, Iterator intermediate_values):
  // output_key: a word
  // intermediate_values: a list of counts
  int result = 0;
  for each v in intermediate_values:
    result += ParseInt(v); // get integer from key-value
  Emit(AsString(result));
```

Example Execution of WordCount Program