MapReduce: Simplified Data Processing on Large Clusters

Jeff Dean and Sanjay Ghemawat, Google, Inc.

Presenter: Xiaoying Wang

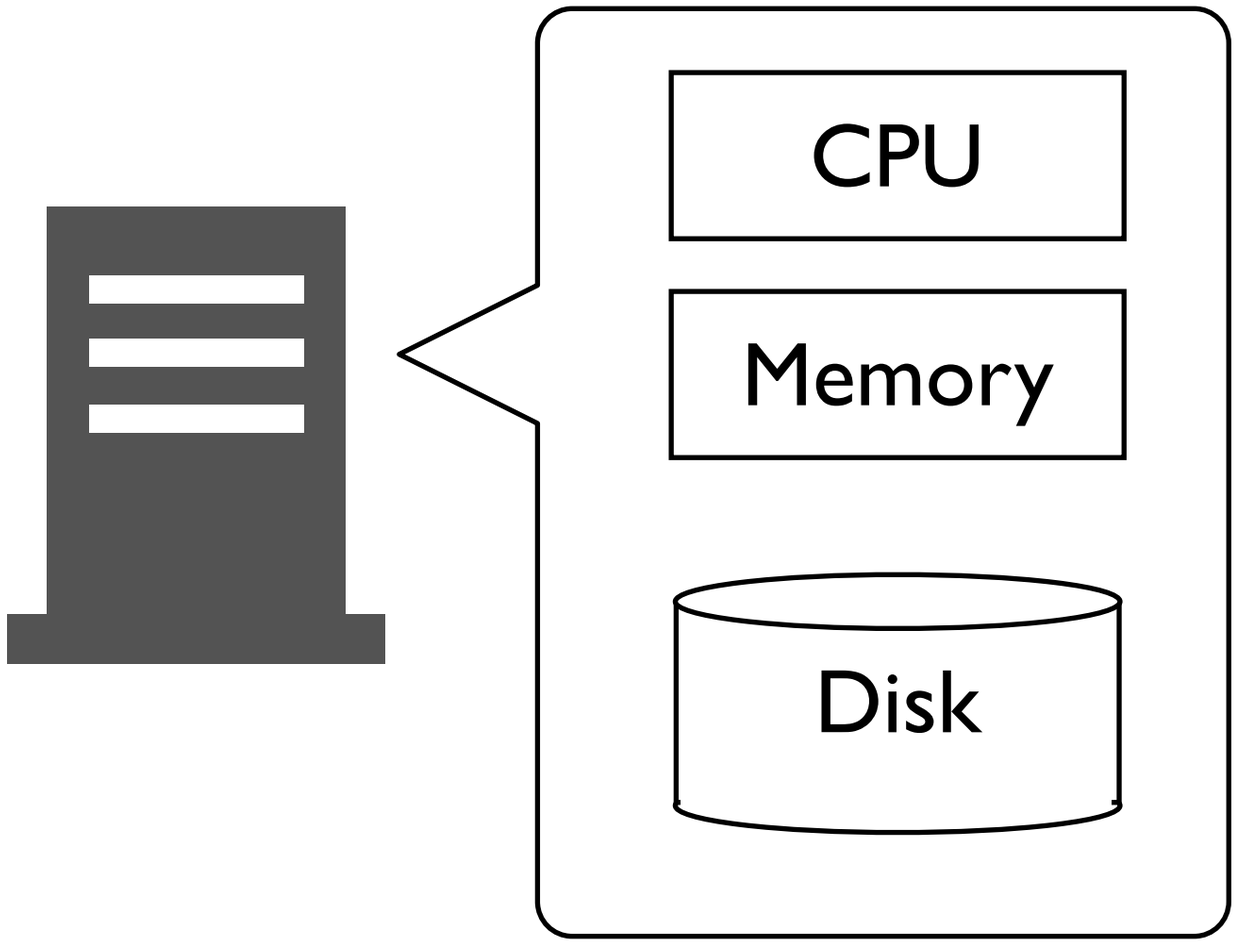
MapReduce: A programming model and an associated implementation for processing and generating large datasets

- Background
- Programming Model Definition
- Implementation & Refinement

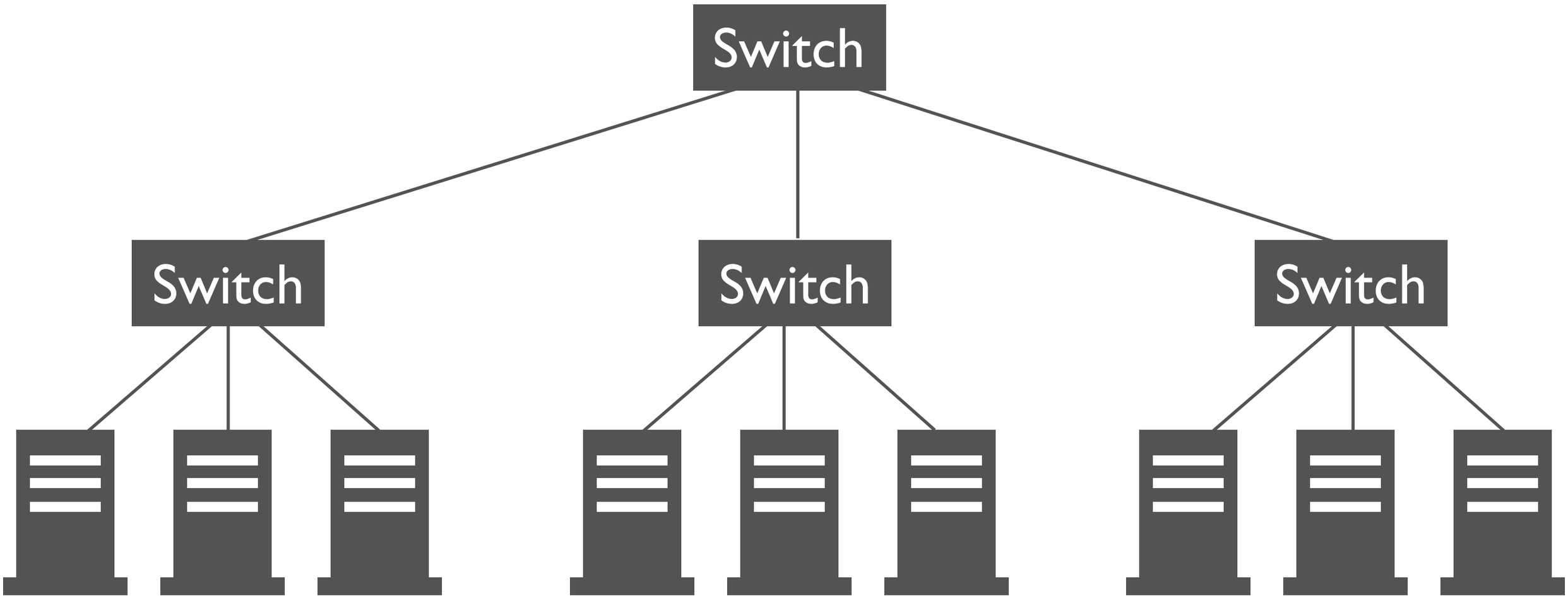
- **Background**
- Programming Model Definition
- Implementation & Refinement

Single Machine

- Fail to deal with rapidly growing data in Storage & Computation

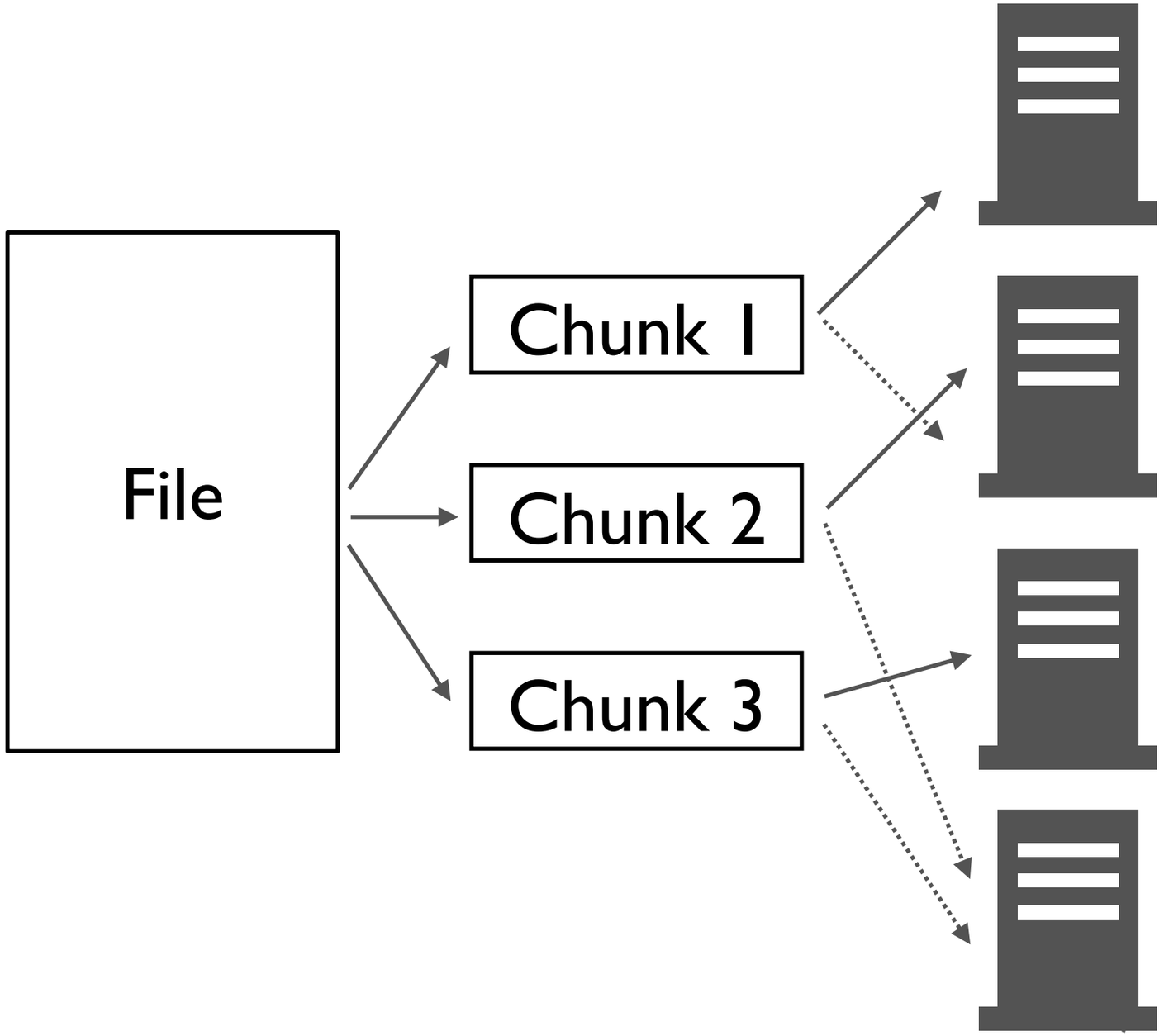


Commodity Cluster



Storage: Google File System (GFS)

- Split file into chunks
- Chunk Server
  - Store chunks
  - Has duplication
- Master server
  - Store meta data



Computation: General Implementation

- Defining computation
- Partitioning input
- Scheduling
- Handling failures
- Managing inter-machine communication
- …

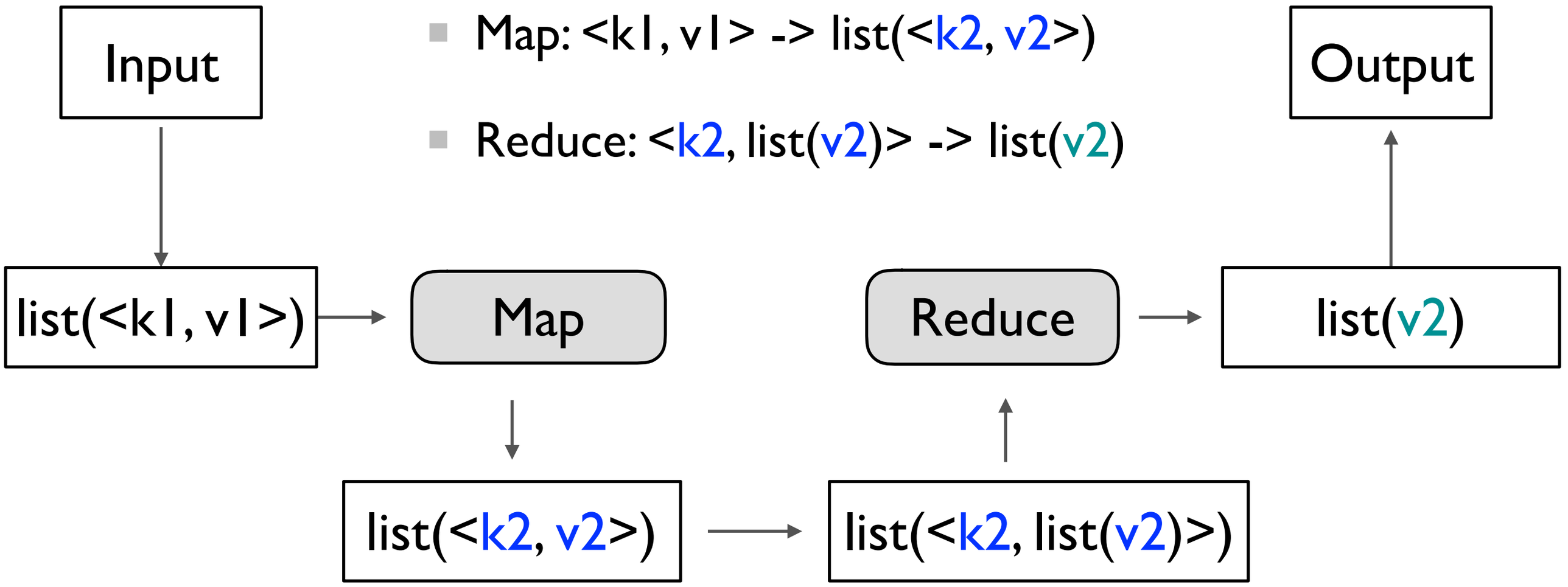
Computation

- Defining computation ← Interface
- Partitioning input
- Scheduling
- Handling failures ← Hide from user
- Managing inter-machine communication
- …

- Background
- **Programming Model Definition**
- Implementation & Refinement

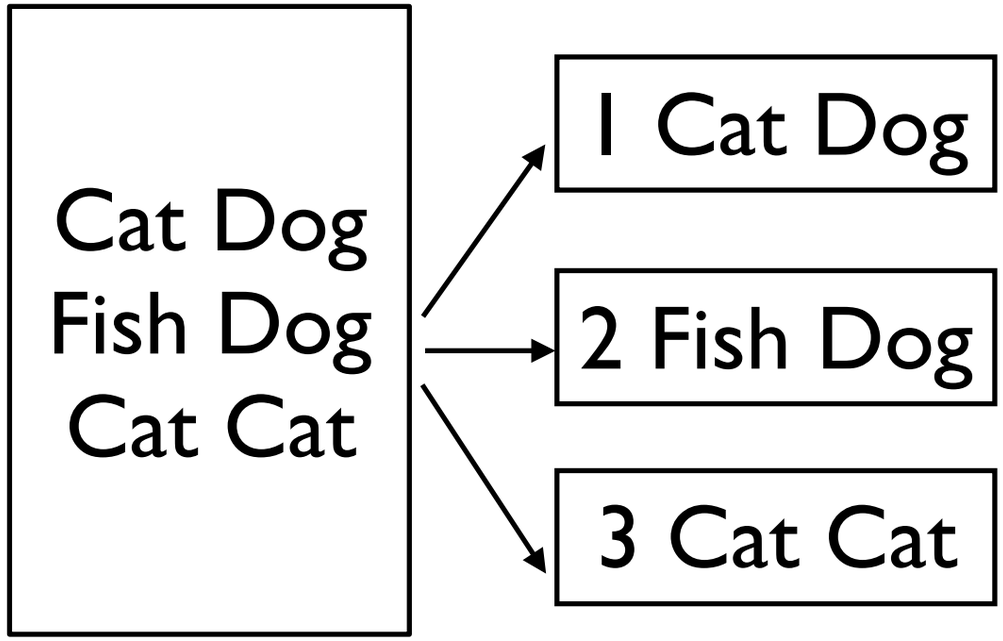
The MapReduce Programming Model

- Map: <k1, v1> -> list(<k2, v2>)
- Reduce: <k2, list(v2)> -> list(v2)



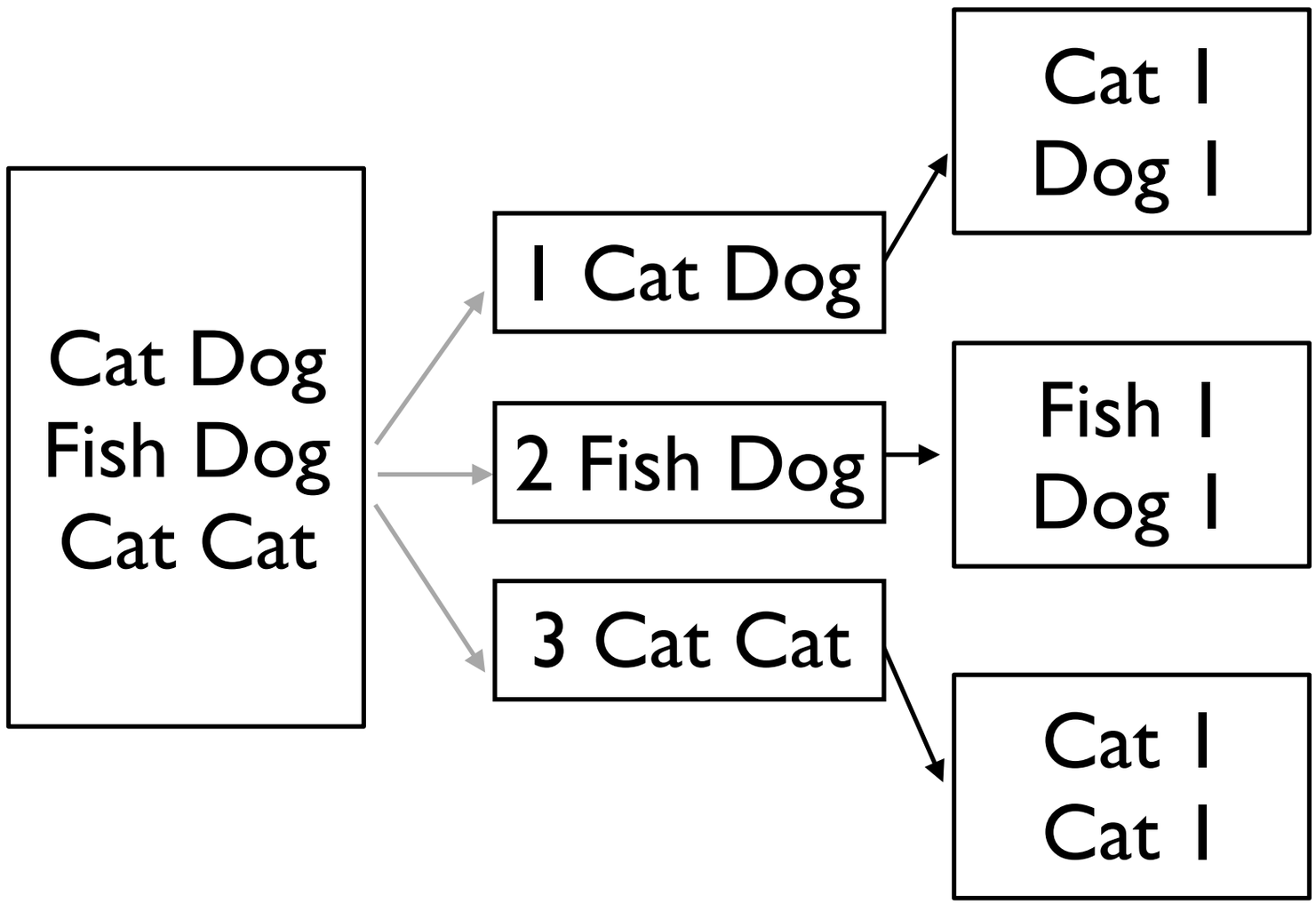
Example: Word Count

- Splitting input: file -> list(<index, line>)



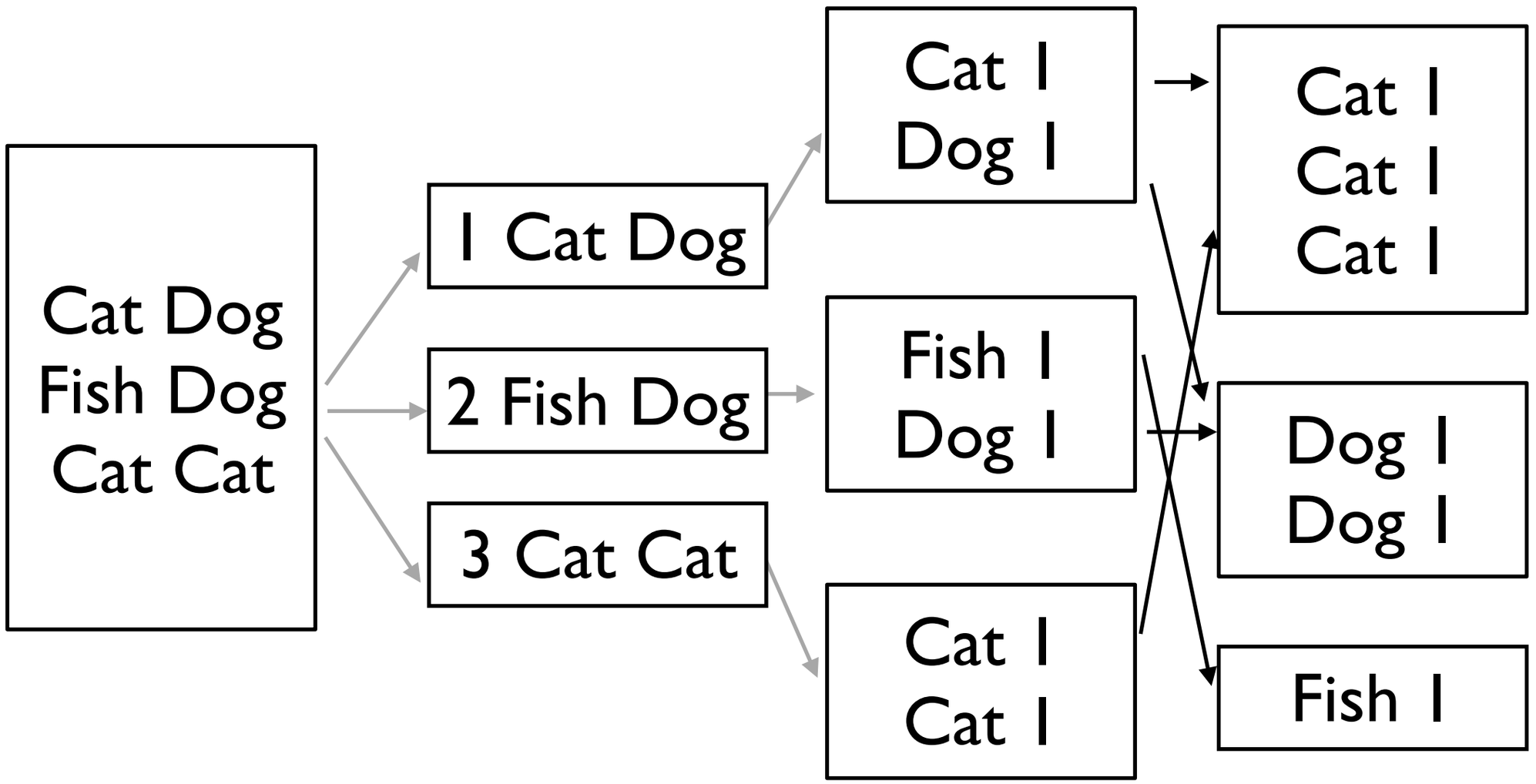
Example: Word Count

- Map: <index, line> -> list(<w, 1>)



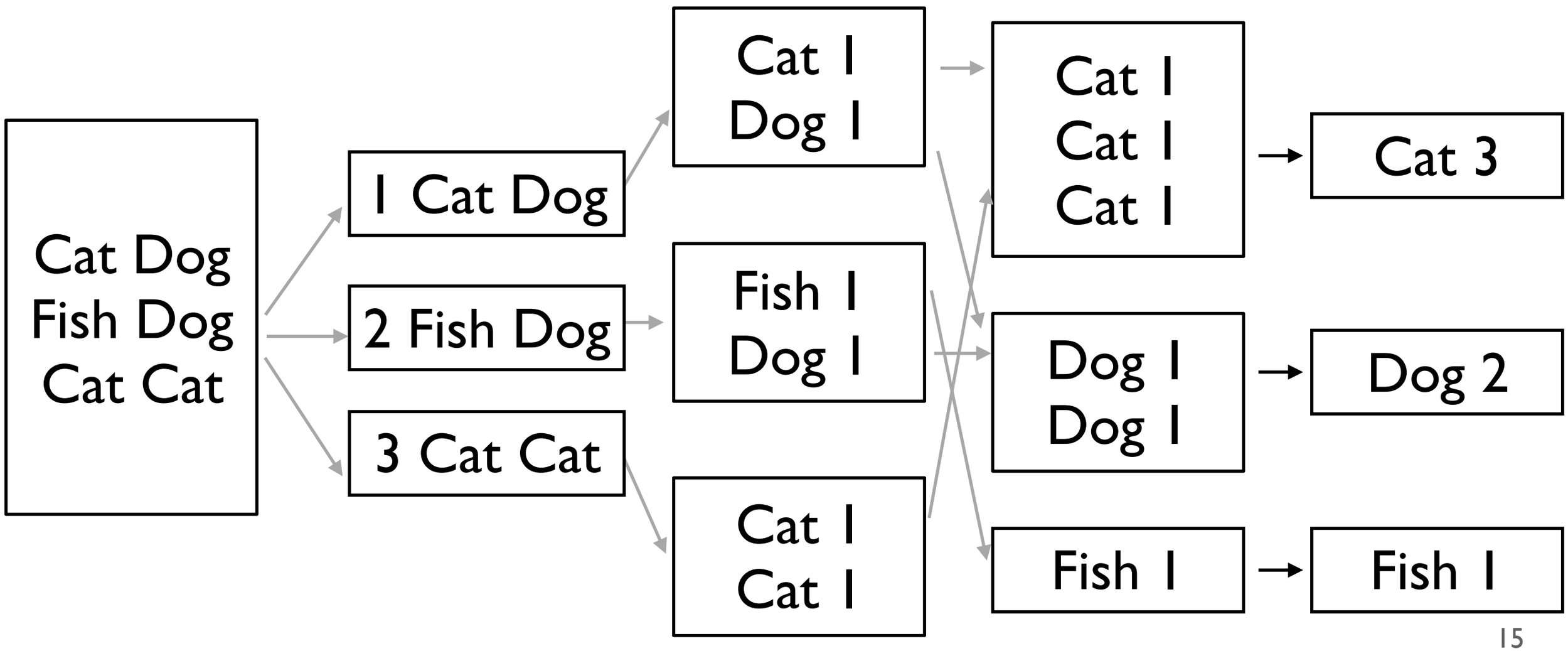
Example: Word Count

- Aggregate: list(<w, 1>) -> list(<w, list(1)>)



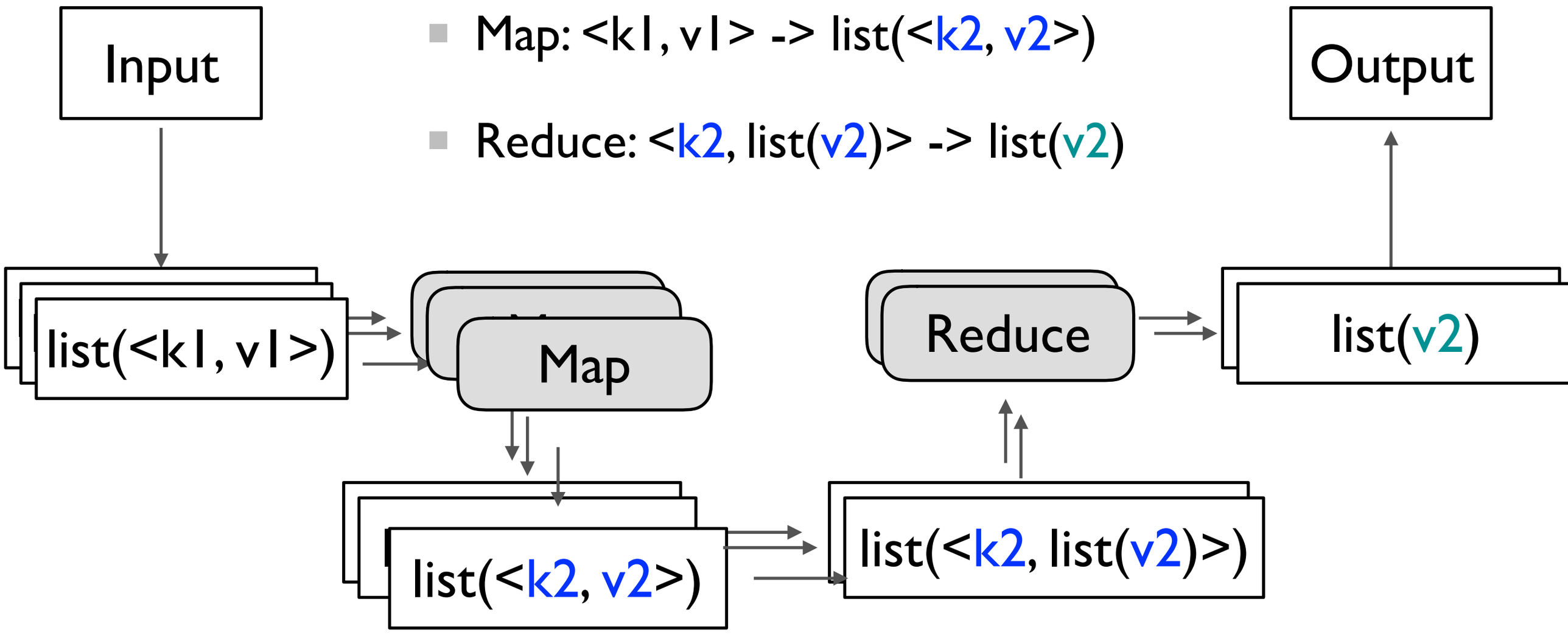
Example: Word Count

- Reduce: <w, list(1)> -> list("w, n")



Parallel Computing

- Map: <k1, v1> -> list(<k2, v2>)
- Reduce: <k2, list(v2)> -> list(v2)

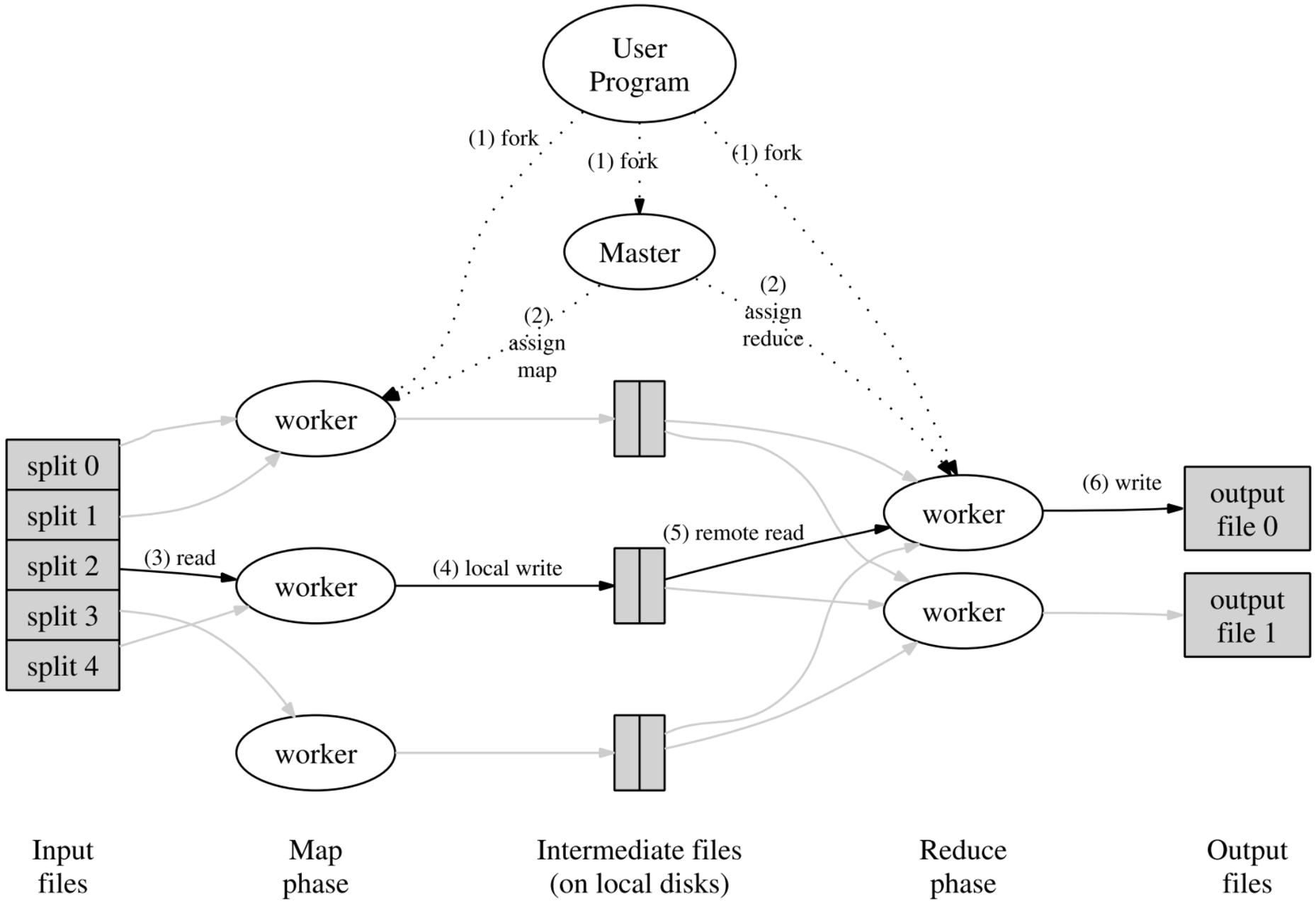


Generalization Capability

- Build Inverted Index for Search Engine
  - Map: <doc ID, content> -> list(<word, doc ID>)
  - Reduce: <word, list(doc ID)> -> list("word+list(doc ID)")
- Count URL Access Frequency
- Grep from distributed file system
- Sort data on distributed file system
- …

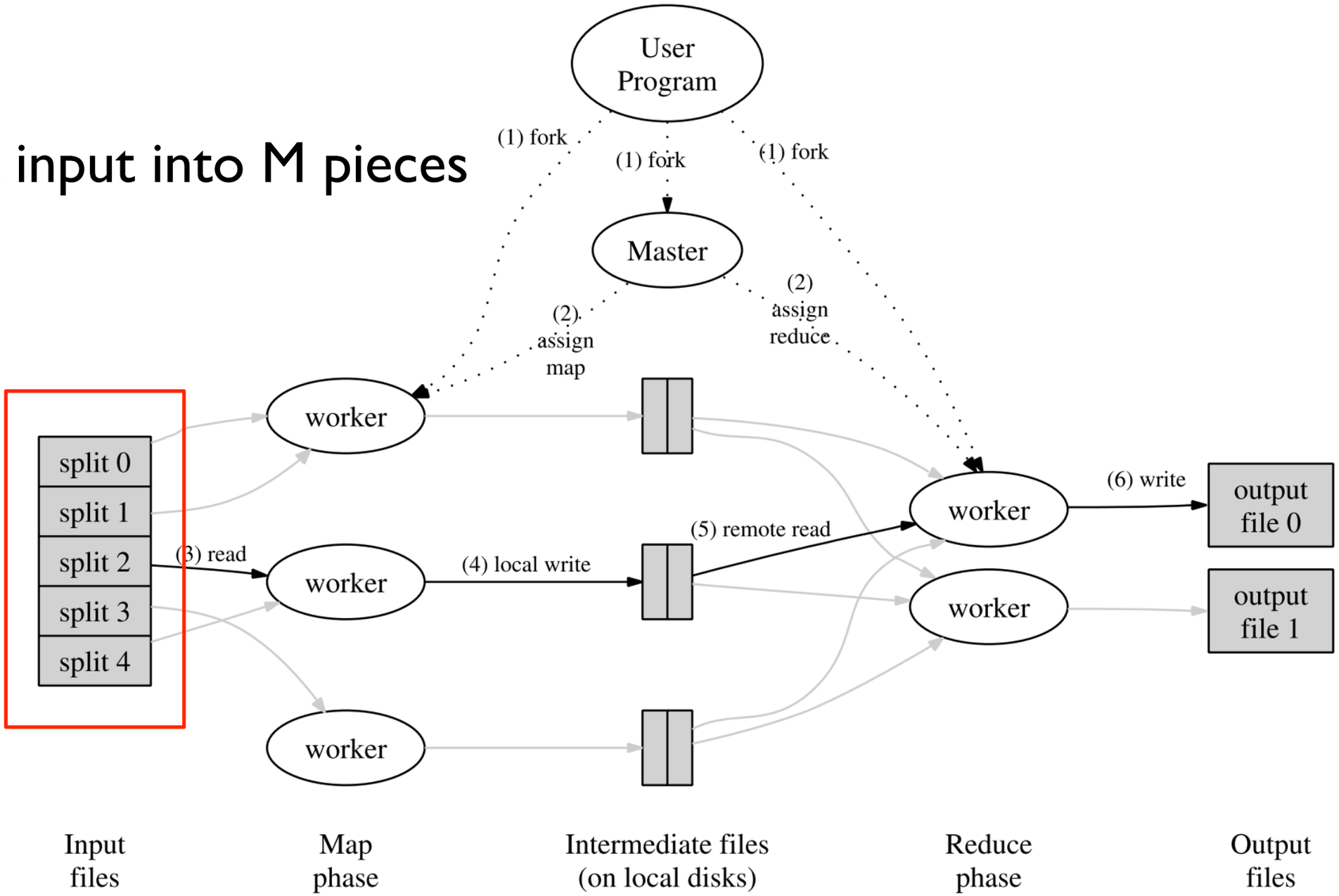
- Background
- Programming Model Definition
- **Implementation & Refinement**

Execution: Overview



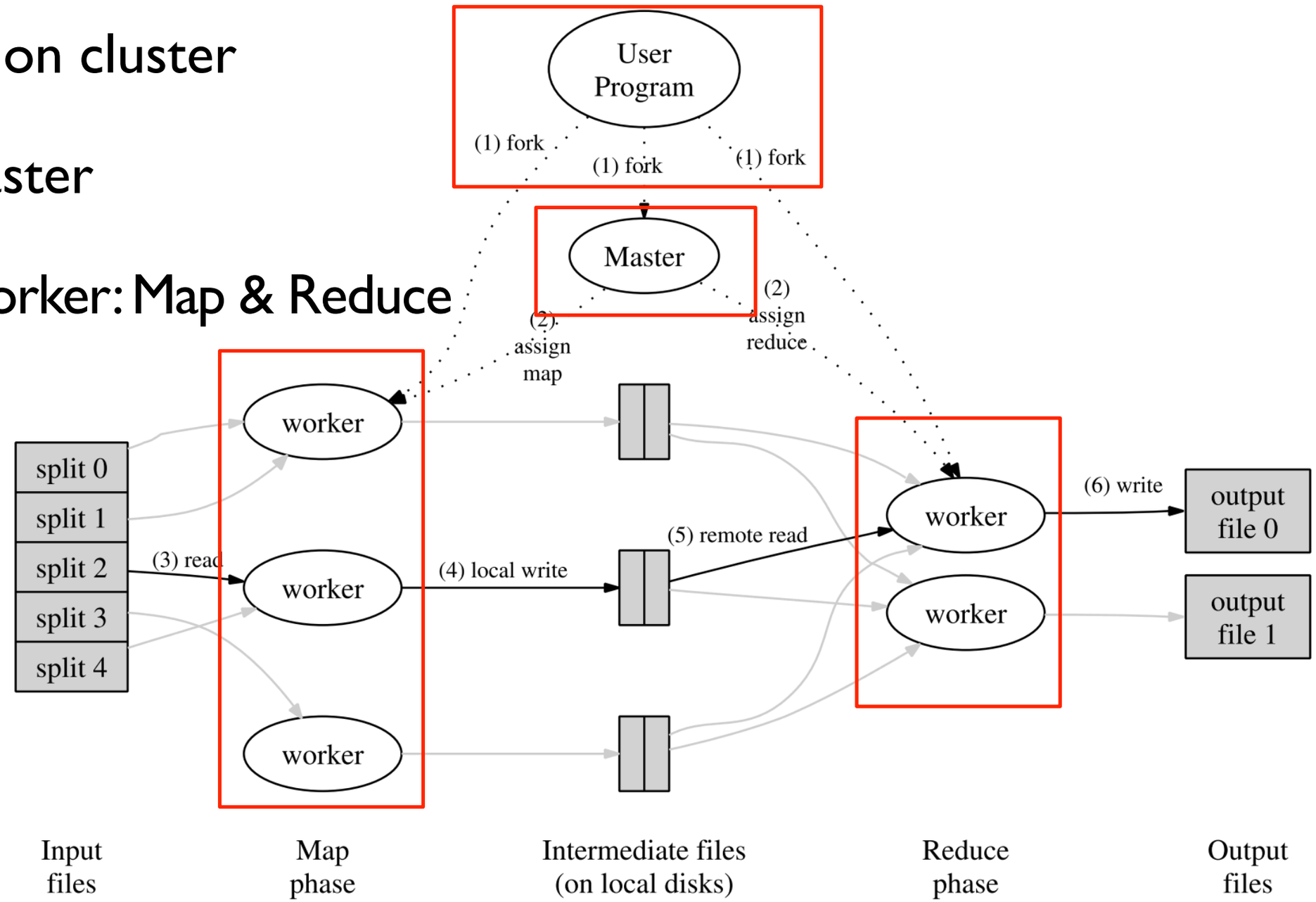
Execution: Overview

- Split input into M pieces



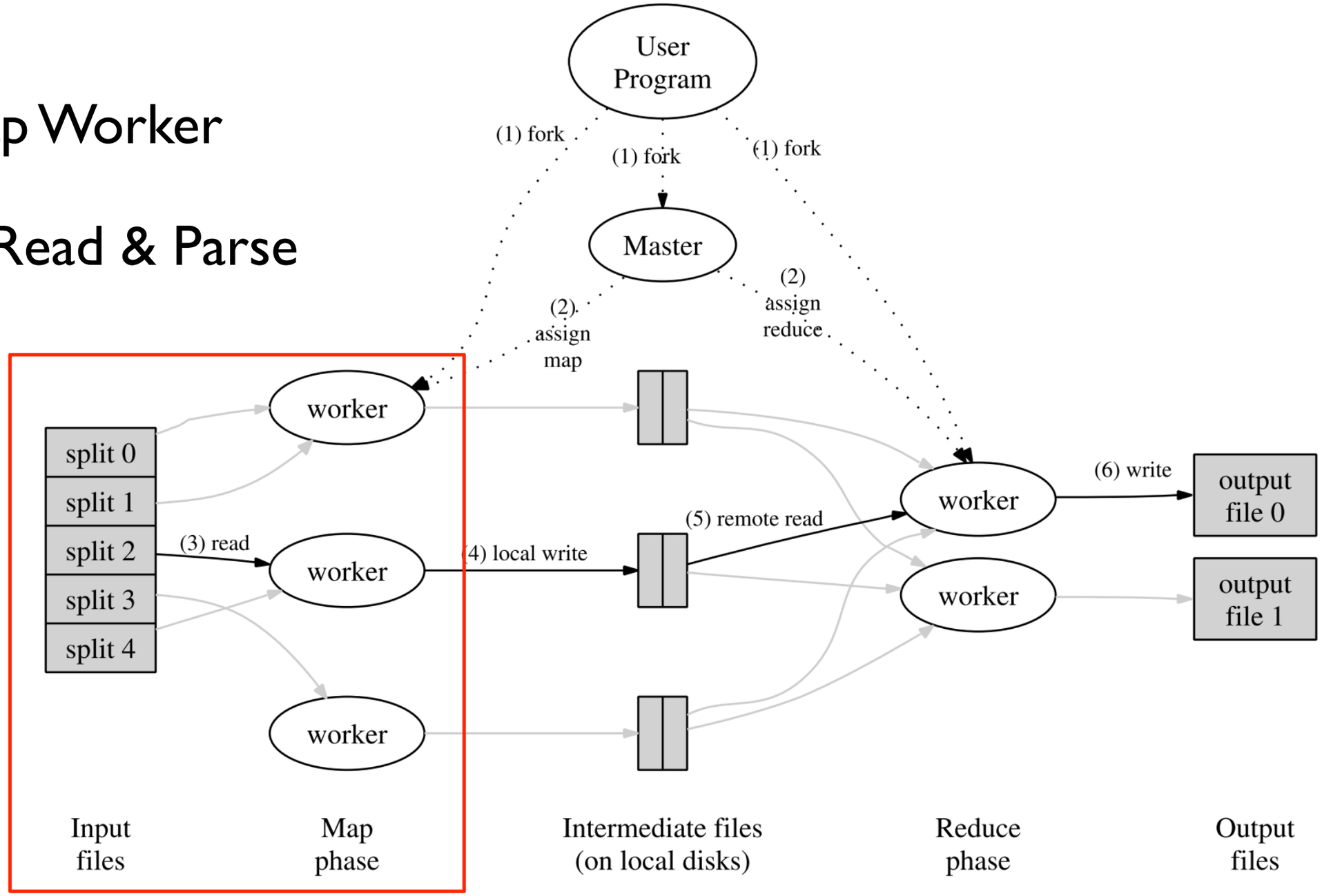
Execution: Overview

- Start on cluster
  - Master
  - Worker: Map & Reduce



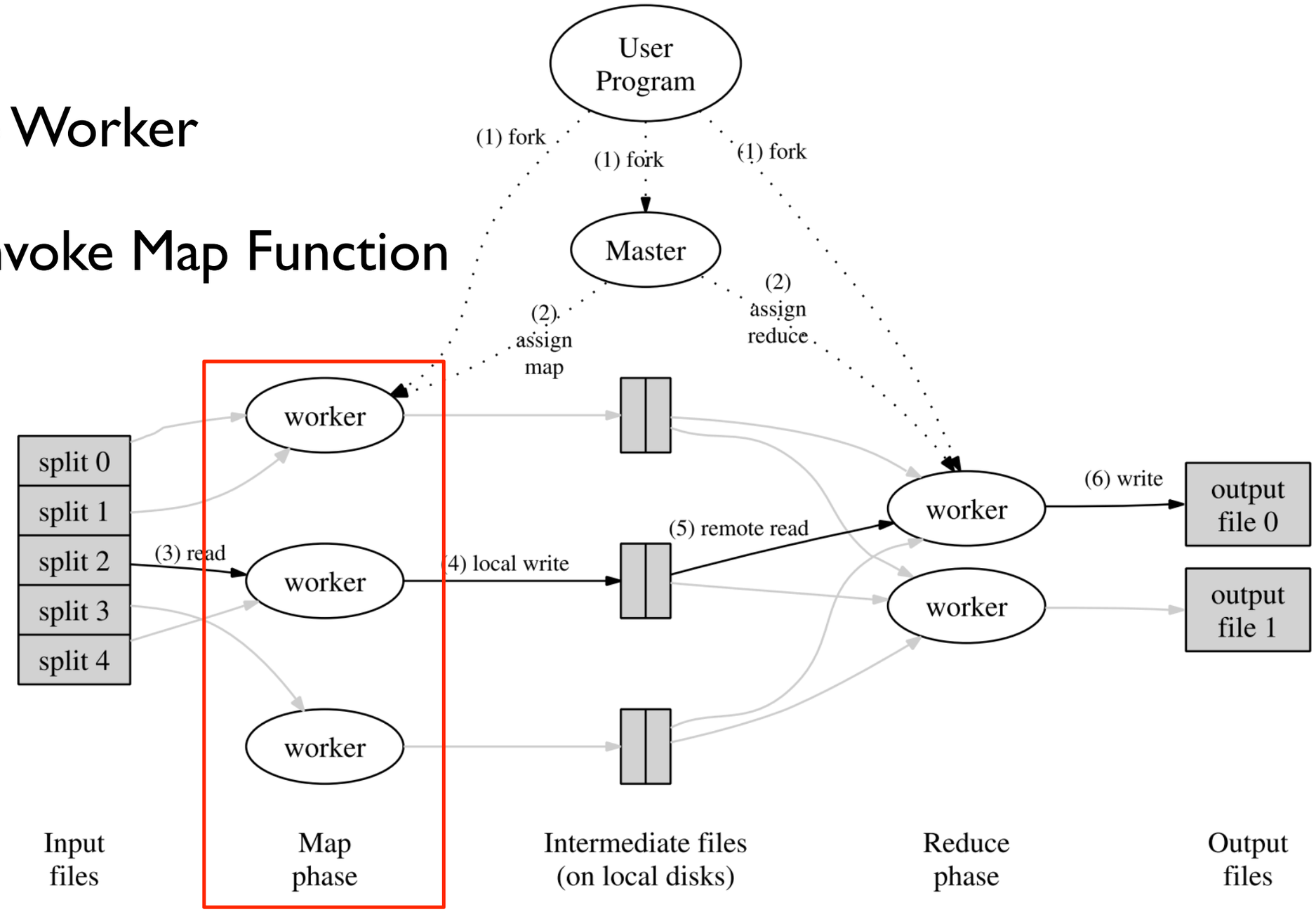
Execution: Overview

- Map Worker
  - Read & Parse



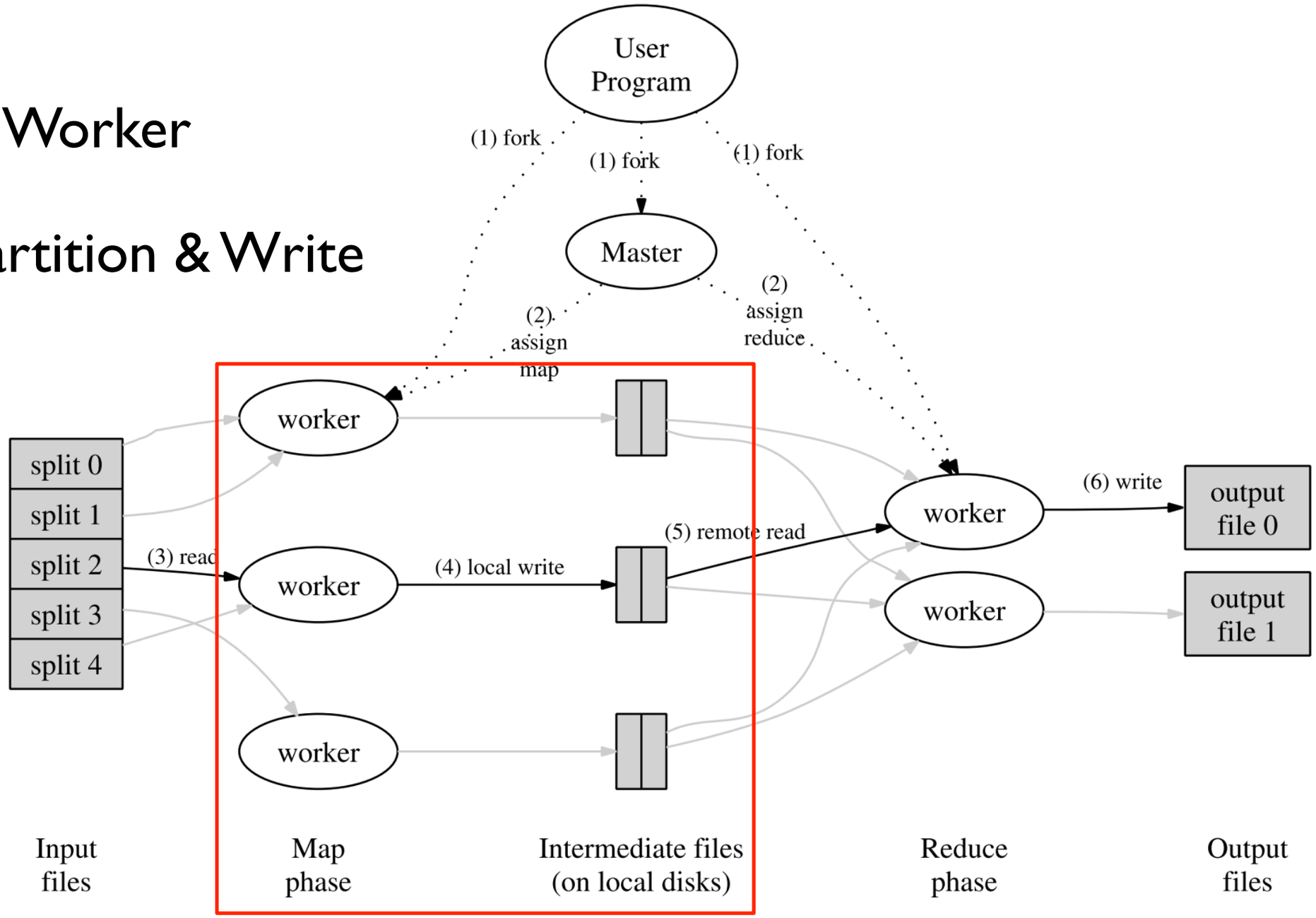
Execution: Overview

- Map Worker
  - Invoke Map Function



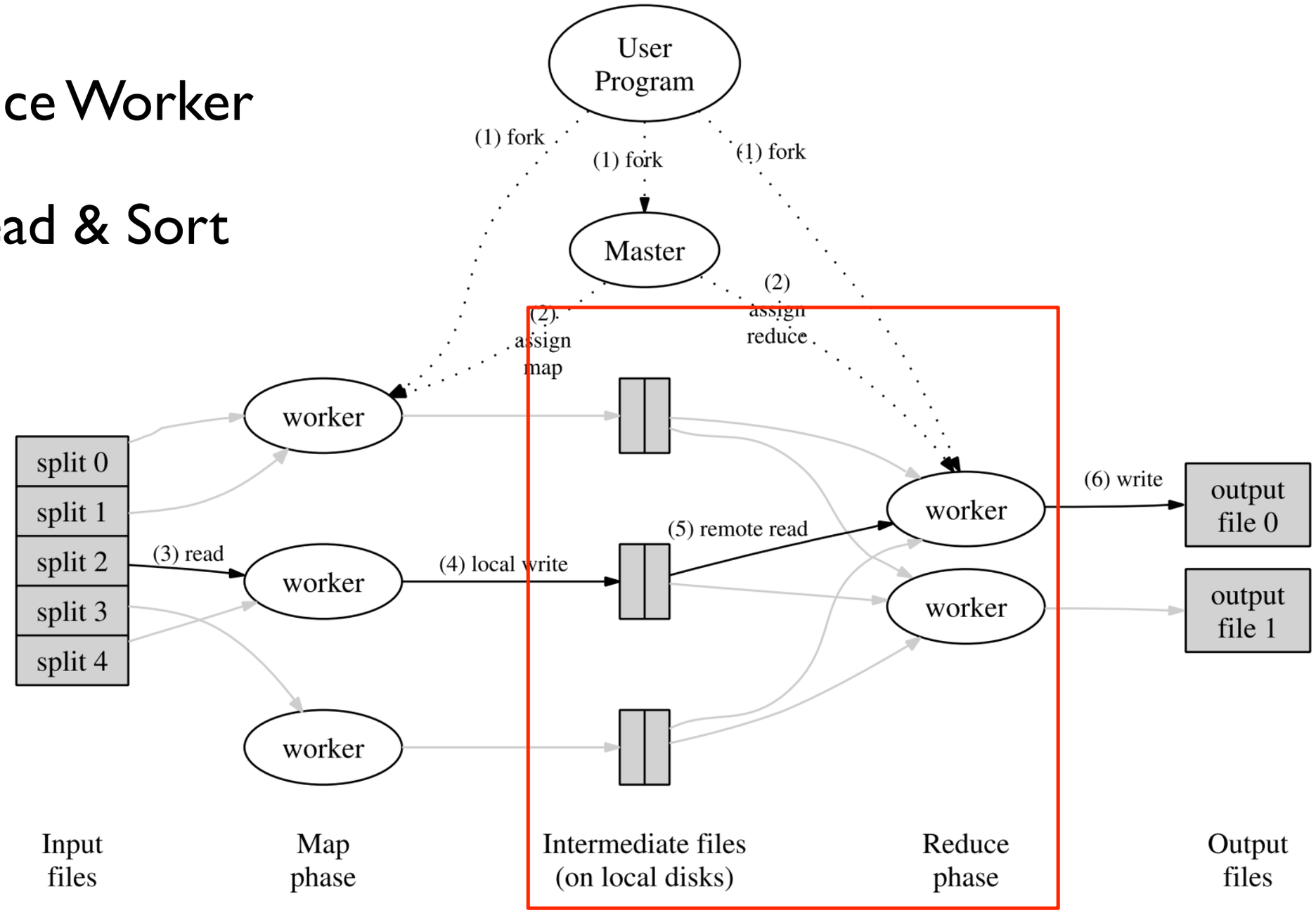
Execution: Overview

- Map Worker
  - Partition & Write



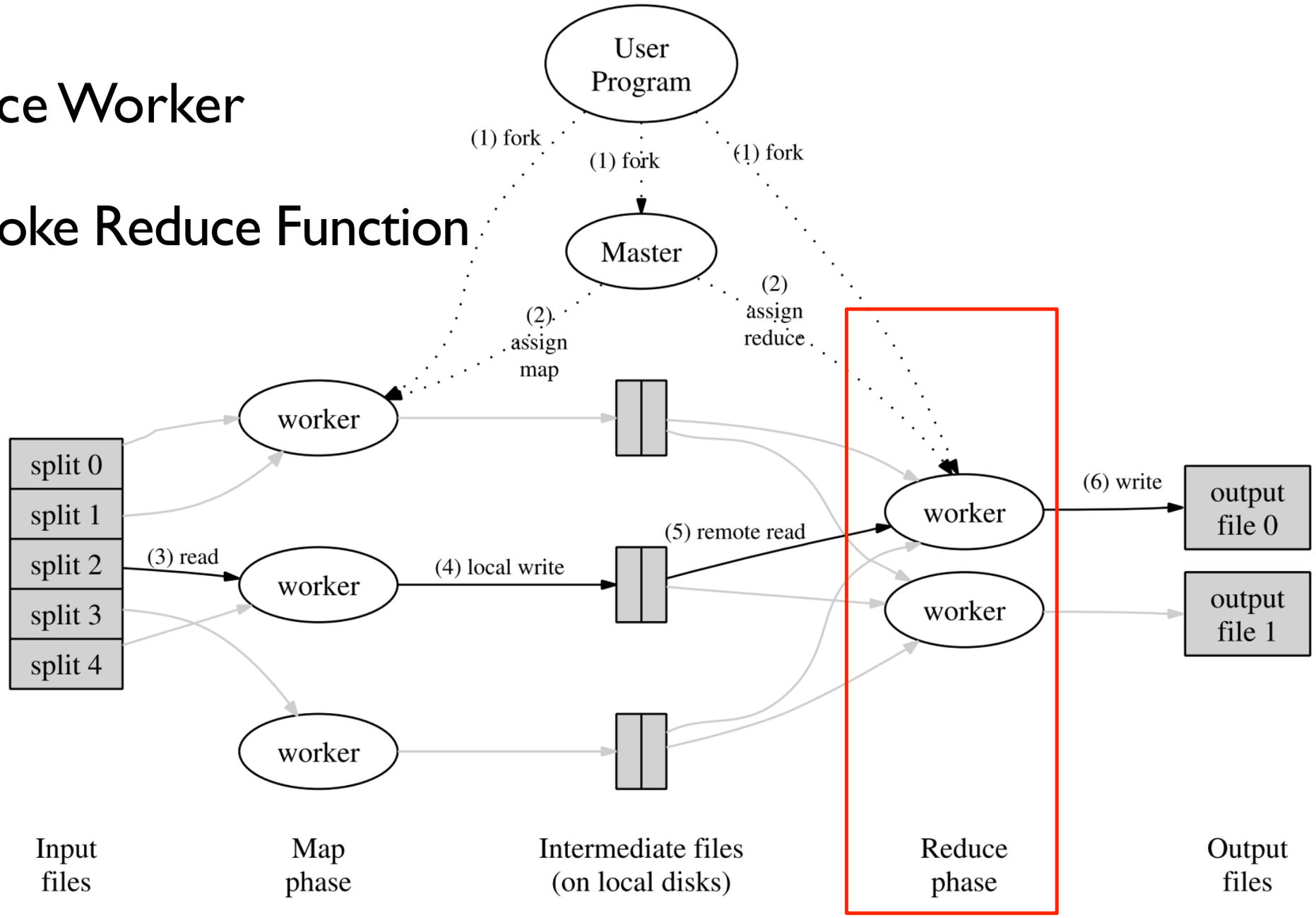
Execution: Overview

- Reduce Worker
  - Read & Sort



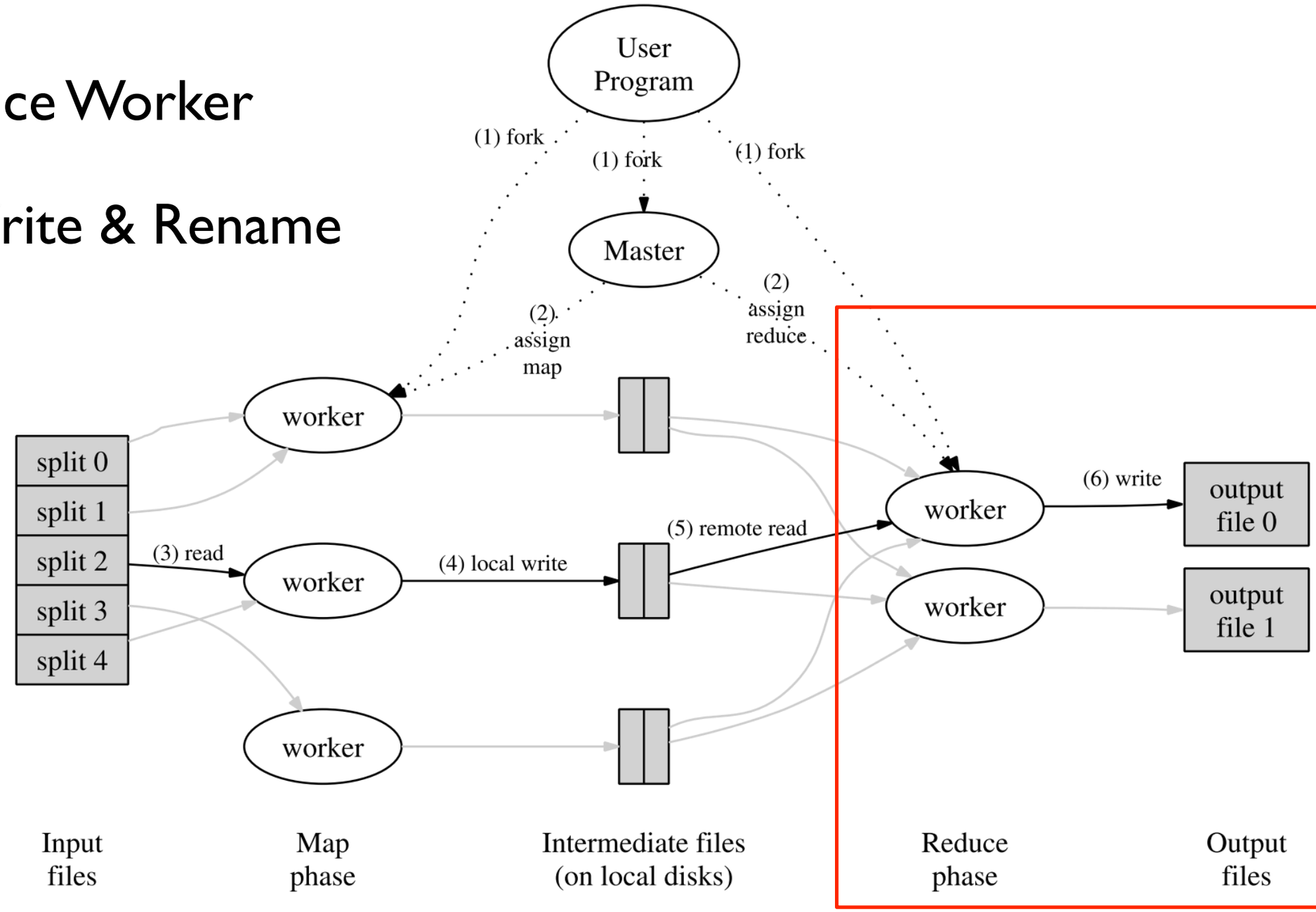
Execution: Overview

- Reduce Worker
  - Invoke Reduce Function



Execution: Overview

- Reduce Worker
  - Write & Rename



Master

- Schedule
  - Task status: idle, in-progress, completed & worker machine (for non-idle tasks)
  - Worker status: ping periodically
- Communicate
  - Location and size of intermediate results

Fault Tolerance

- Worker
  - Completed and in-progress map tasks at this worker are reset to idle
  - Only the in-progress reduce task is reset to idle
- Master
  - Start a new copy by checkpoints / Abort

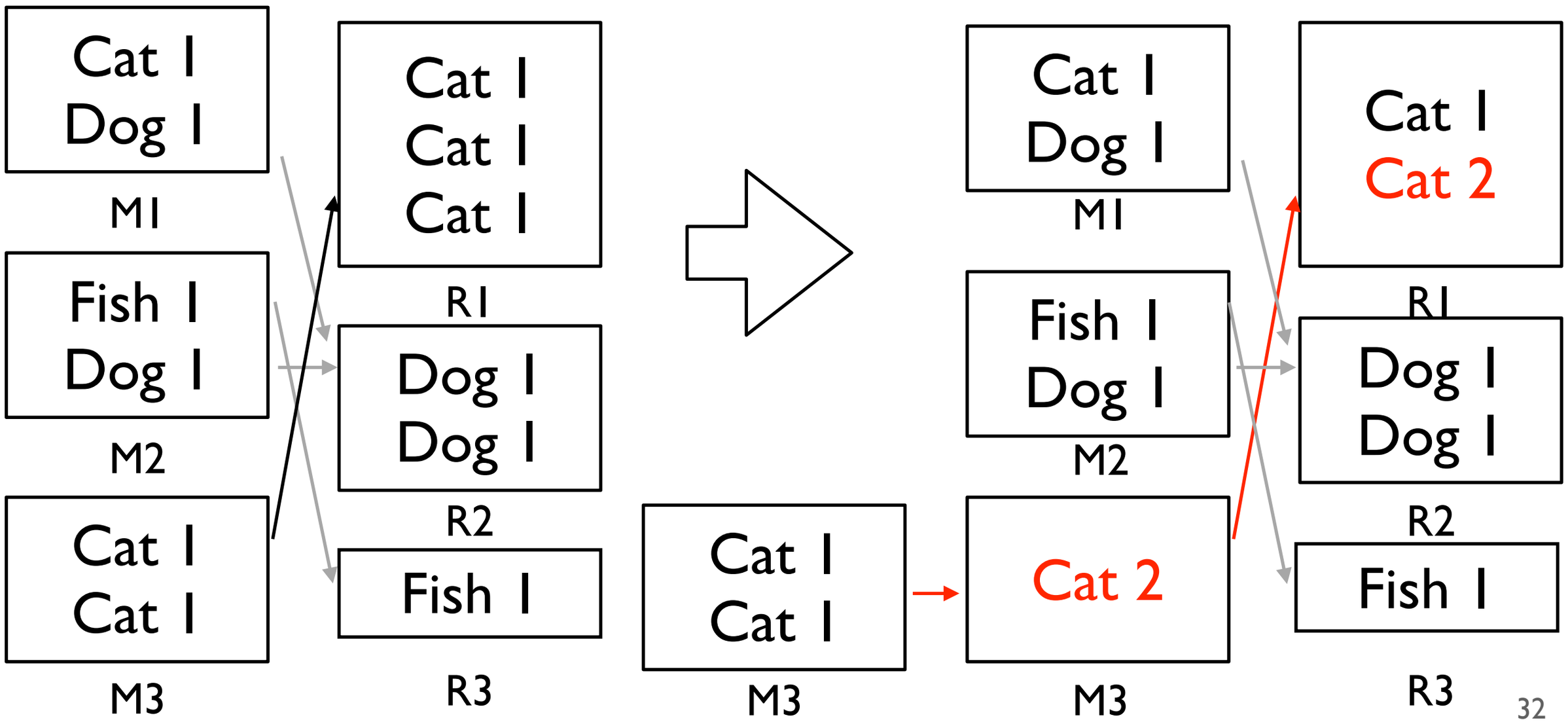
Task Granularity

- Split input into M pieces -> M map tasks
- Divide intermediate result into R pieces -> R reduce tasks
- M & R should be much larger than the number of machines
  - Dynamic load balancing
  - Speed up recovery
- 2000 machines: M=200,000, R=5000

Speed Up: Network Bandwidth

- Locality
  - Master assign a map task to a worker machine that
    - Contains a replica of the input data
    - Near a replica of the input data
- Combiner Function
  - Merge intermediate result before sent through network

Combiner Function in Word Count



Speed Up: Backup Tasks

- Issue: straggler
  - A machine that takes unusually long time to complete one task
- Solution: backup tasks
  - Schedule backup executions for in-progress tasks when the MapReduce operation is close to completion

Other Details

- Customization
  - Partitioning Function, Input & Output Types
- Debug & Monitor
  - Local Execution, Status Information, Counters…
- Skipping Bad Records
- …

TakeAways

- MapReduce is a **general** programming model that it can be used in various application scenarios
- MapReduce hides the detail implementation from users and provide **simple & flexible** interfaces
- The implementation of MapReduce is **efficient** and highly **scalable**

Q & A