An Efficient Solution for Processing Skewed MapReduce Jobs

Reza Akbarinia et al.
Hoscar 2015

Work presented in :
**** Int. Conf. on Database and Expert Systems Applications (DEXA), 2015***
**** Int. Conf. On Very Large DataBases (VLDB), 2015***

Overview

- The Context
- The Problem
- FP-Hadoop
  - Foundation
  - How does FP-Hadoop work?
  - Experimental evaluation
- Conclusions

MapReduce : A Framework for Bigdata Processing

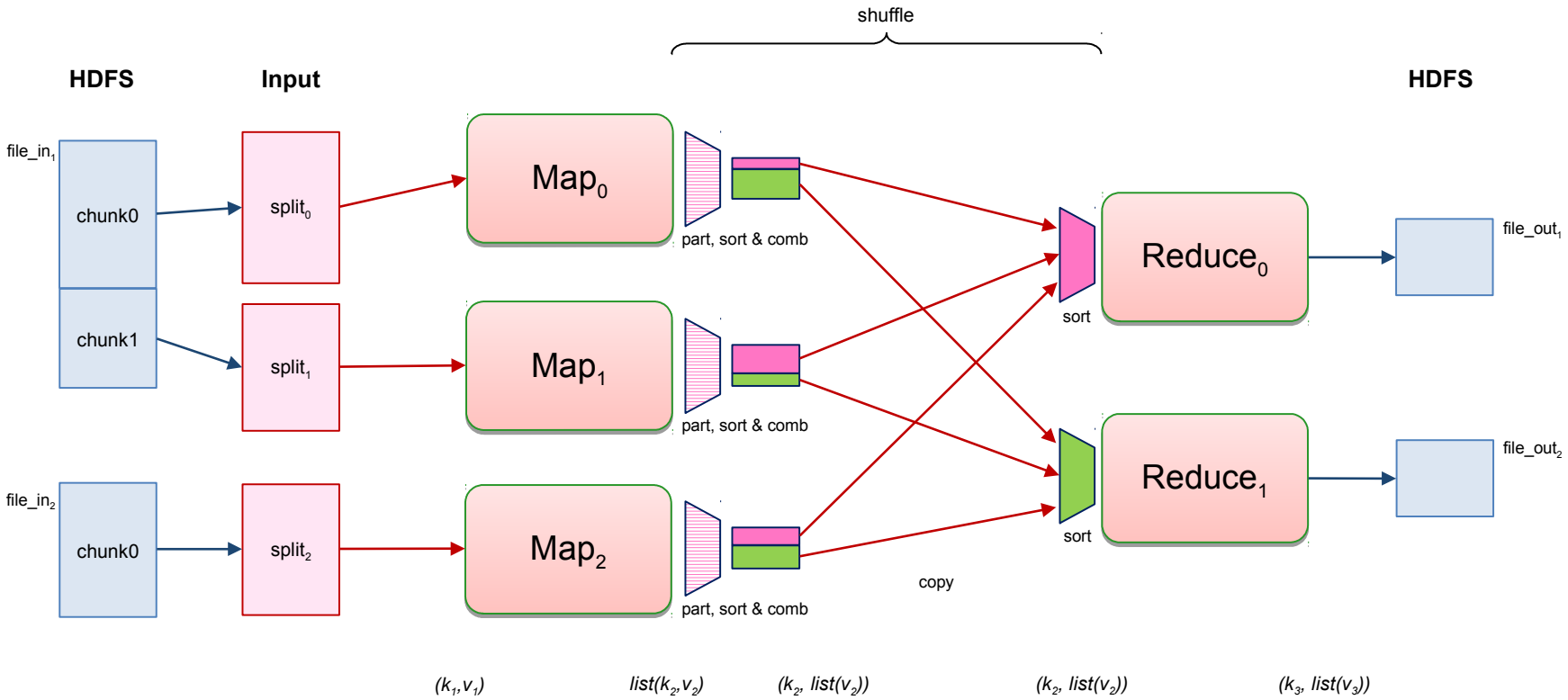
- *Programming model and framework*
  - Big data parallel processing in data centers
    - e.g., PageRank, inverted index
  - Automatic parallelization, distribution and fault tolerance

- Proposed by Google [OSDI 04]
  - Executed on top of Google File System (GFS)
- Inspired Apache project ▭ Hadoop

MapReduce Programming Model

- Data consists of
  - (key, value) pairs, aka tuples
- Functions
  - map: $(k_1,v_1) \rightarrow list(k_2,v_2)$
  - reduce: $(k_2, list(v_2)) \rightarrow list(k_3,v_3)$
  - Optionally
    - combine: $(k_2, list(v_2)) \rightarrow (k_2, list(v_2))$

MapReduce Job Execution

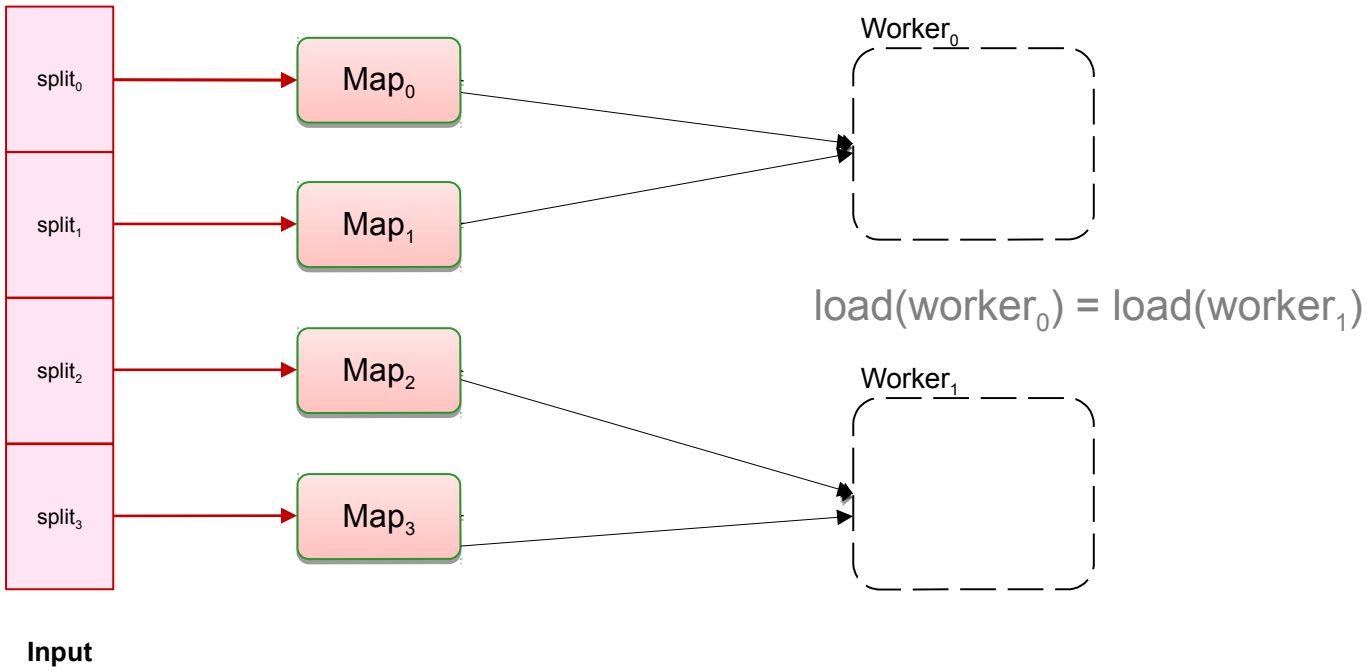


How Load Is Distributed to Workers?

- Map phase
  - Input data is divided into splits of similar size
  - Splits are assigned to map tasks
    - Assigned independently to free workers
- Reduce phase
  - Intermediate keys assigned to partitions
  - Partitions assigned tor reduce tasks
    - Assigned independently to free workers

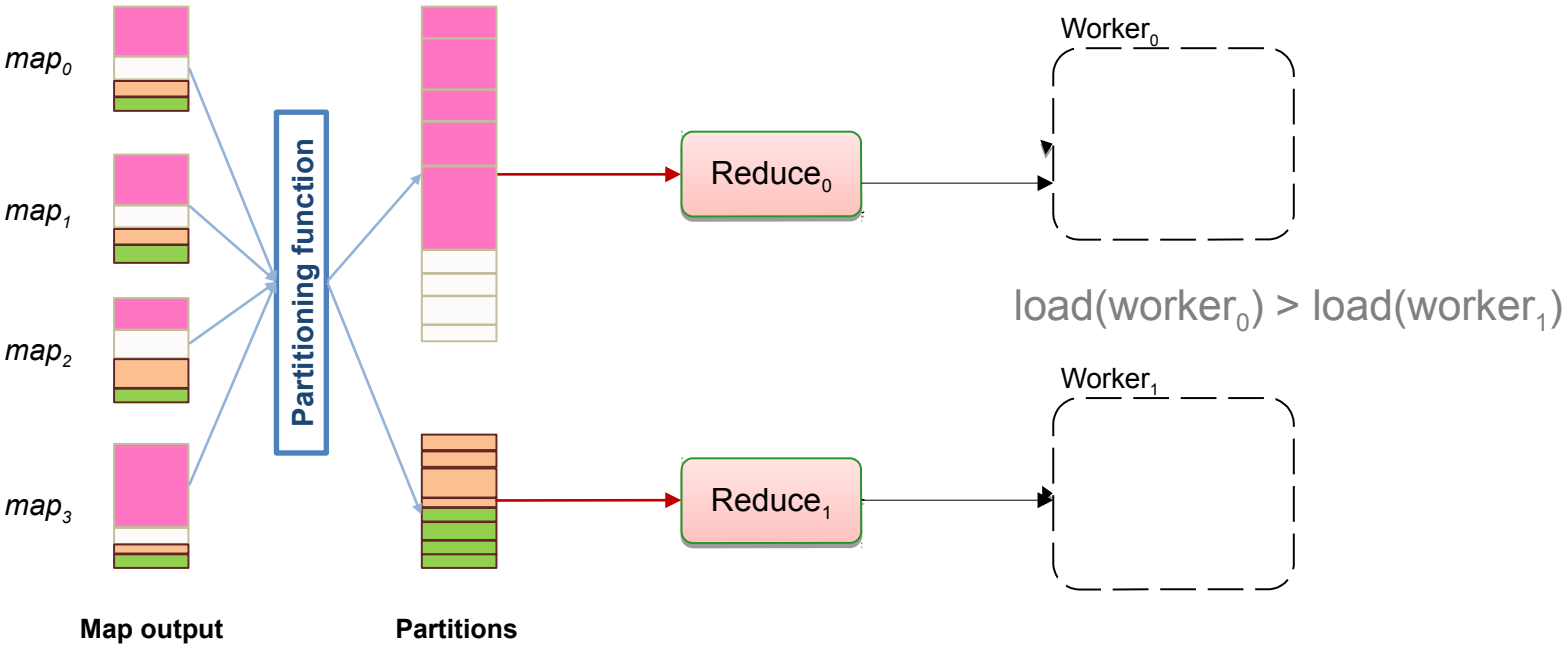
How Load Is Distributed to Workers?

- Map phase



How Load Is Distributed to Workers?

- Reduce phase



Problem: Data Skew

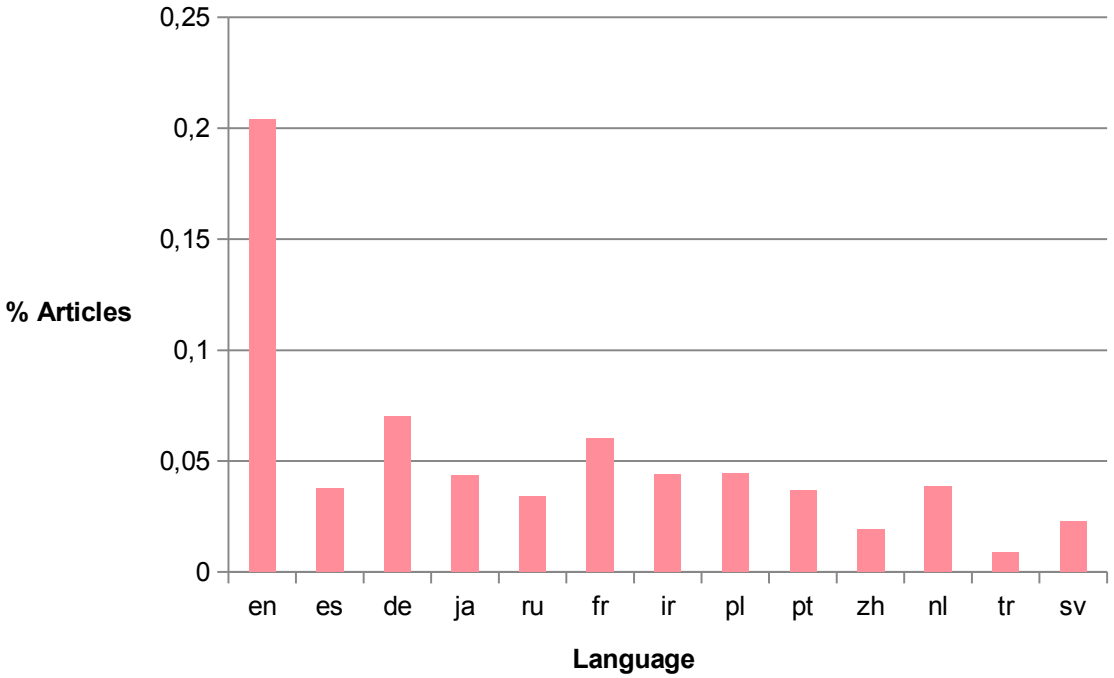
- Load balancing problem
  - Some reduce tasks receive more data to process
  - Thus, *parallelism is not complete*

- Causes
  - Bad partitioning function

    May be solved but needs *a priori* knowledge*
  - Popular keys
    - Hard to solve : data of each key should be sent to the same worker

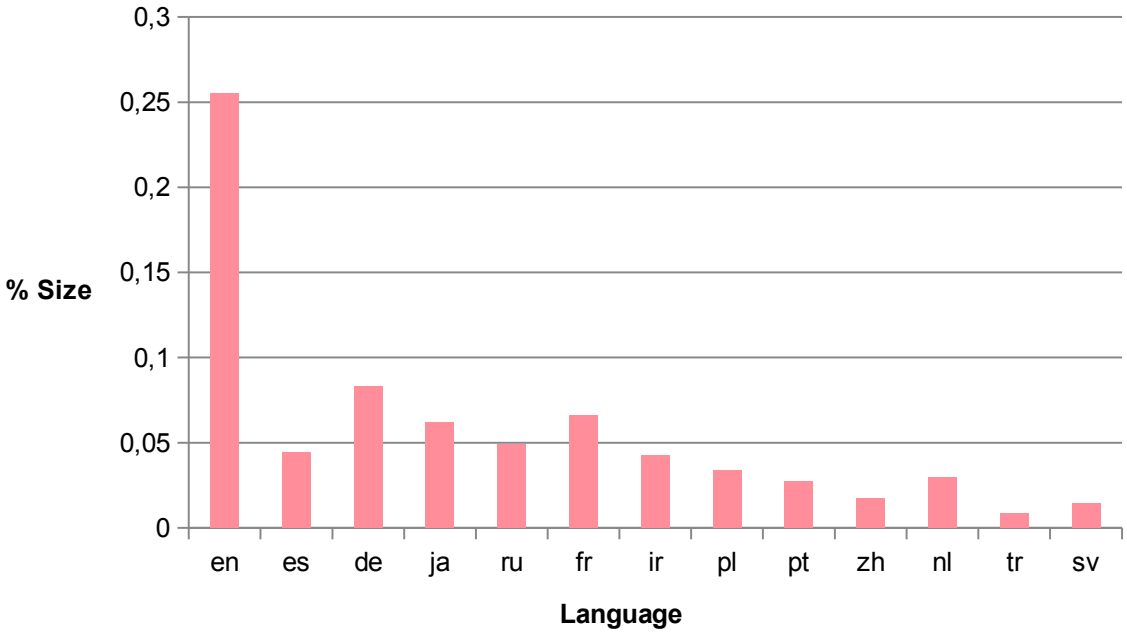
Motivating Example

- Wikipedia **number** of articles per language



Motivating Example

- Wikipedia **size** per language



FP-HADOOP

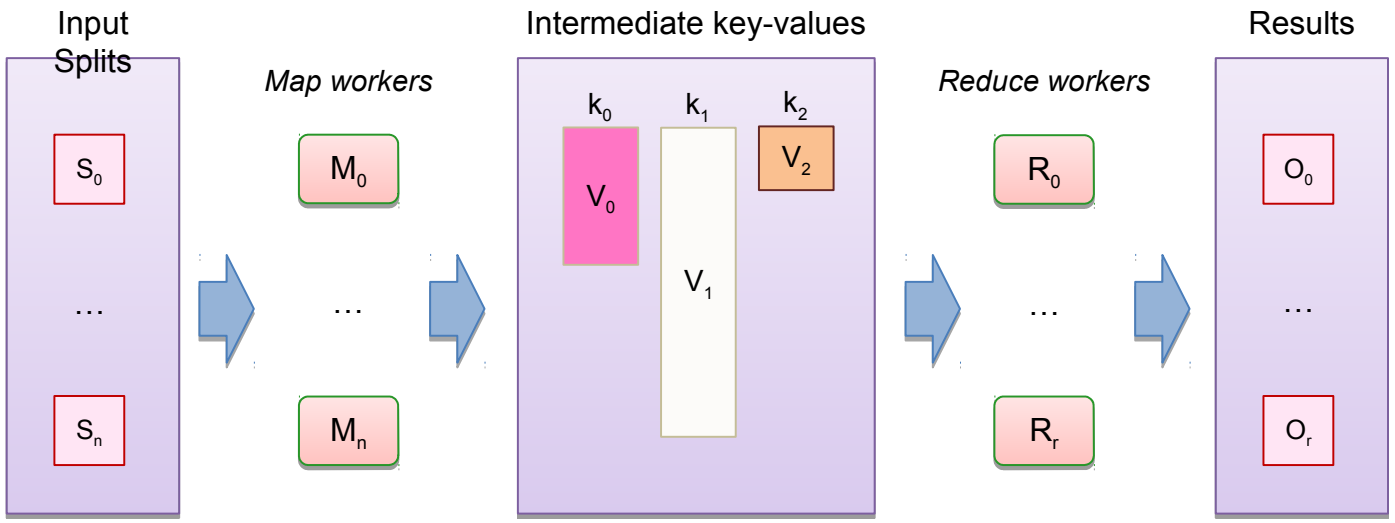
FP-Hadoop: Foundation

- Main idea
  - Workers collaborate in the processing of intermediate key
  - Obtain maximum parallelism

- FP-Hadoop: 3 phases
  - Map : unchanged
  - **Intermediate reduce (IR)**
    - IR tasks process *bounded splits* (IR splits) of intermediate data
    - *Iterative*: Popular keys generate several waves of IR tasks
  - Final Reduce (FR) : same as original reduce

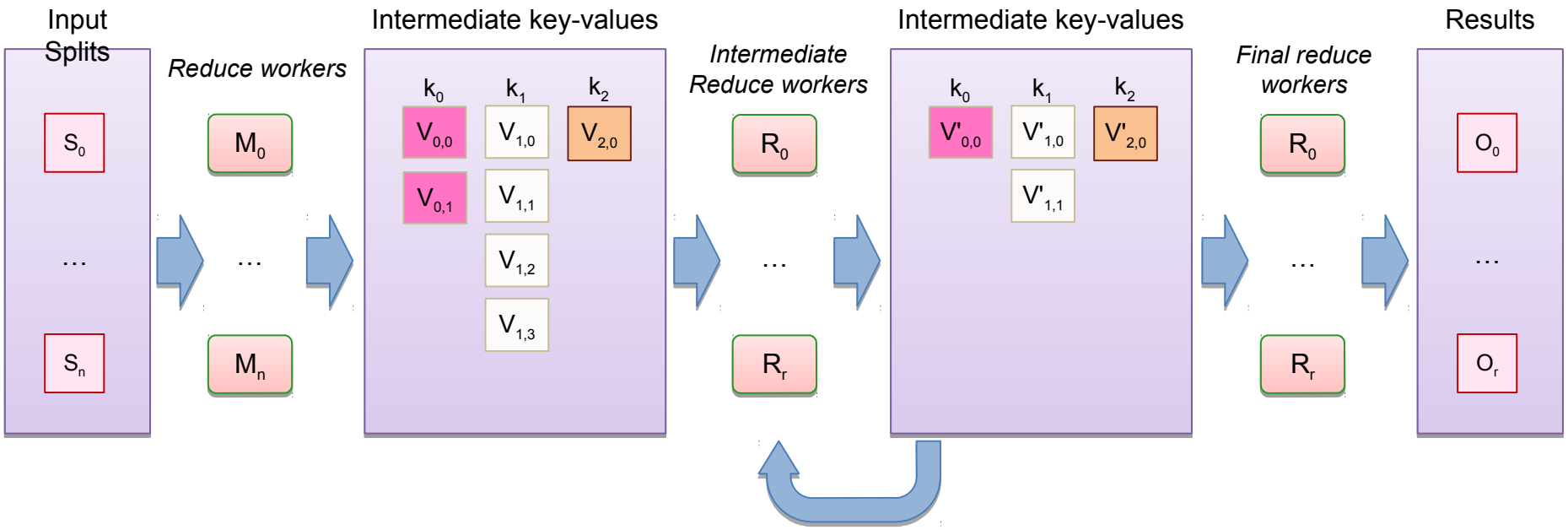
FP-Hadoop: Foundation

- Job processing with **Hadoop**



FP-Hadoop: Foundation

- Job execution with **FP-Hadoop**

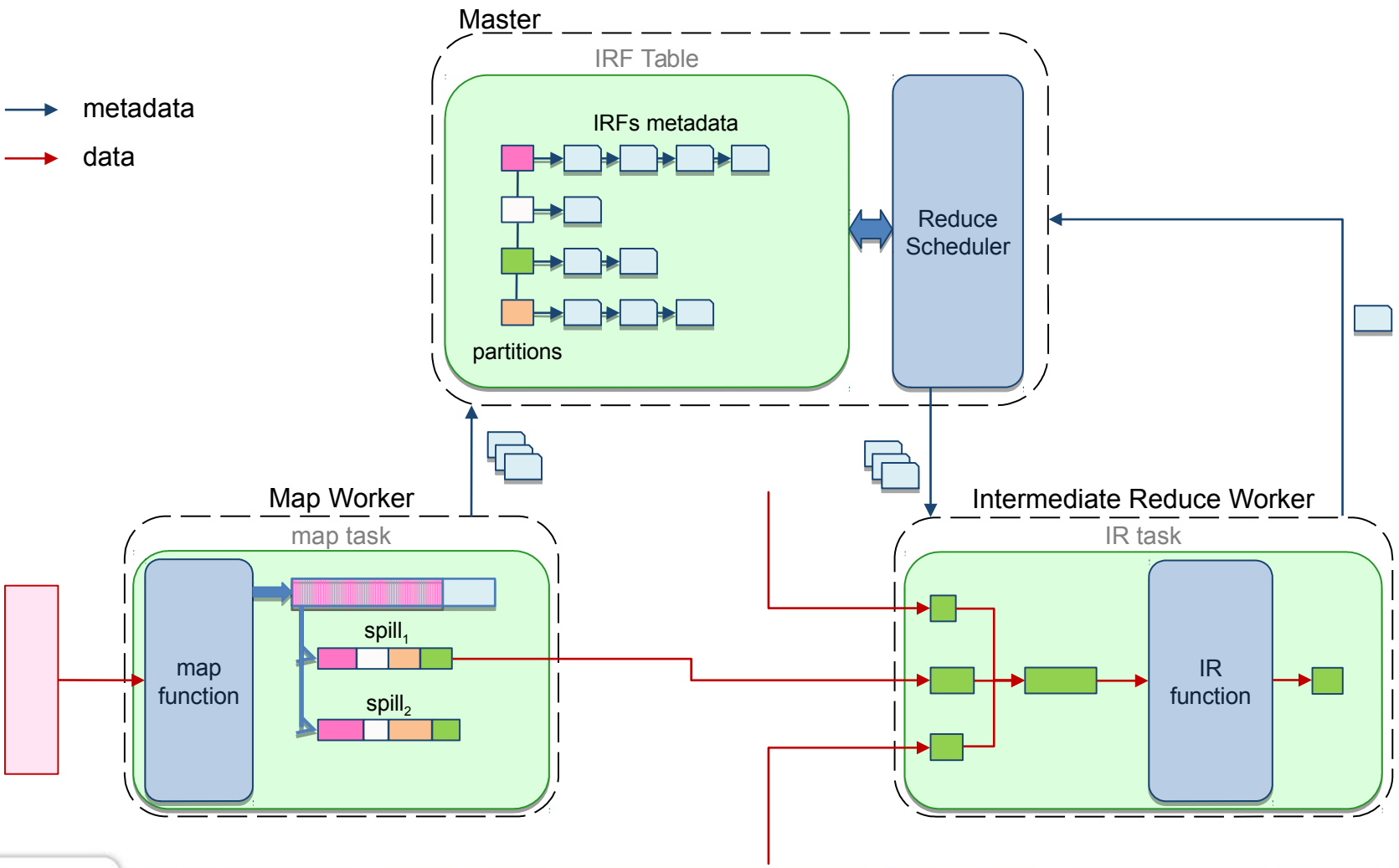


HOW DOES FP-HADOOP WORK?

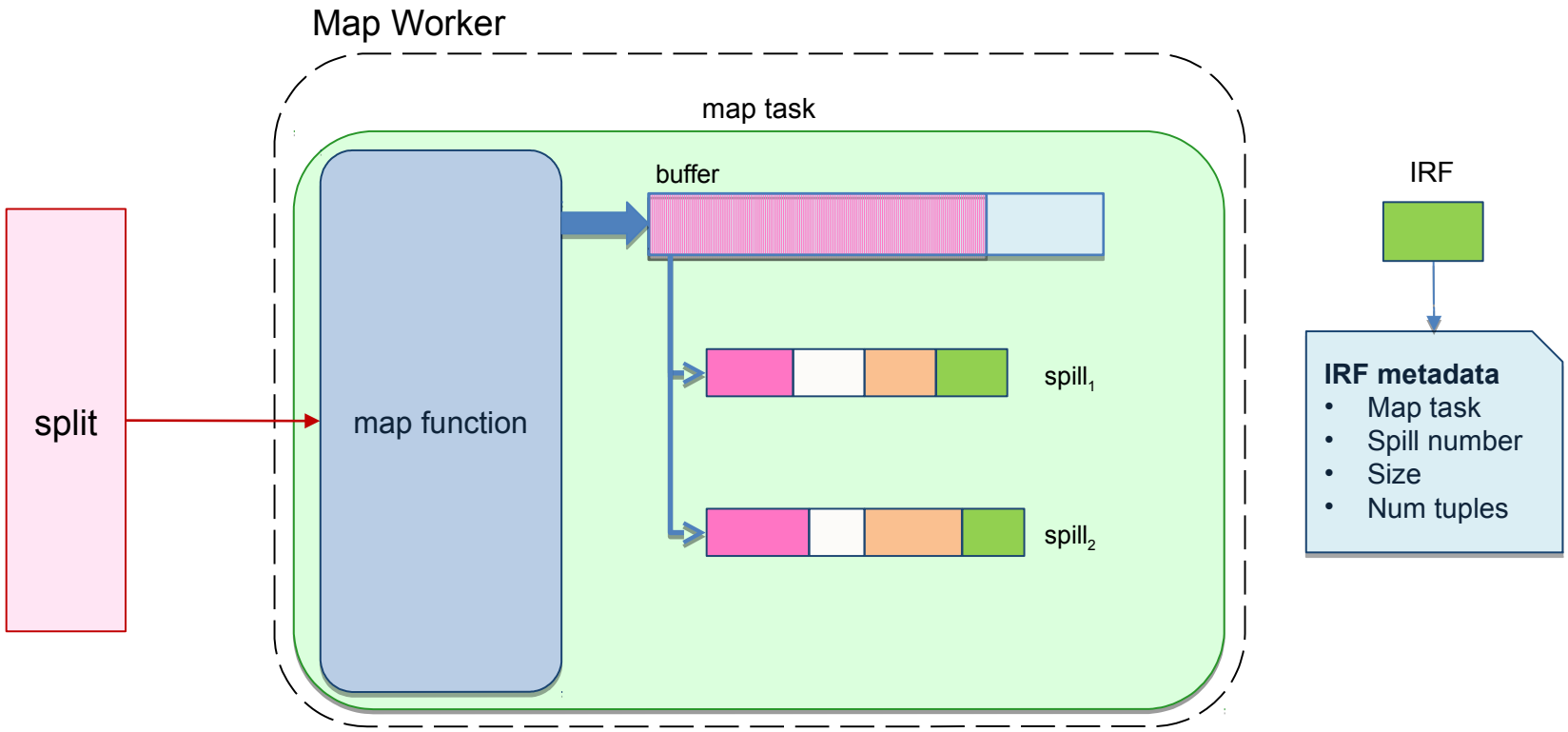
Dynamic Generation of IR Splits

- **IR Split**
  - Unit of processing in the intermediate phase
    - 1 IR split ▭ 1 IR task
  - Bounded: min and max size
  - Composed of *IR fragments*
    - 1 IRF = 1 partition in map output spills
- Reduce scheduler
  - Generates IR splits from IR fragments
  - Dynamically generate and assign IR split when free worker

Dynamic Generation of IR Splits

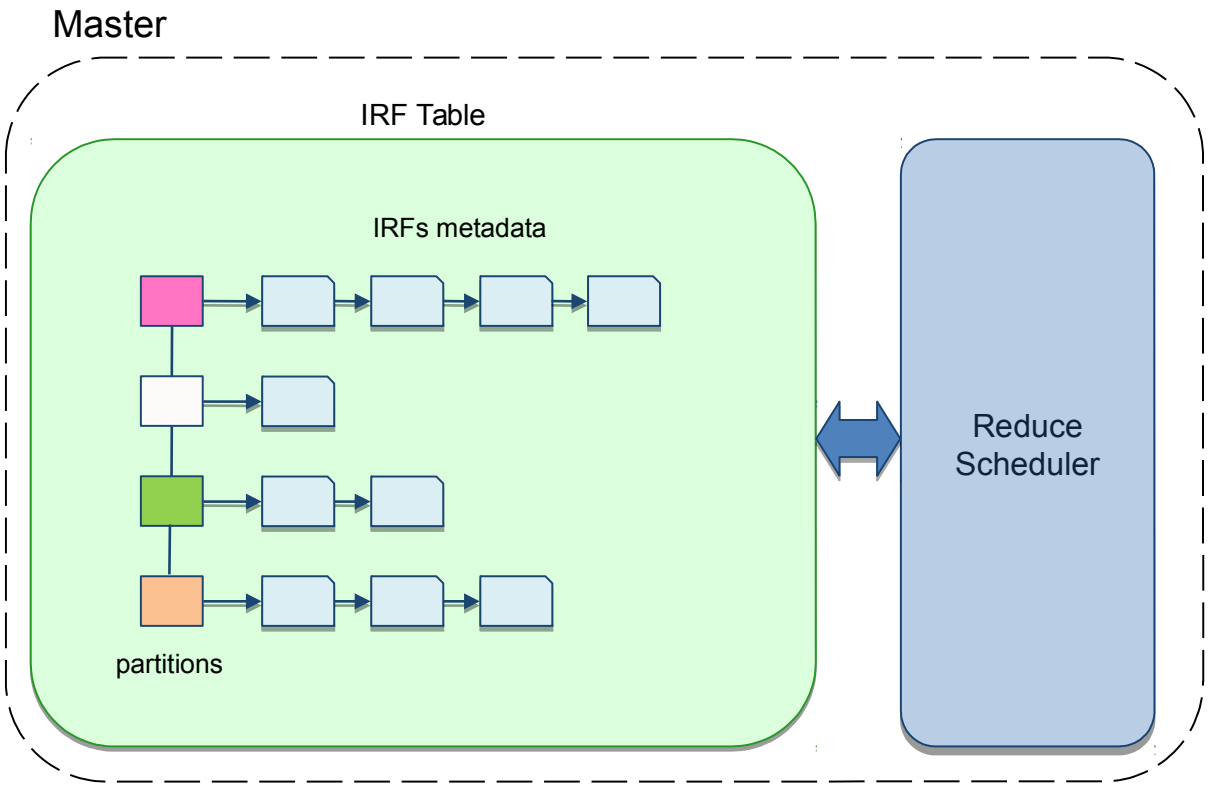


Dynamic Generation of IR Splits



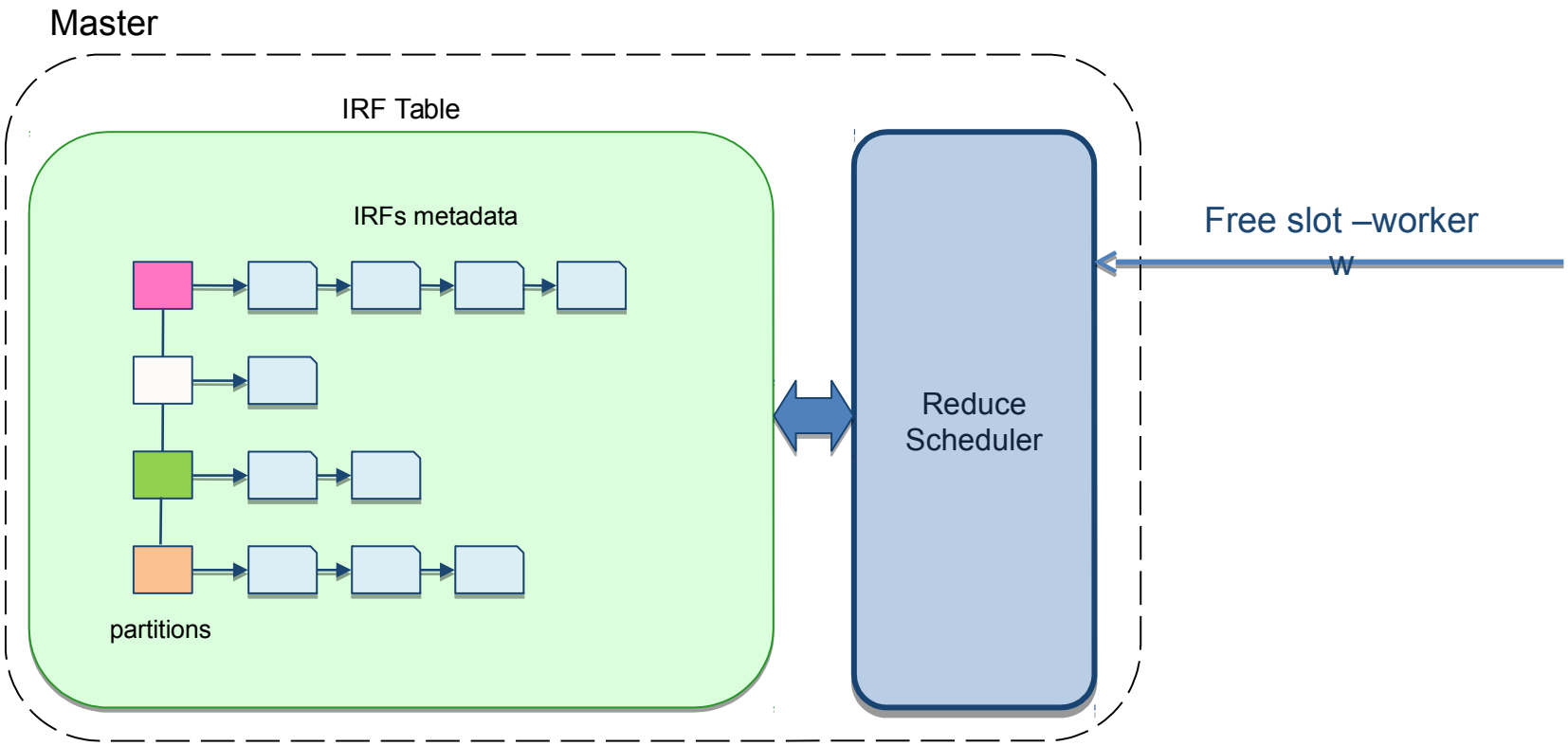
IRF fragments and metadata

Dynamic Generation of IR Splits



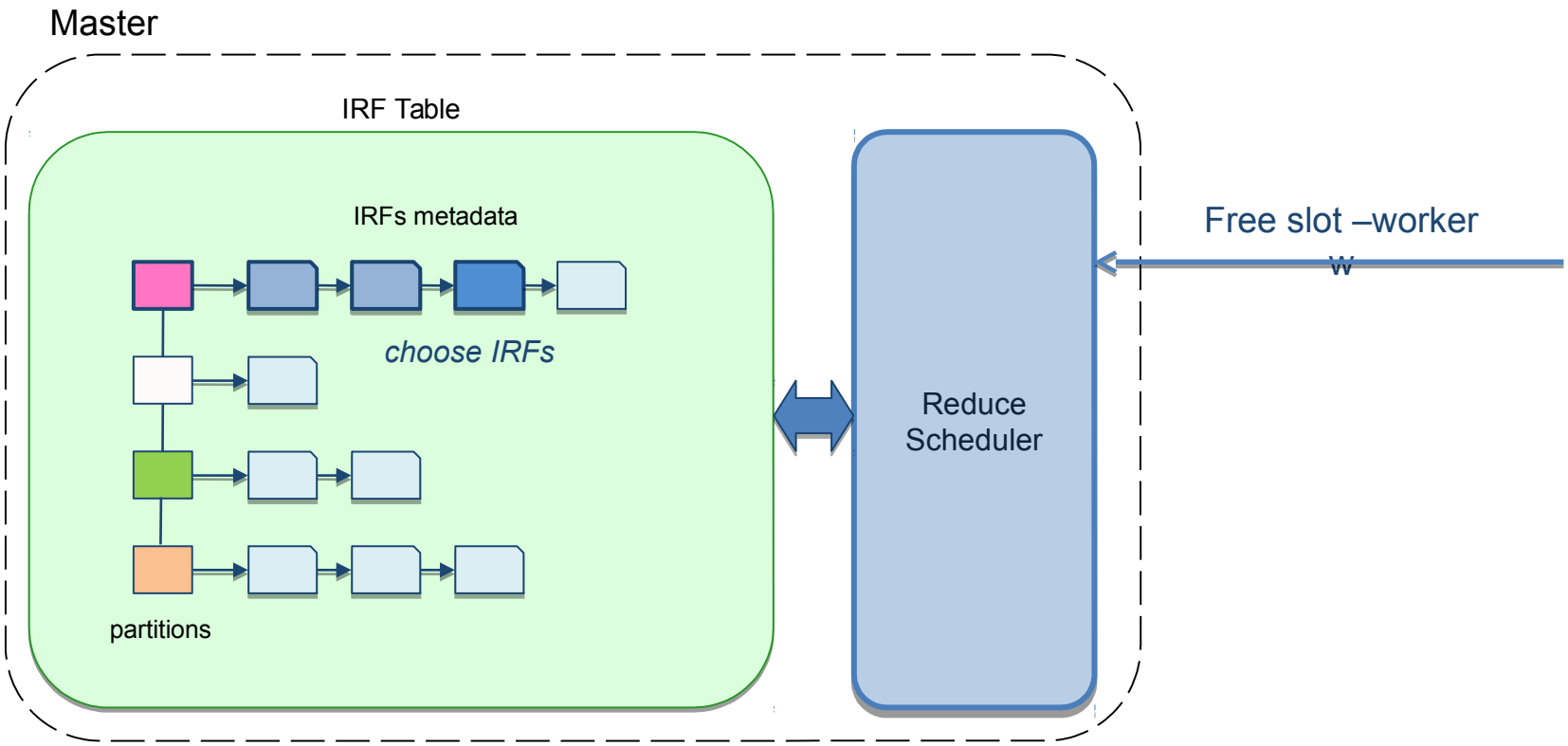
IR Split creation and scheduling

Dynamic Generation of IR Splits



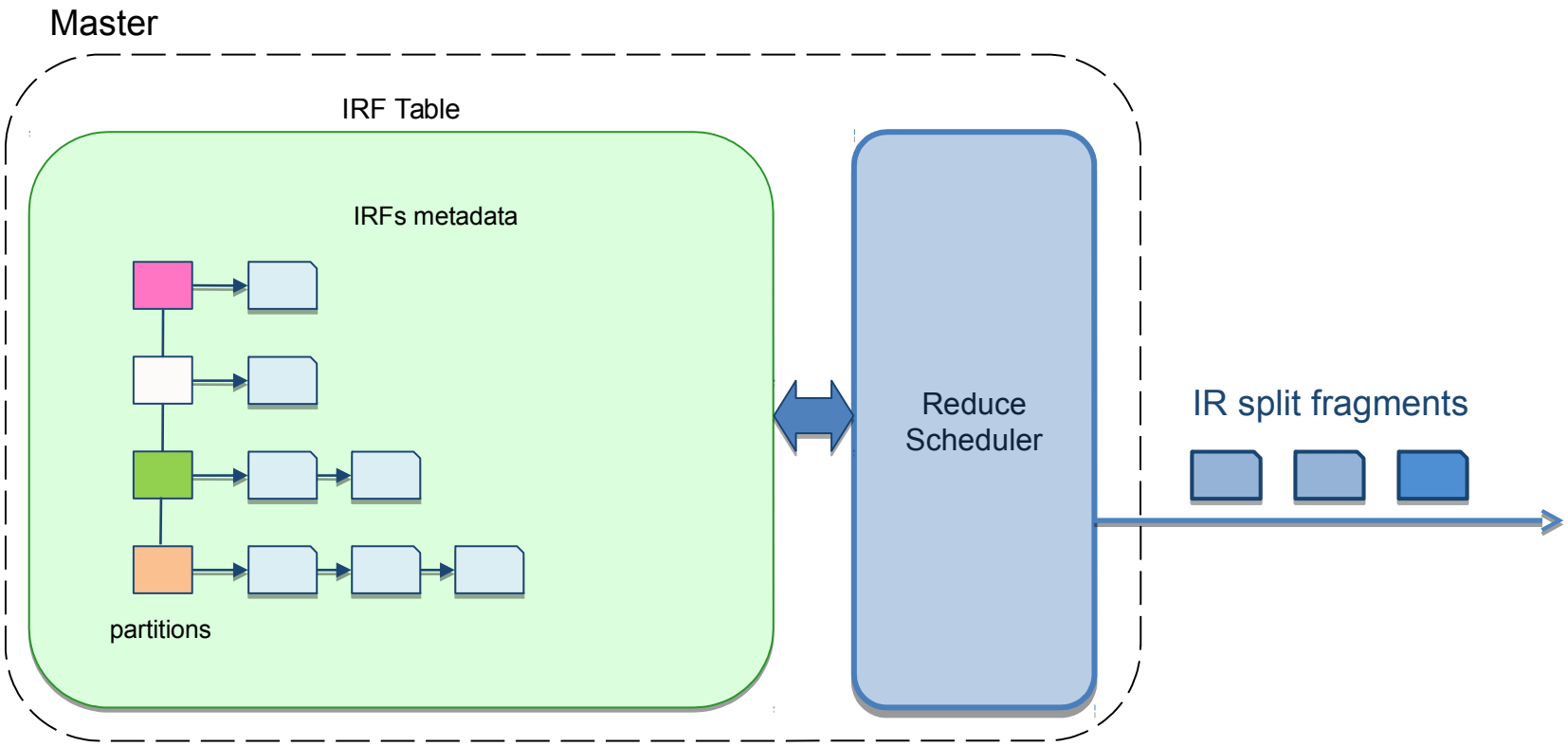
IR Split creation and scheduling

Dynamic Generation of IR Splits



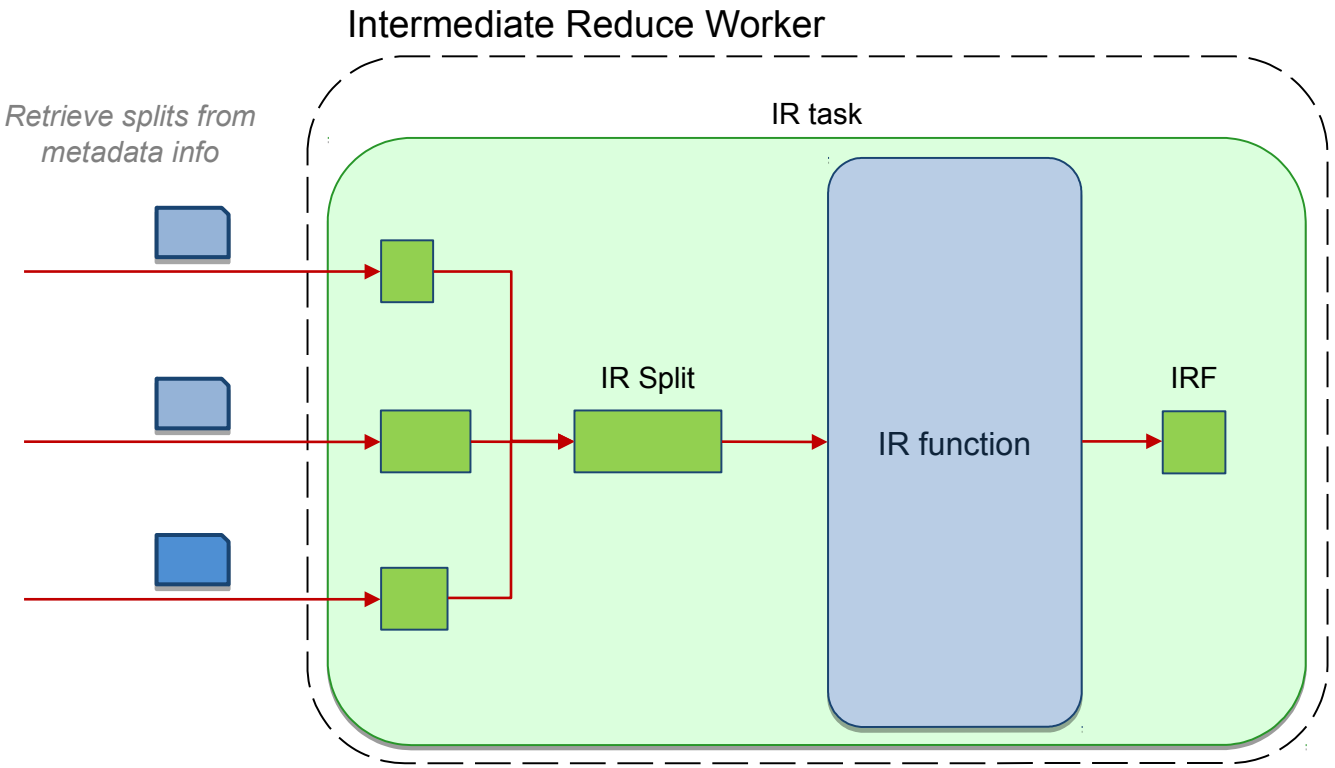
IR Split creation and scheduling

Dynamic Generation of IR Splits



IR Split creation and scheduling

Dynamic Generation of IR Splits

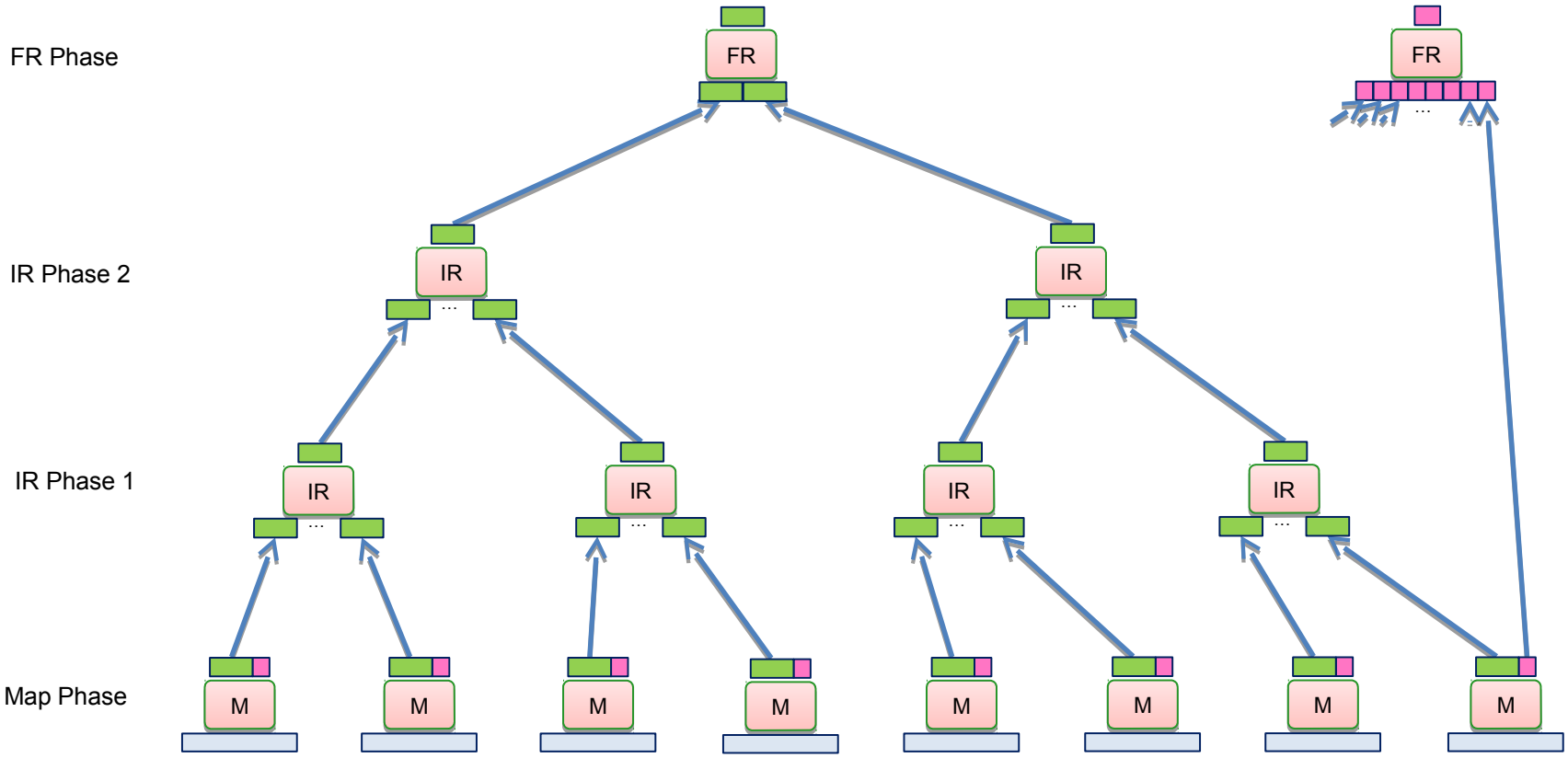


IR Split processing

Scheduling

- Reduce scheduler specifies
  1. When a partition goes to FR
  2. Which partition is chosen
  3. Which IRFs are chosen within the partition
- Implemented strategies
  - . Greedy (default)
    - Biggest partition is given priority
    - Select IR fragments until `maxIRSize` is chosen
  - . Locality-aware
    - Partition with highest amount of local data
    - Local IR fragments are chosen first

Iterative Reduce Phase



EXPERIMENTAL EVALUATION

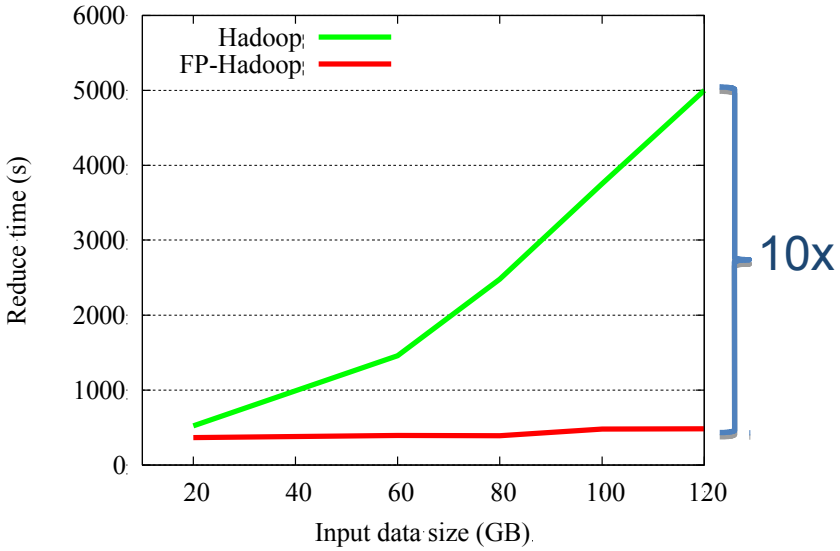
Configuration

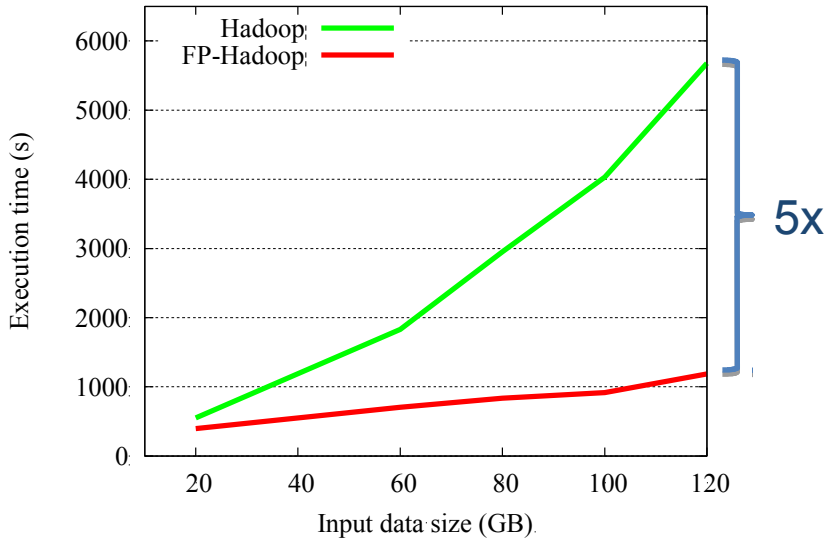
- Platform and setup
  - FP-Hadoop as prototype
  - Real-tests in G5k platform
- Metrics
  - Execution time
  - Reduce time

Queries

- Top-k % (TK)
  - k% most popular articles
  - *Datasets*
    - Synthetic from Zipfian distribution
      - f(I, S, N), S: skew, N: number of classes
    - Wikipedia article views statistics
- Inverted Index (II)
  - *Dataset*: English Wikipedia articles
  - *Dataset*: PLD graph from Web Data Commons
- WordCount (WC)
  - *Dataset*: synthetic, generated with `RandomWriter`

Scalability

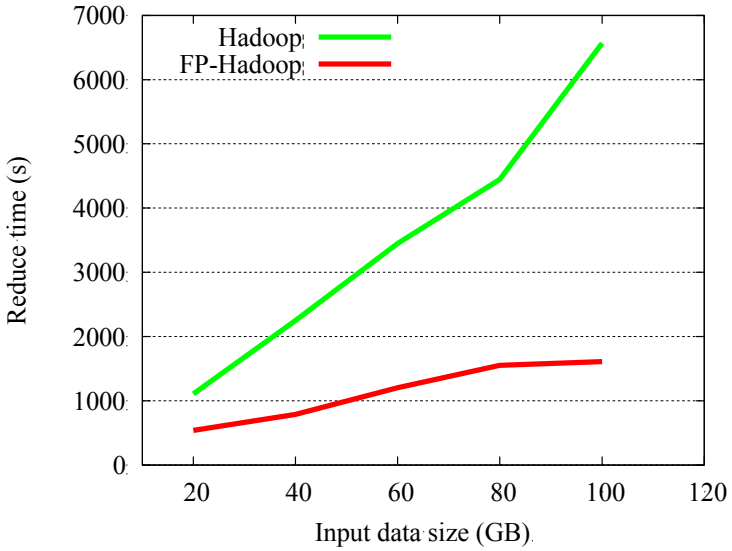


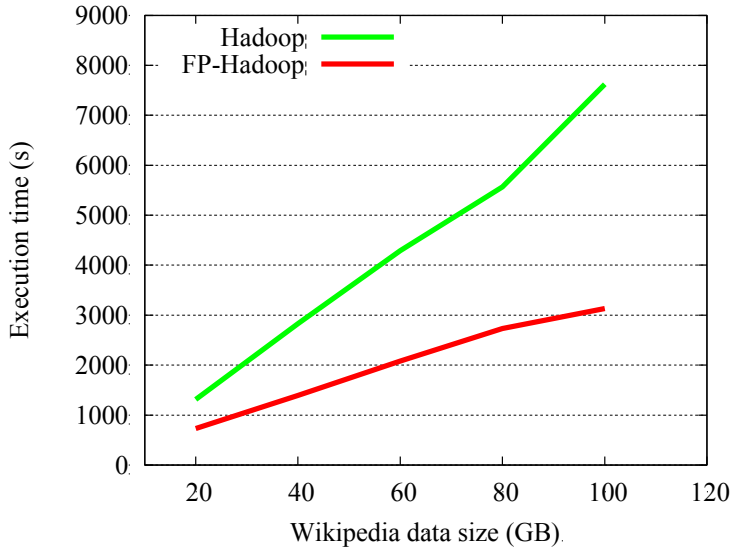


+ size : + speedup

Synthetic dataset; N = 10, S = 1; nodes = 20

Scalability

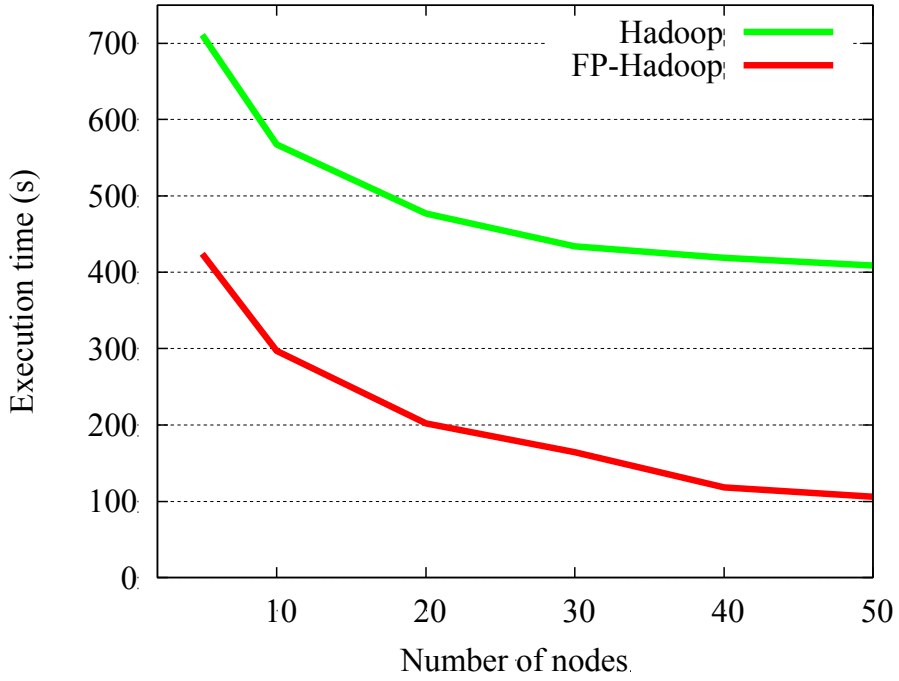




+ size : + speedup

Real dataset: Wikipedia article accesses; nodes = 20

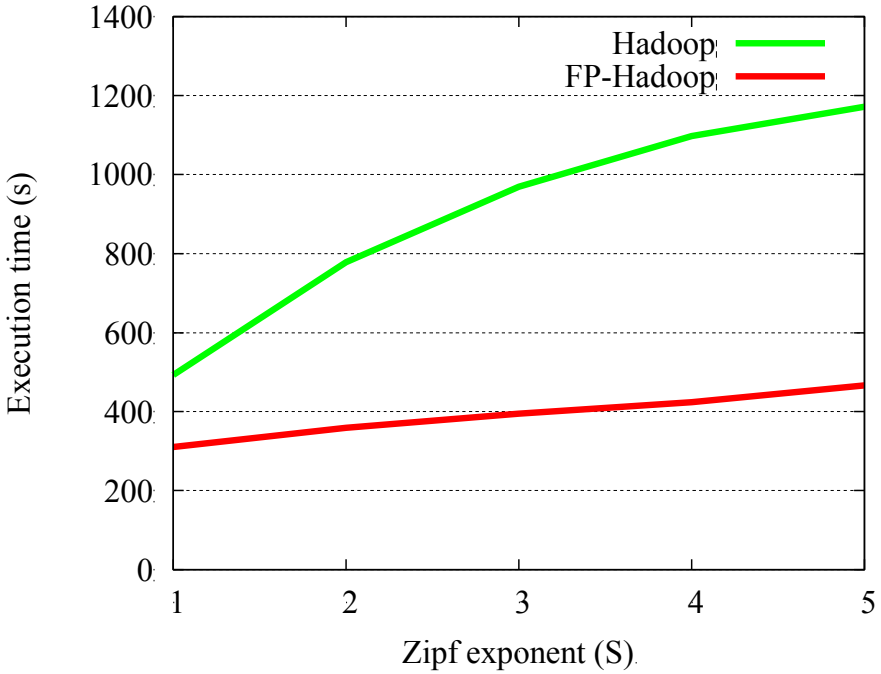
Cluster Size



+ nodes : + speedup

Synthetic dataset; N = 10, S = 1

Effect of Skew



+ skew : + speedup

Synthetic dataset; N = 10, S = 1; nodes = 20

Conclusions

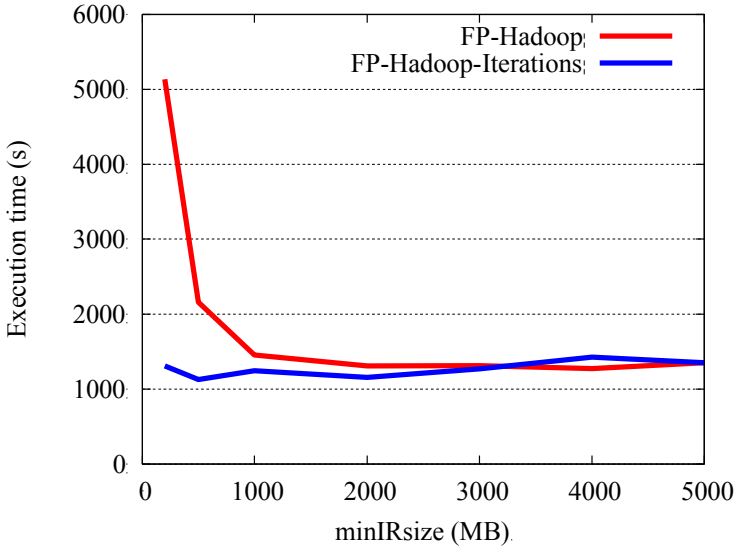
- FP-Hadoop features
  - Parallel reducing of each intermediate key
  - Distributed IR Split construction
    - Based on user-defined scheduling strategy
  - Hierarchical iterative execution
  - Non-overwhelming reduce

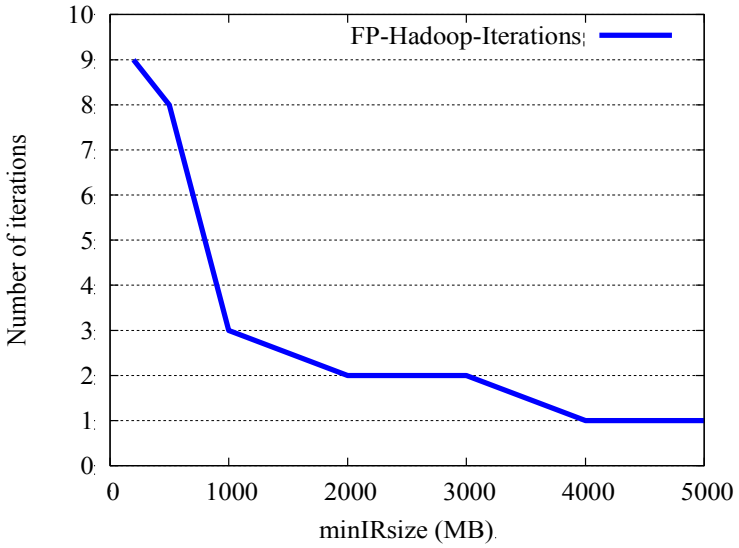
- Experimental evaluation
  - Significant gains that increase with
    - Size
    - Parallelism
    - Skew

QUESTIONS?

BACK-UP SLIDES

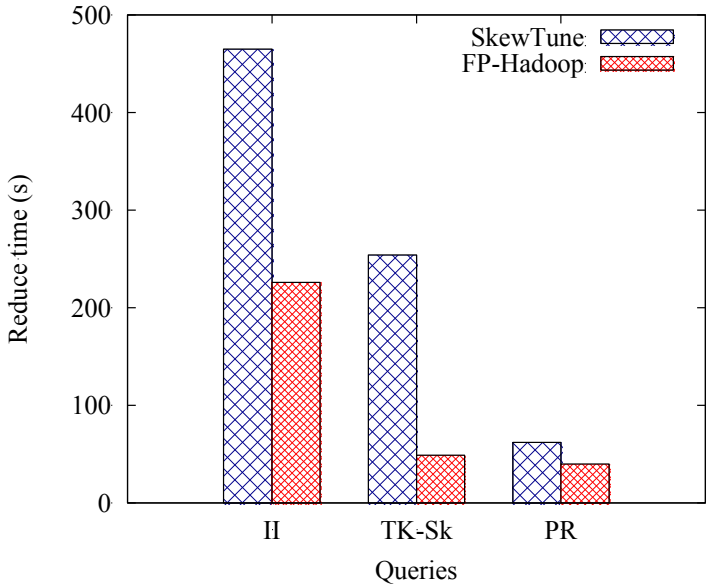
Results: Iterations and IR Split size

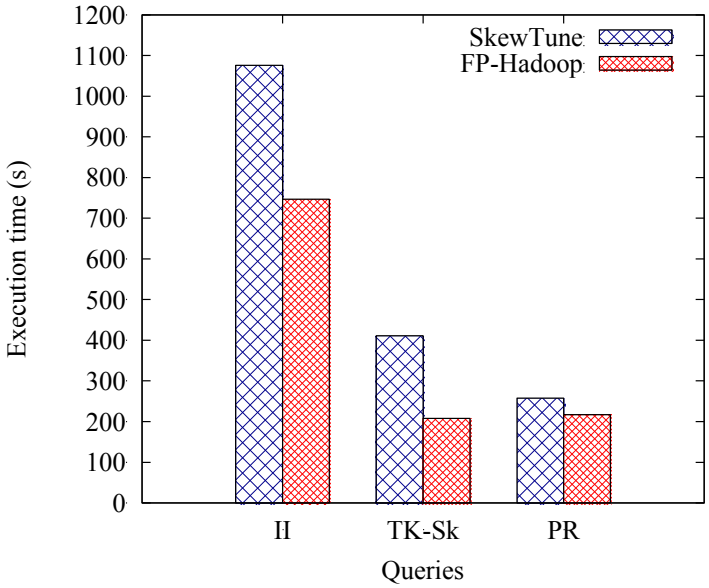




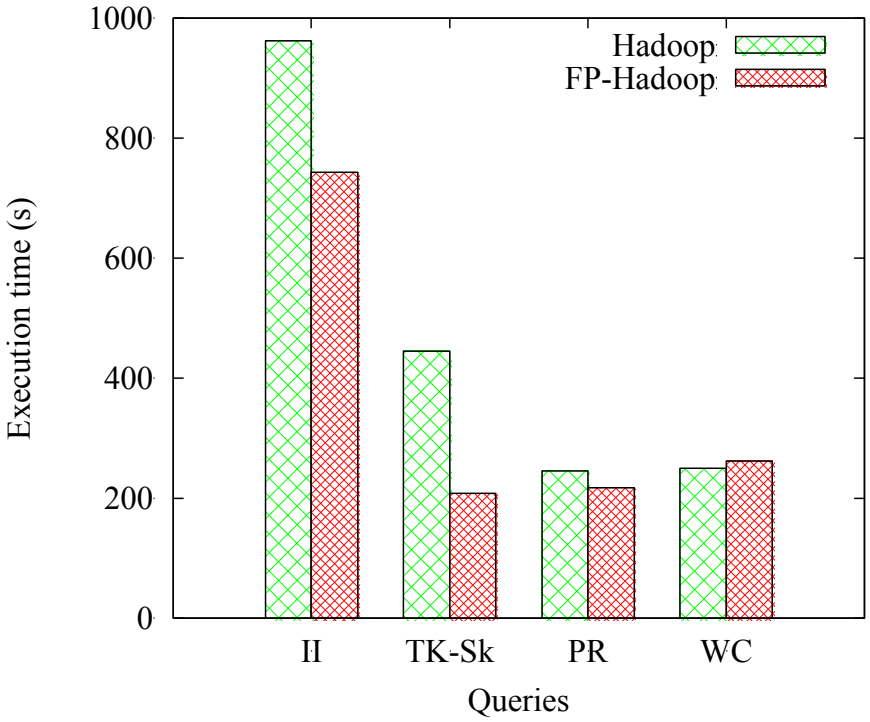
Synthetic dataset; N = 10, S = 1; Input = 20GB

Results: Comparison with SkewTune





Different Queries



nodes = 20