Predoop: Preempting Reduce Task for job execution accelerations

Yi Liang

Beijing University of Technology

2014.09.05

Outline

- Motivation
- Main Contributions
- Performance Evaluation
- Conclusion and Future work

Outline

- Motivation
- Main Contributions
- Performance Evaluation
- Conclusion and Future work

Background

- Hadoop Map/Reduce

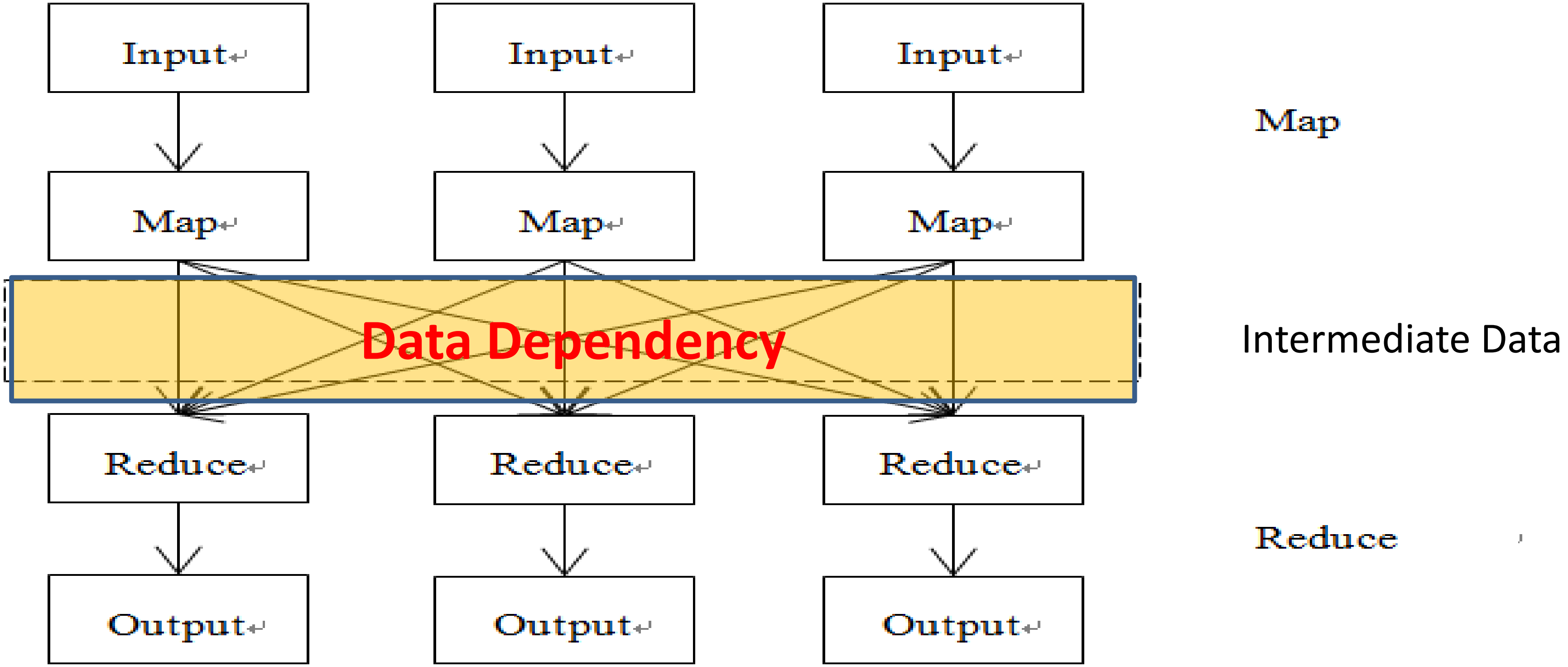
□ Programming the commodity computer clusters to perform the large-scale data processing

- Scheduling granularity --- task level

- Resource allocation --- once allocated, held till task ends

Motivation: Idle period

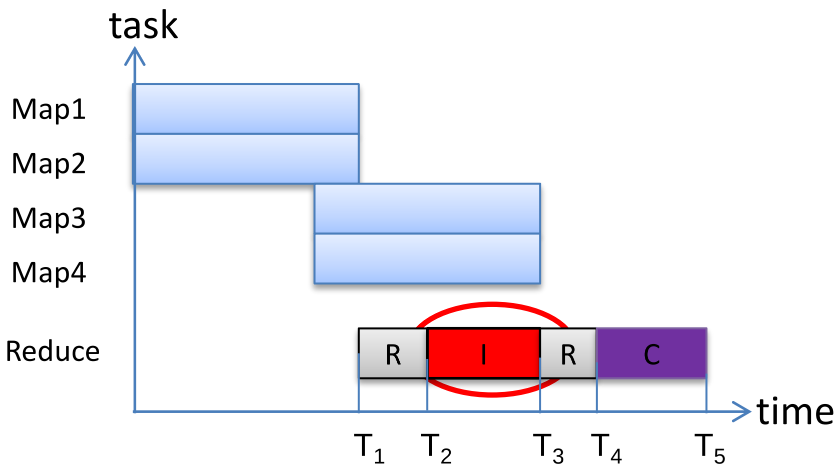
- Data dependency among Map/Reduce tasks --- map output -→ reduce input



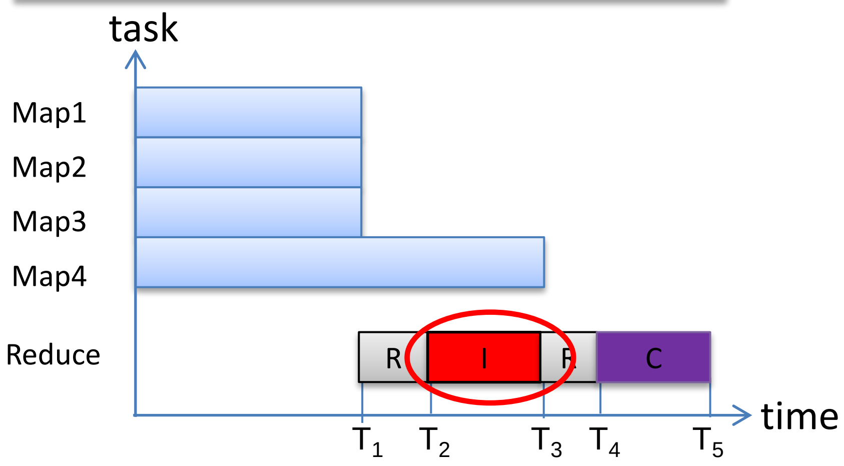
Motivation

- Idle time of reduce tasks

Different Start time of map tasks



Different execution time of map tasks



R: Read data
I: Idle
C: Compute

Motivation

- Running 20 *WordCount* map/reduce jobs on a 12-node cluster

Job Number	5	15	20
Idle time of reduce task / total execution time of reduce task	31.2%	31.8%	44.5%
Idle time of reduce task / total execution time of job	13.9%	23.3%	15.7%

Outline

- Motivation
- Main Contributions
- Performance Evaluation
- Conclusion and Future work

Basic idea of Predoop

- Idea 1: Preempt the idle *reduce* tasks to mitigate the idle time
- Idea 2: Allocate the *preempted resources* to map tasks on schedule to accelerate the job execution

Main Contributions

- The preempting-resuming model for the reduce task
  - To determine the candidate time point of reduce task preempting and resuming
- Preemption-aware task scheduling
  - Scheduling strategy to allocate preempted resources
- The preemptive mechanism for map tasks and reduce tasks
  - To enable the preemption of map tasks and reduce tasks

The preempting-resuming model

- Basic idea
  - Once the length of a reduce task's idle time is *long* enough, the start point of its idle time can be determined as the candidate preempting time point.
- Determination factors
  - the estimation of *the start point* of reduce task's idle time
  - the estimation of *the length* of reduce task's idle time

The preempting-resuming model

- Estimation of the start point of reduce task's idle time
  - In predoop, the estimated start time point is the candidate time point to preempt a reduce task

The preempting-resuming model

- Estimation of the length of reduce task’s idle time
  - ***Remaining execution time of map task (T_{rm})*** is calculated based on the hypothesis that map task spends the same time on processing each data element.

The preempting-resuming model

- Preempting model of reduce tasks
  - Idea: Once the minimum possible length of a reduce task's idle time accounts for a specific proportion(**D_P**) of the average execution of the map tasks
  - The start point is determined as the preempting point.

The preempting-resuming model

(2) Resuming model of reduce task

- ***Condition 1***
  - A reduce task can be resumed only when a specific proportion(**D_r**) of its depended map tasks completed since its last preemption.
- ***Condition 2***
  - All map tasks allocated with the preempted computing resource of the reduce task are not in the intermediate data partition phase.

Preemption-aware task scheduling

- Preemption-aware task scheduling
  - **Basic idea**

(1) Queue map/reduce jobs in FIFO way

(2) Perform the scheduling based on **three rules**

(3) Assign the preempted resources to map tasks with the consideration of data locality

Preemption-aware task scheduling

- **Three Scheduling rules**

Rule 1

The allocation of preempted resource is prior to the regular resource.

Rule 2

The preempted resource can only be allocated to the map tasks.

Rule 3

The resources allocated to a map task can only released from one preempted reduce task.

Outline

- Motivation
- Main Contributions
- Performance Evaluation
- Conclusion and Future work

Performance evaluation setup

- Experimental Methodology
  - Comparison: *Predoop vs. YARN* with FIFO scheduler
  - Workload
    - **Single-application** workload: Wordcount and Sort from BigDataBench
    - **Mix** workloads from SWIM
  - Cluster
    - 13-node cluster, Each node is equipped with two Intel(R) Pentium(R) 4 cpus, 3GB memory and one 160GB SATA hard driver.
    - HDFS Block: 64MB
  - Performance Metric
    - Average Turnaround Time

Evaluation target

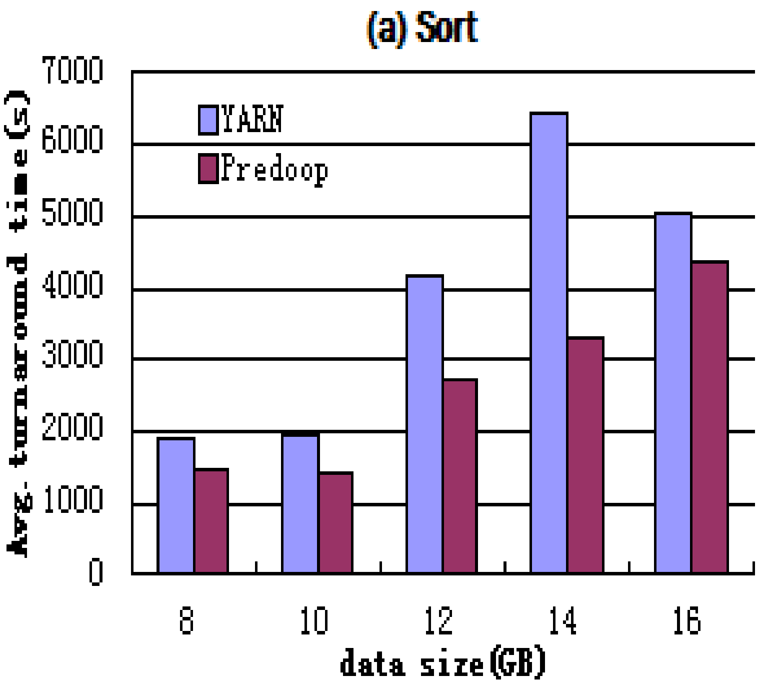
- Evaluation
  - Performance of single-application workloads
  - Performance of the mix workloads
  - Performance sensitivity to the threshold configurations
  - Performance scalability

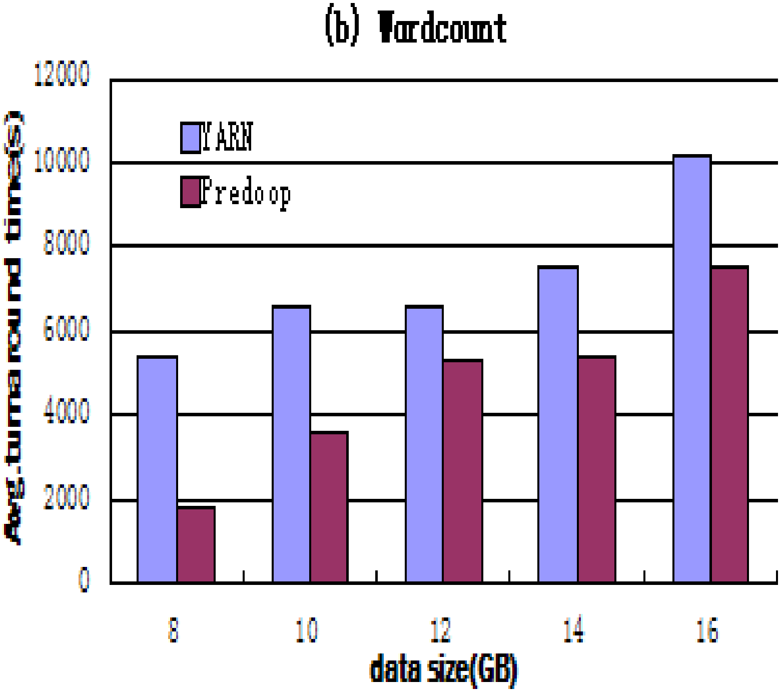
Performance of single-application workloads

Configuration

- the input data size set as: 8GB, 10GB, 12GB, 14GB, 16GB.
- the reduce task number set as 8 for each job
- Memory requirement of each task set as 1024MB as default.
- D_p : 20%, D_r :40%

Performance of single-application workloads





Performance of the mix workloads

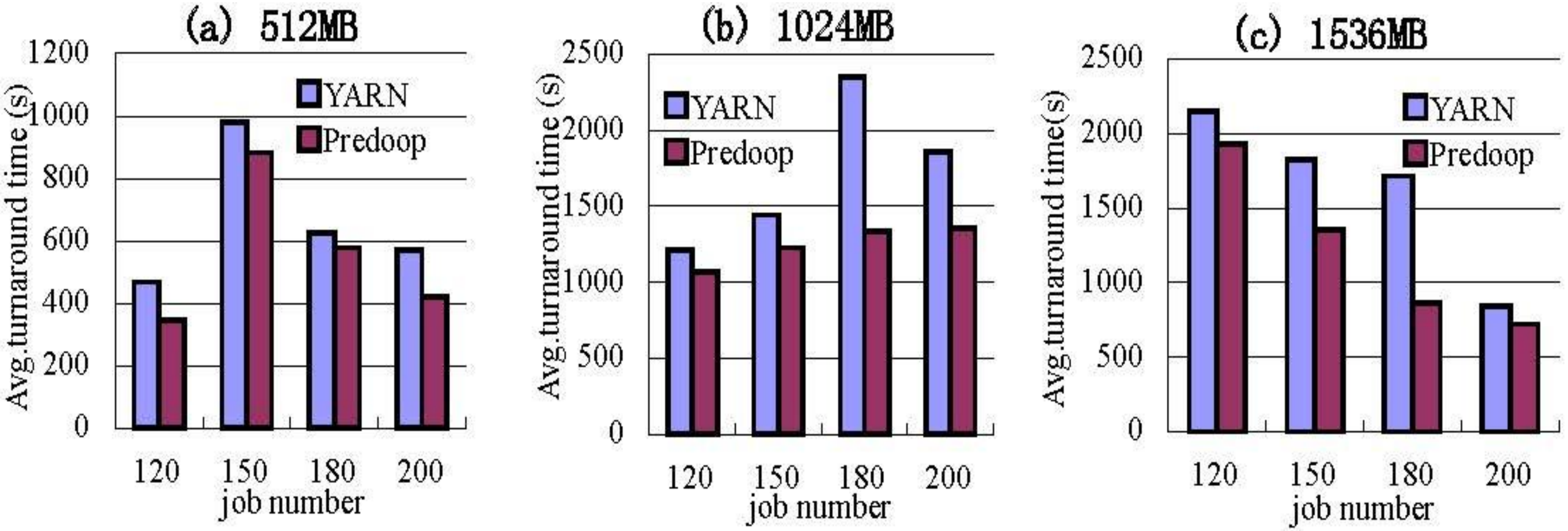
Configuration

- 4 mix workloads from SWIM
- the memory requirement of each map and reduce task varies as 512MB, 1GB (default set in YARN), and 1.5GB
- D_p : 20%, D_r :40%

Performance of the mix workloads

	Bin1	Bin2	Bin3	Bin4
Job number	120	150	180	200
Total size of Map Input data (GB)	46.66	64.19	72.32	82.94
Total size of Intermediate data (GB)	6	6.25	6.47	6.58
Total size of Reduce Output data (GB)	1.36	1.44	2.26	7.19

Performance of the mix workloads

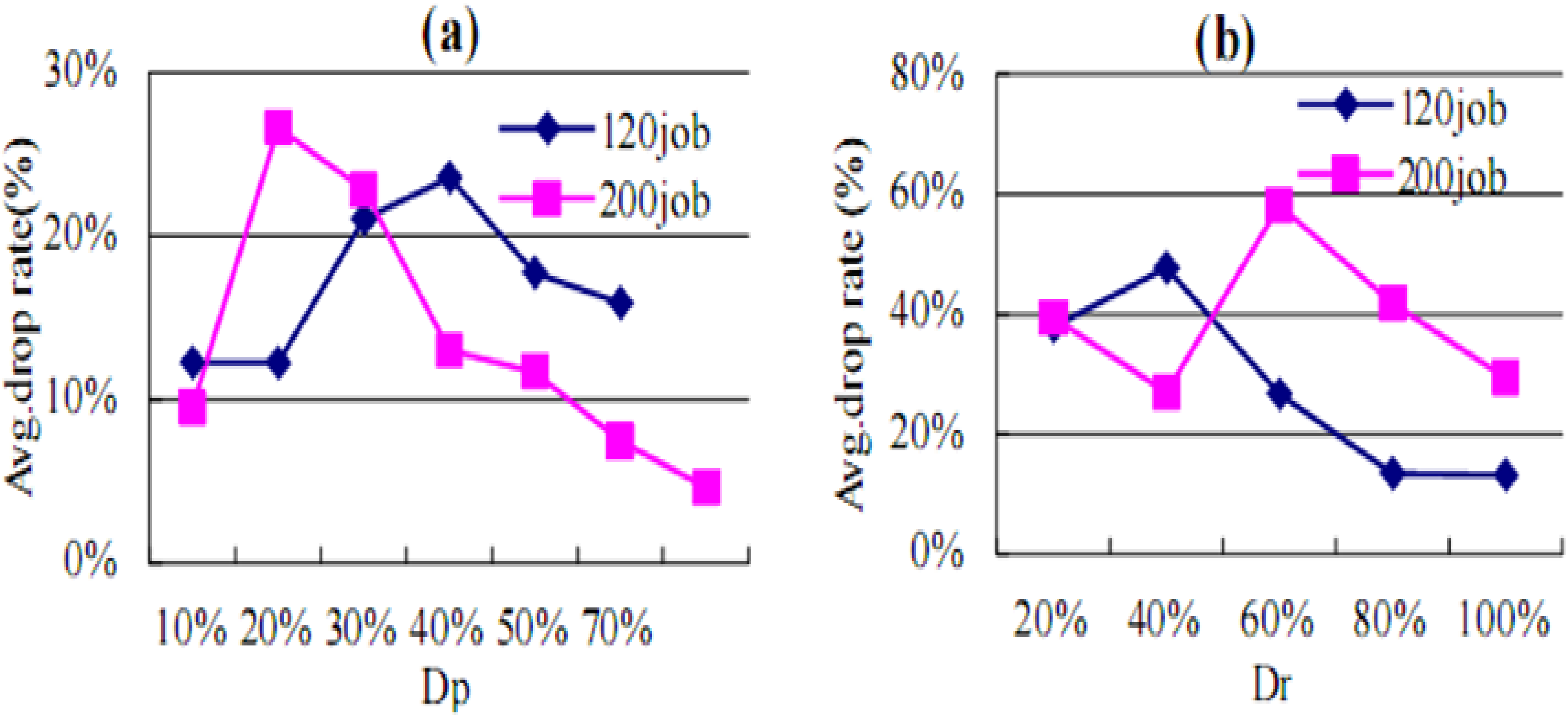


Performance sensitivity to the threshold configurations

Configuration

- – Choose Bin1 and Bin4
- – Dp varies as 10%,20%, 30%, 40%, 50%, 60%, 70%
- – Dr varies as 20%, 40%, 60%, 80%, 100%

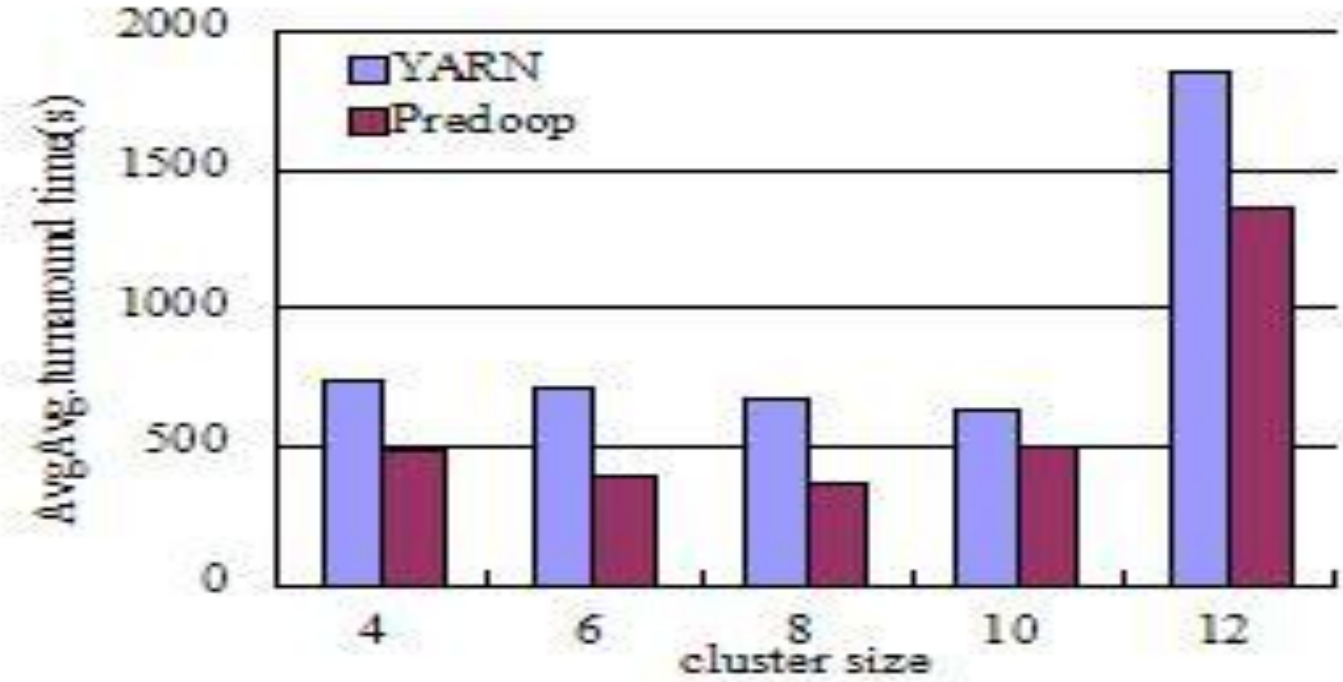
The Percentage of Job Acceleration



Performance Scalability

- Configuration
  - generate five groups of workloads for the cluster size of 4,6,8,10,12
  - For each group, generate three workloads with 120 jobs each
  - calculate the average job turnaround time of the corresponding three workloads

Performance Scalability



Conclusion and Future work

- Predoop: Preempting resources of idle reduce tasks to on-schedule map tasks to accelerating job execution
  - Preempting-resuming model of reduce task
  - Preemption-aware task scheduling
  - Preemptive mechanism of reduce tasks and map tasks

- Ongoing work
  - Improving the preempting-resuming model for more complex map/reduce jobs
  - The online adjustment of the threshold in the preemption model.

Outline

- Motivation
- Main Contributions
- Performance Evaluation
- Conclusion and Future work

Thanks