# DREAMS: Dynamic REsource Allocation for MapReduce with Data Skew

Zhihong Liu[*+], Qi Zhang[+], Mohamed Faten Zhani[+], Raouf Boutaba[+],
Yaping Liu[*] and Zhenghu Gong[*]

[*]National University of Defense Technology, China
[+]University of Waterloo, Canada

# Outline

- **Introduction**
- **Our solution**
- **Evaluation**
- **Conclusion**

# Introduction

Introduction

# Introduction

## How much data is created every minute?

Google: **4,000,000** search queries

facebook: **2,460,000** pieces of new content are shared

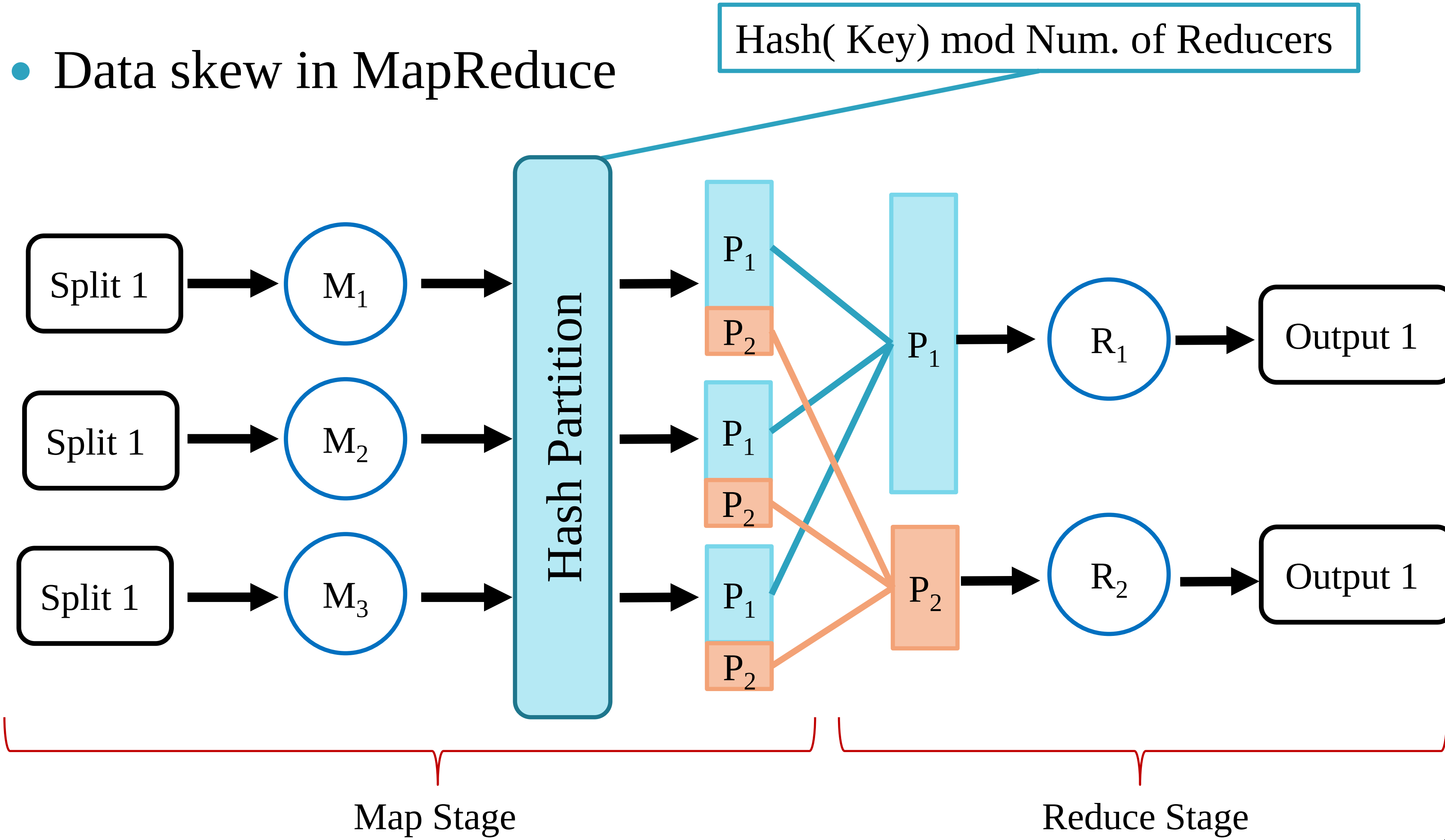
twitter: **270,000** tweets

YouTube: **72 hours** of new videos are uploaded


**Source:** Josh James. Data Never Sleeps 2.0, https://www.domo.com/blog/2014/04/data-never-sleeps-2-0/

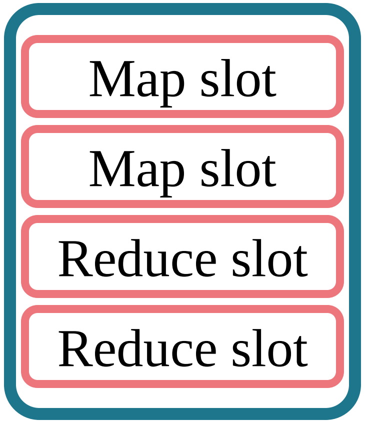
# Introduction

- MapReduce is a popular framework for big data analytics
- Data skew in MapReduce

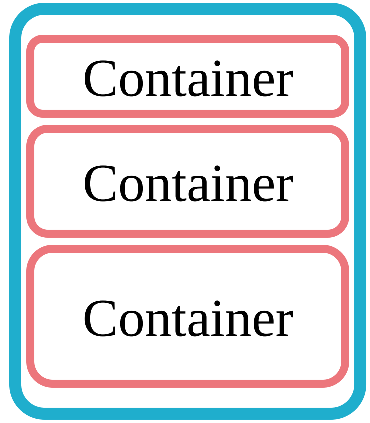


# Introduction

- Resource management schemes in Hadoop




slot-based




container-based

- Limitations
  - Assume the same kind of tasks (map or reduce) in a job has uniform resource requirement
  - Do not support dynamic resource allocation to each task

**Incur** → 1) Prolonging the job completion time

2) Reducing the resource utilization

# Introduction

- Existing solutions
  - Rebalance the key-value pairs among reduce tasks based on the key distribution
    - cause a synchronization barrier
  - Run speculative tasks on other machines
    - may waste resource while omitting the correlation between task load and progress rate
  - Repartition the unprocessed load of slow tasks to another tasks
    - incur large overhead to repartition the load

# Our solution

# DREAMS

**Idea** → Dynamically **adjusting the container size** based on the load of each reduce task, thereby mitigating the negative impact of data skew

**Benefits:**

- Eliminates the overhead of rebalancing the load
- Mitigates data skew at run-time
- Simple to implement

**Limitation:**

- Needs job profiles

# Challenges

- How to predict the load of each reduce task at run-time?
- How much amount of resources should be allocated to each reduce task?

# Challenge One

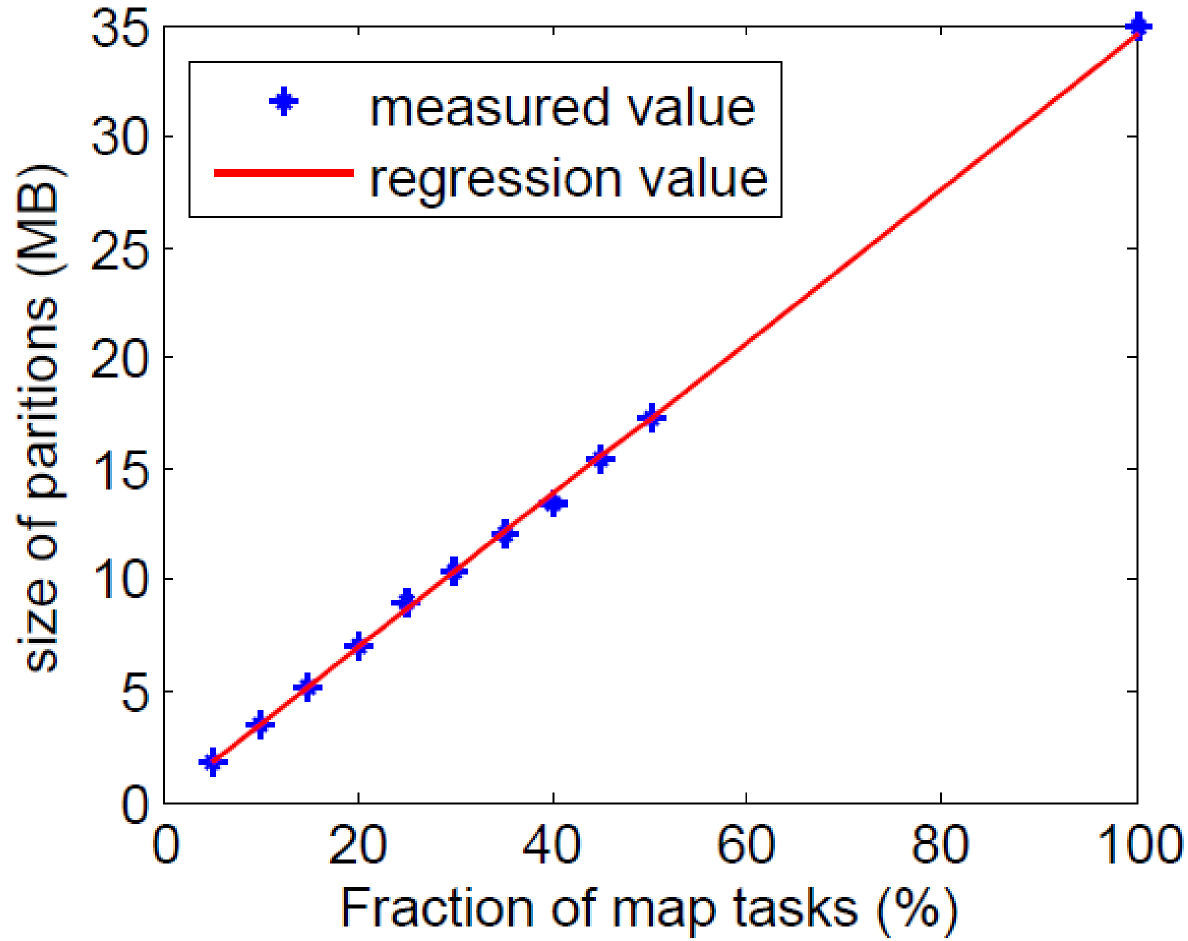
# How to predict the load of each reduce task

Load of the reduce task

- Usin

(1)

- $F^j$ is … leted
- $S_i^j$ is … ompleted map tasks
- Once … ze the linear mod




InvertedIndex on Wikipedia dataset

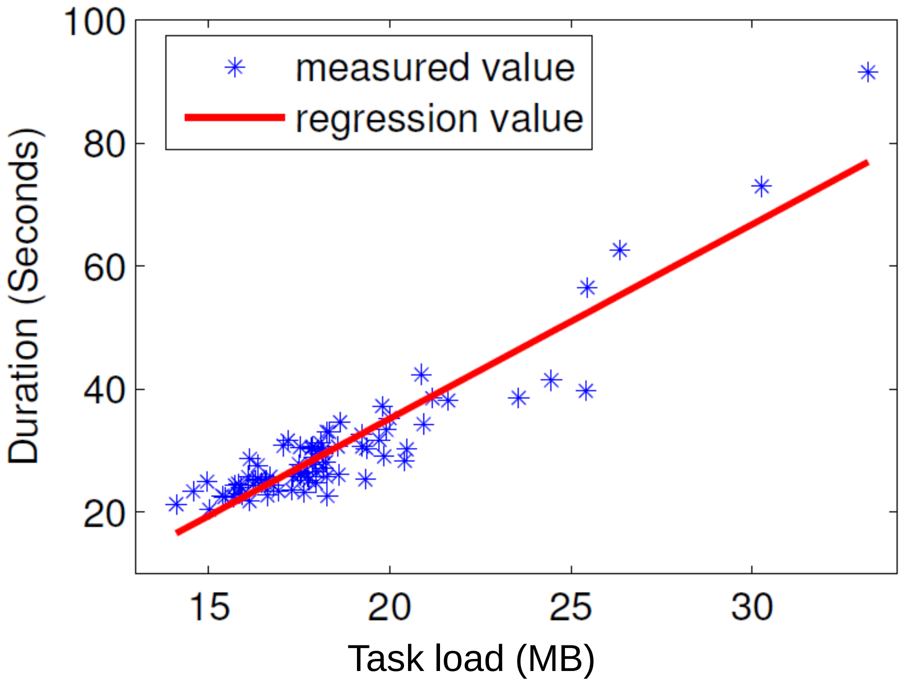
# Challenge Two

# How much resource should be allocated?

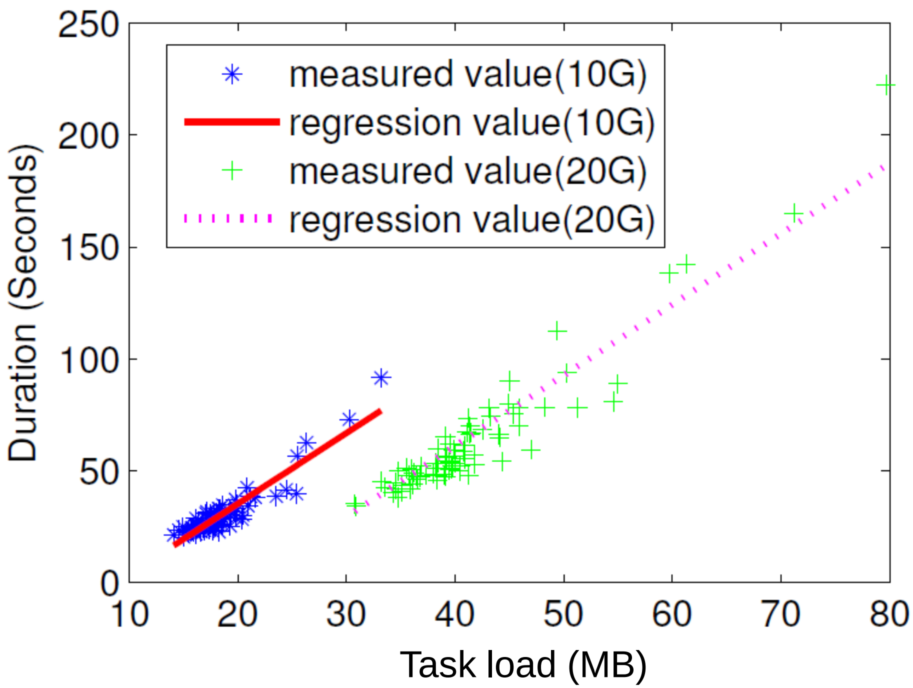
Task duration = *f*( Task load, Amount of resource )

- We need to know:
  - What is the relationship between the task duration and the task load?
  - What is the relationship between the task duration and the resource allocation?

# The relationship between task duration and task load



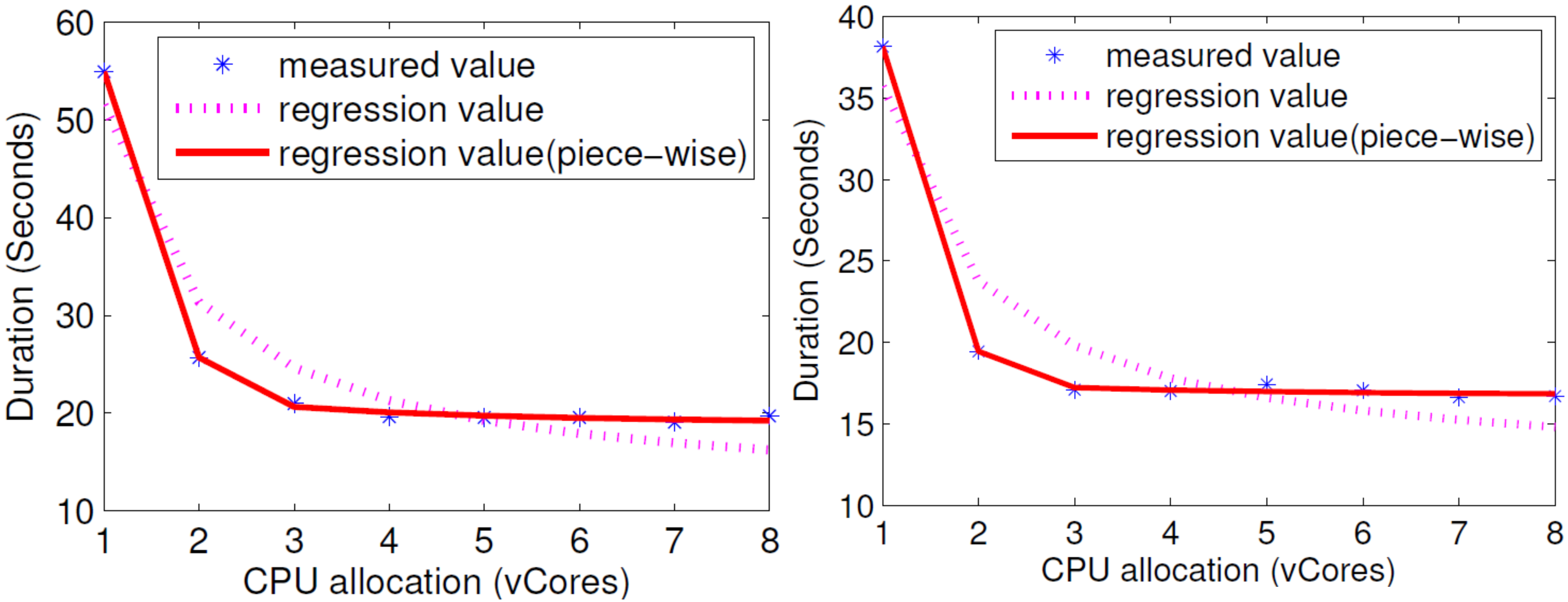

(a) InvertedIndex 10G




(b) InvertedIndex 10 and 20G

The task duration is linearly correlated with the task load

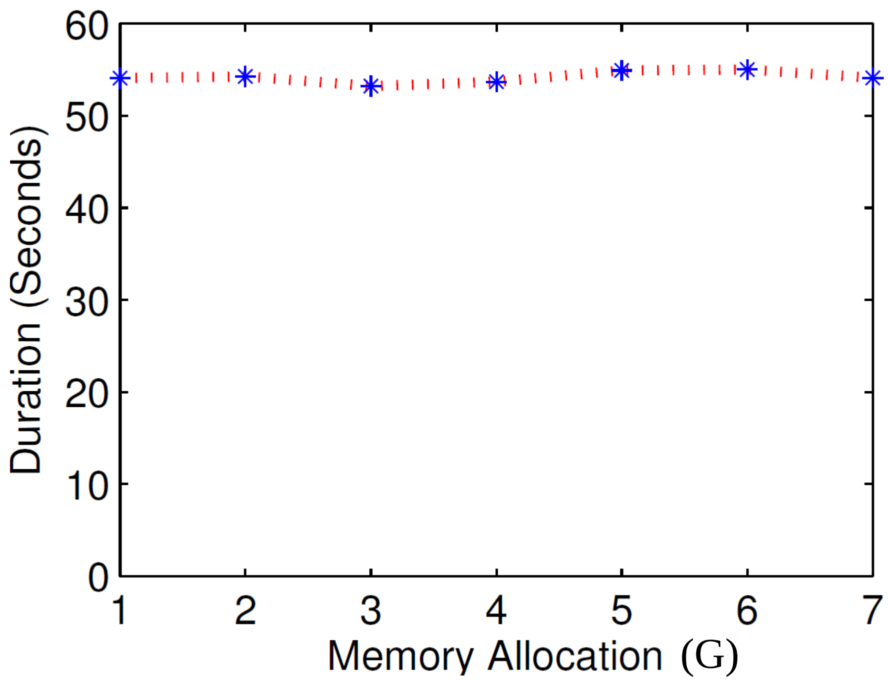
# The relationship between task duration and CPU



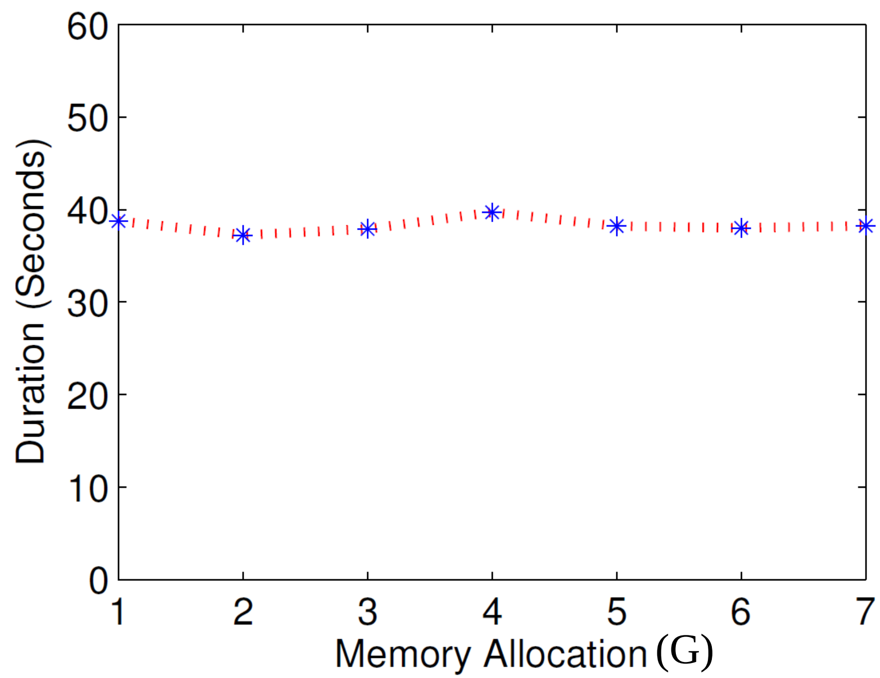

(a) Sort10G (b) InvertedIndex 10G

The task duration is inverse proportionally correlated with the CPU allocation

# The relationship between task duration and memory




(a) Sort10G




(b) InvertedIndex 10G

Memory is not the bottleneck resource for this workload

# Reduce task performance model

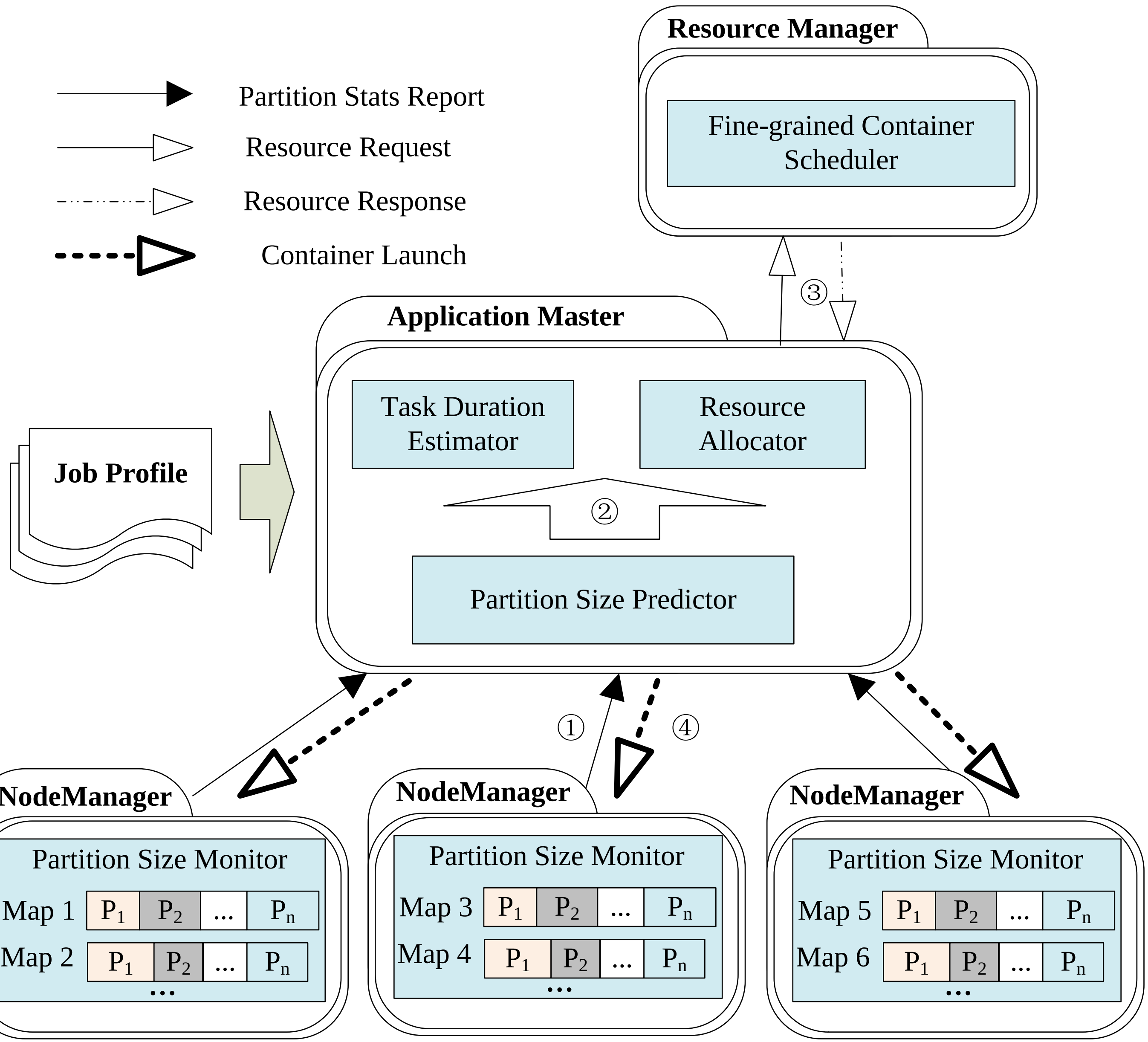
$$T_i = \begin{cases} \alpha + \beta P_i + \gamma D + \dfrac{\zeta}{Alloc_i^{cpu}} + \dfrac{\eta P_i}{Alloc_i^{cpu}} + \dfrac{\xi D}{Alloc_i^{cpu}} & Alloc_i^{cpu} \le \varphi \\ \alpha' + \beta' P_i + \gamma' D + \dfrac{\zeta'}{Alloc_i^{cpu}} + \dfrac{\eta' P_i}{Alloc_i^{cpu}} + \dfrac{\xi' D}{Alloc_i^{cpu}} & Alloc_i^{cpu} > \varphi \end{cases}$$

Task duration = *f(* Task load, Amount of resource *)*

| | |
|---|---|
| $T_i$ | task duration |
| $P_i$ | task load |
| $D$ | sum of all reduce loads |
| $Alloc_i^{cpu}$ | CPU allocation |

- Use non linear regression to determine the coefficient factors
- Each tuple of ($T_i$, $P_i$ , $D$, $Alloc_i^{cpu}$) is a training data
- This performance model is used as a job profile for allocating resource

# Architecture of DREAMS



# Evaluation

# Evaluation

- Accuracy of reduce task load prediction
  - Metric

$$ARE = \frac{1}{N}\sum_{i=1}^{N}\frac{\left|P_i^{pred} - P_i^{measrd}\right|}{P_i^{measrd}}$$

  - Results

Different datasets

Different *slowstart* settings

| APP | Input Size(GB) | $\delta =$ 0.05 | $\delta =$ 0.06 | $\delta =$ 0.07 | $\delta =$ 0.08 | $\delta =$ 0.09 | $\delta =$ 0.10 |
|---|---|---|---|---|---|---|---|
| Sort | 10 | 2.28% | 2.09% | 1.94% | 1.81% | 1.71% | 1.71% |
| Sort | 20 | 1.60% | 1.43% | 1.32% | 1.26% | 1.17% | 1.13% |
| Sort | 50 | 1.1% | 1.01% | 0.94% | 0.90% | 0.84% | 0.78% |
| IvIndex | 9.01 | 8.2% | 7.63% | 7.05% | 7.05% | 6.43% | 5.87% |
| IvIndex | 21.02 | 5.62% | 5.25% | 5.08% | 4.79% | 4.53% | 4.38% |
| IvIndex | 49.04 | 4.73% | 4.43% | 4.21% | 4.07% | 3.90% | 3.70% |

# Accuracy of reduce task performance model

- Metric

$$ARE = \frac{1}{k}\sum_{l=1}^{k}\frac{\left|T_l^{pred} - T_l^{measrd}\right|}{T_l^{measrd}}$$

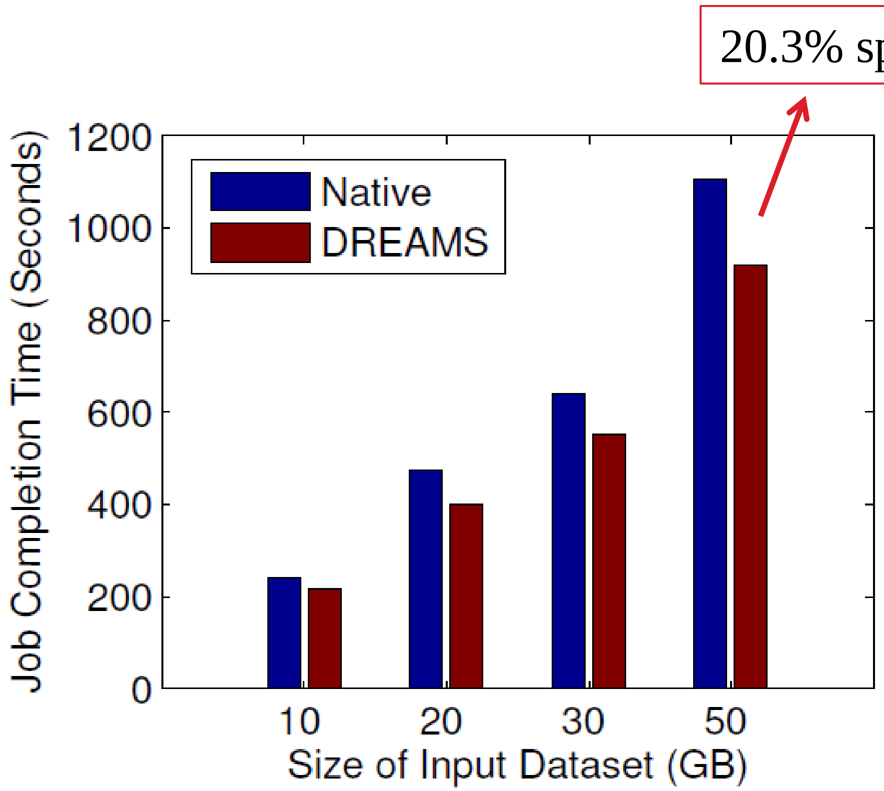
- Results

Different datasets

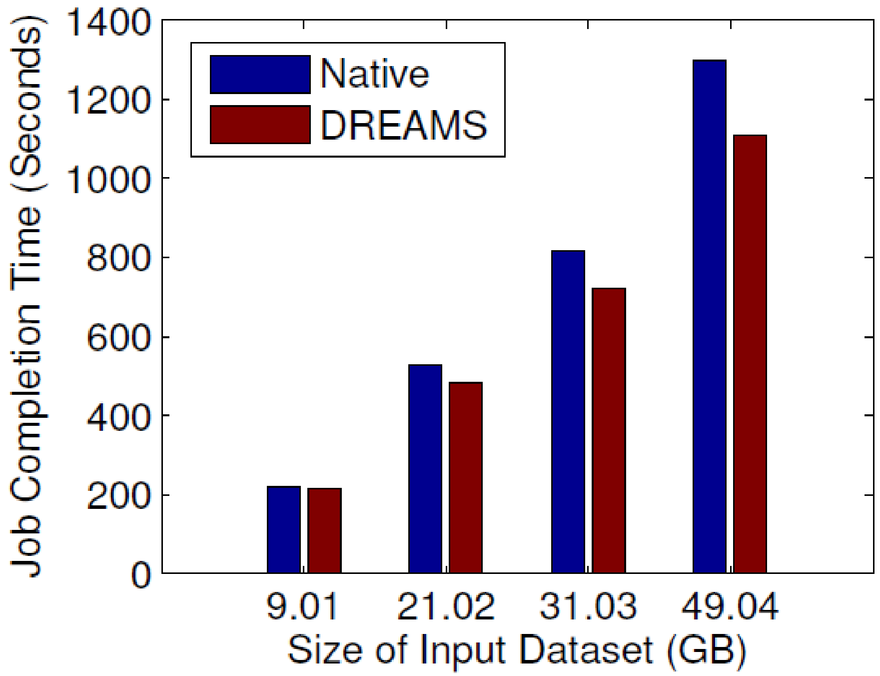
Two kinds of validations

| Application | Input Data Type | Input Data Size(GB) | Test-on-training | Test-on-unknown |
|---|---|---|---|---|
| Sort | Synthetic | 10 | 5.44% | 9.36% |
| Sort | Synthetic | 20 | 7.91% | 10.62% |
| Sort | Synthetic | 30 | 12.28% | 16.38% |
| Sort | Synthetic | 50 | 11.09% | 19.57% |
| InvertedIndex | Wikipedia | 9.01 | 11.67% | 13.97% |
| InvertedIndex | Wikipedia | 21.02 | 12.89% | 13.31% |
| InvertedIndex | Wikipedia | 31.03 | 14.67% | 16.44% |
| InvertedIndex | Wikipedia | 49.04 | 14.56% | 17.06% |

# Job performance evaluation



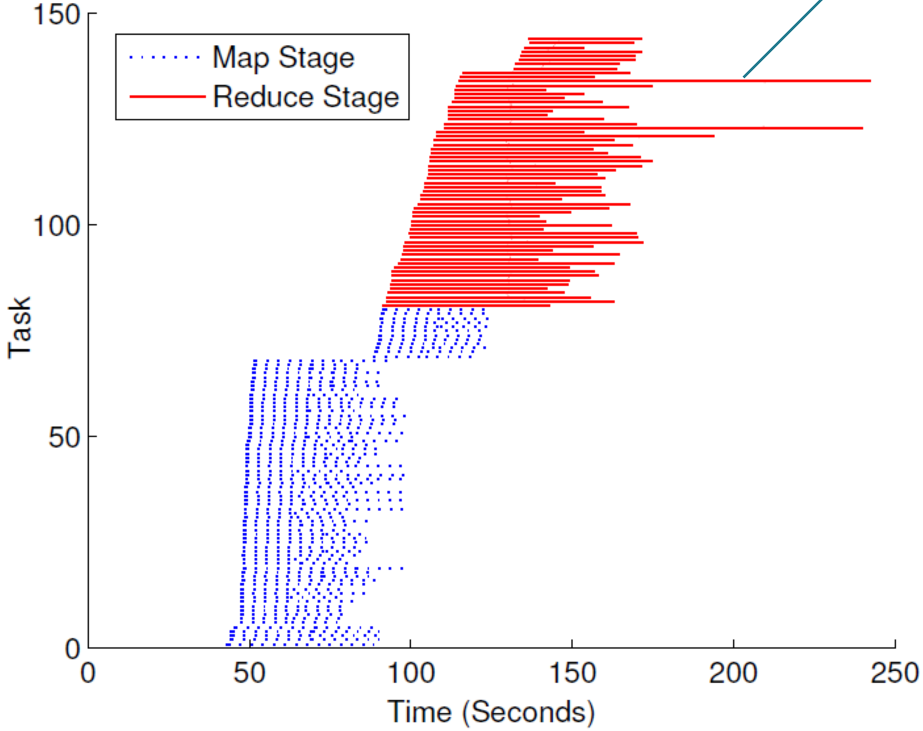

(a) Sort



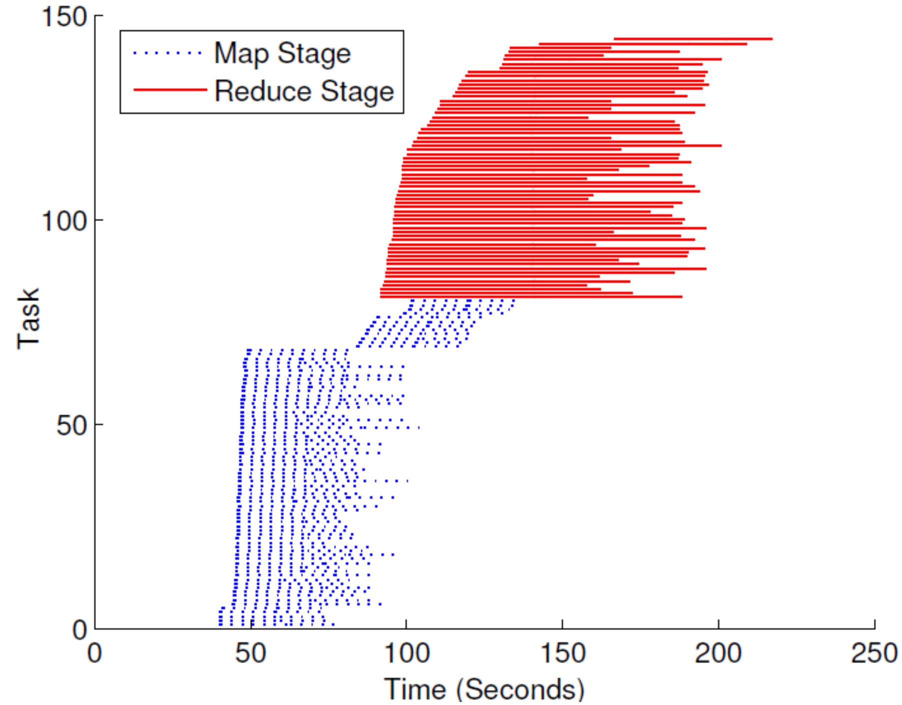

(b) InvertedIndex

# Task execution timeline

The straggling tasks prolong the job completion

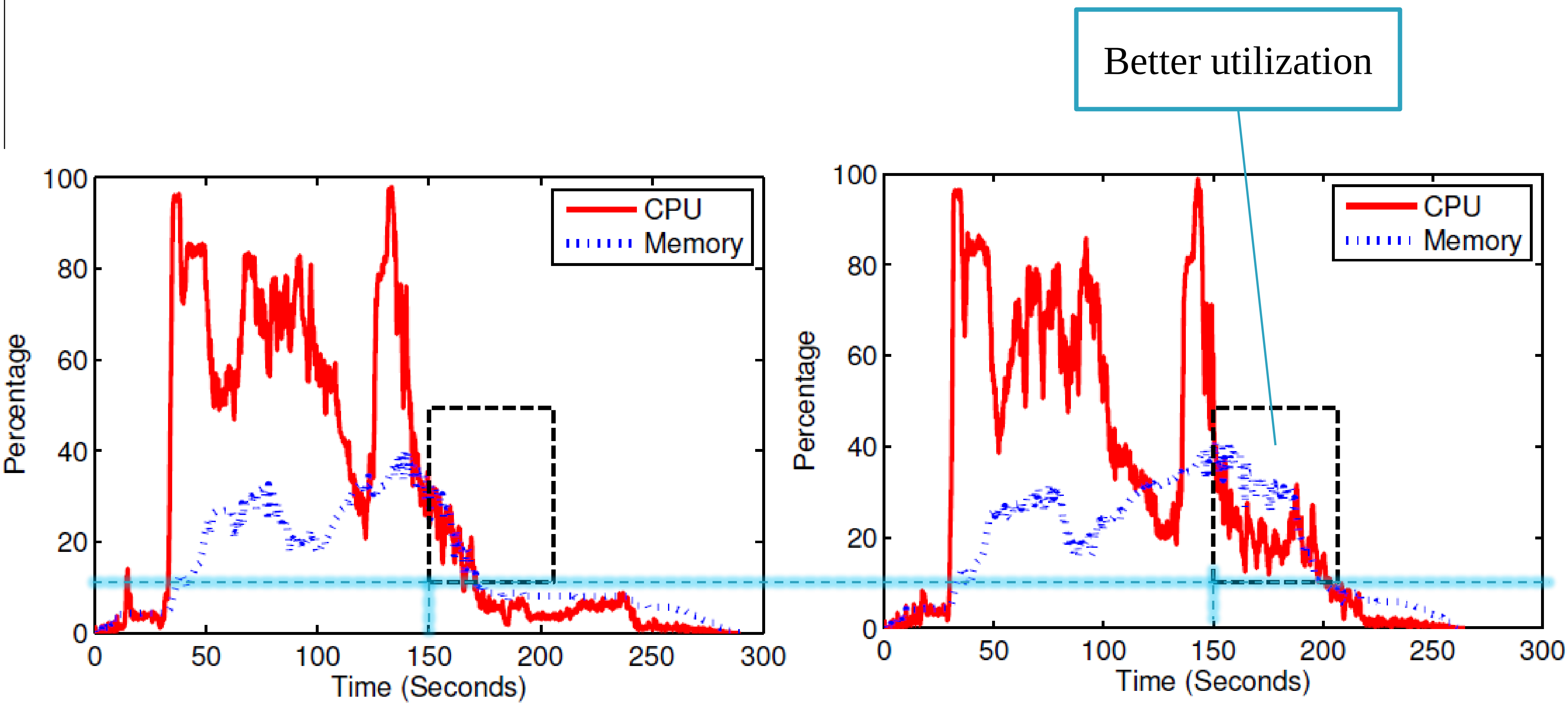



(a) Sorting 10G with Native




(b) Sorting 10G with DREAMS

# Resource utilization




(a) Sorting 10G with Native (b) Sorting 10G with DREAMS

# Conclusion

# Conclusion

- We present DREAMS, a framework that mitigates the data skew for MapReduce by adjusting the container size at run-time

## Our contributions

- We develop an partition size prediction model
  - Perform at run-time
  - The error rate is less than 8.2%
- We design a reduce task performance model
  - The worst error rate is 19.57%
- We demonstrate the benefits of leveraging resource-awareness for data skew mitigation
  - Eliminate the overhead of rebalancing the load
  - Improve the job running time by up to 20.3%

# Thank you ☺

Questions?