Not All Resources are Visible: Exploiting Fragmented Shadow Resources in Shared-State Scheduler Architecture

Xinkai Wang, Hao He, Yuancheng Li, Chao Li, Xiaofeng Hou,
Jing Wang, Quan Chen, Jingwen Leng, Minyi Guo, Leibo Wang

SoCC 2023, Santa Cruz, California

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

ACM Symposium on Cloud Computing

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

CONTENTS

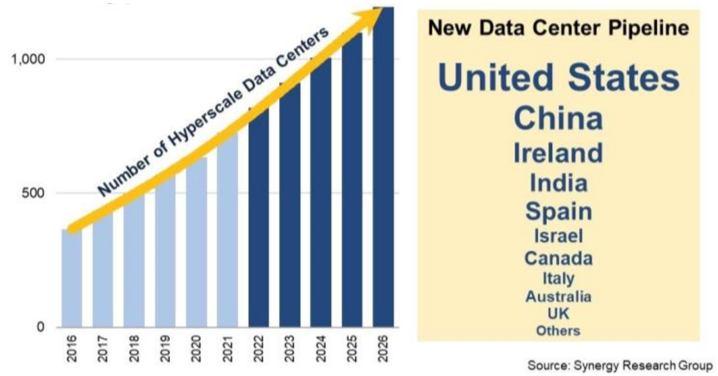
上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

PART ONE

Introduction

Increasing Scale of Datacenters

The scales of lower-level clusters and upper-level requests are increasing greatly!



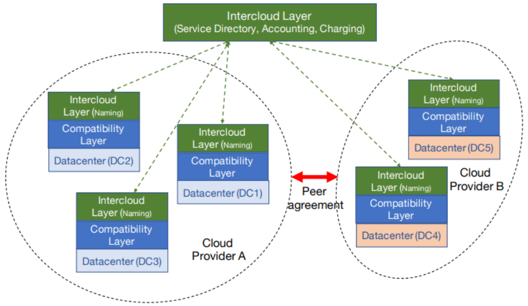
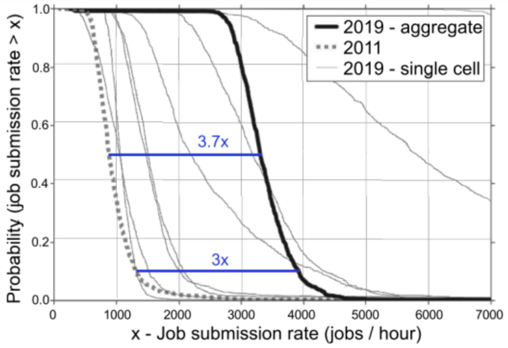


Figure 1: Possible Sky computing architecture.



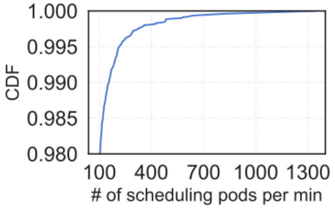


Figure 7. Distribution of the number of pods to be scheduled in each minute.

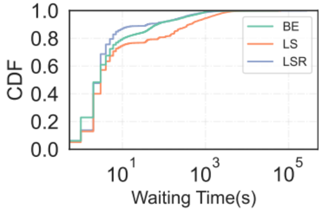


Figure 8. Distribution of waiting time for pods with different SLO types.

The number of hyperscale datacenters are increasing, and the domain of clouds is becoming bigger.

Greater Scheduling Domain

The concurrent job submission rate is increasing, and the scheduling delay are harmful to applications.

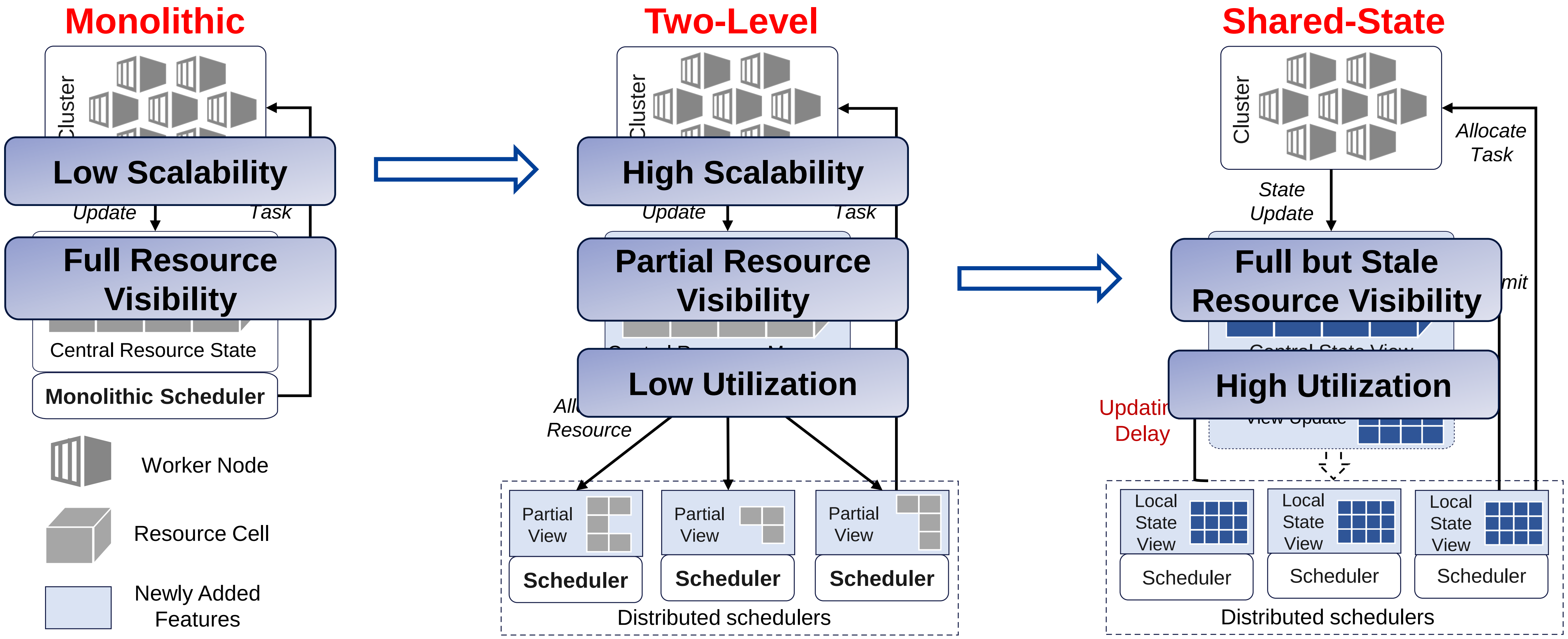
Greater Scheduling Entities

Hyperscale datacenters call for better scheduling capabilities to meet the requirements of request parallelism on heterogeneous clusters.

1. Stoica I, Shenker S. From cloud computing to sky computing[C]//HotOS. 2021: 26-32.
2. Tirmazi M, Barker A, Deng N, et al. Borg: the next generation[C]//EuroSys. 2020: 1-14.
3. Lu C, Xu H, Ye K, et al. Understanding and Optimizing Workloads for Unified Resource Management in Large Cloud Platforms[C]//EuroSys. 2023: 416-432.

Evolution of Large-scale Schedulers

The large-scale scheduler architecture are evolving due to the increasing demands.

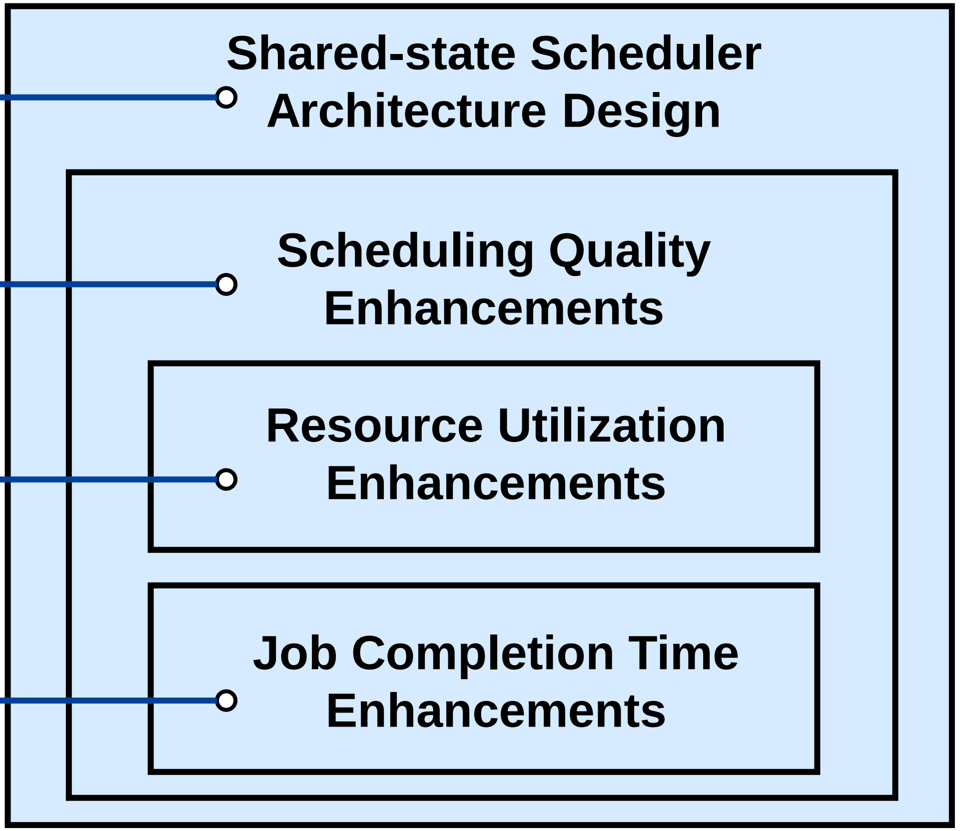


Shared-state scheduler is becoming the popular architecture in datacenters.

Works about Shared-state Scheduler

Shared-state scheduler has been studied widely in both industry and academia.

- **Structure of shared-state scheduler**
  [Omega @ Google, ParSync @ Alibaba, …]
- **Optimize the job wait time quality**
  [Sparrow @ UCB, Tarcil @ Stanford, …]
- **Better estimate resource allocation**
  [Borg @ Google, Apollo @ Microsoft, …]
- **Higher throughput and lower runtime**
  [Mercury @ Microsoft, Hawk @ EPFL, …]



Our work aims at an inherent shortcoming of shared-state scheduler architecture, resource invisibility, to enhance current structure design.

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

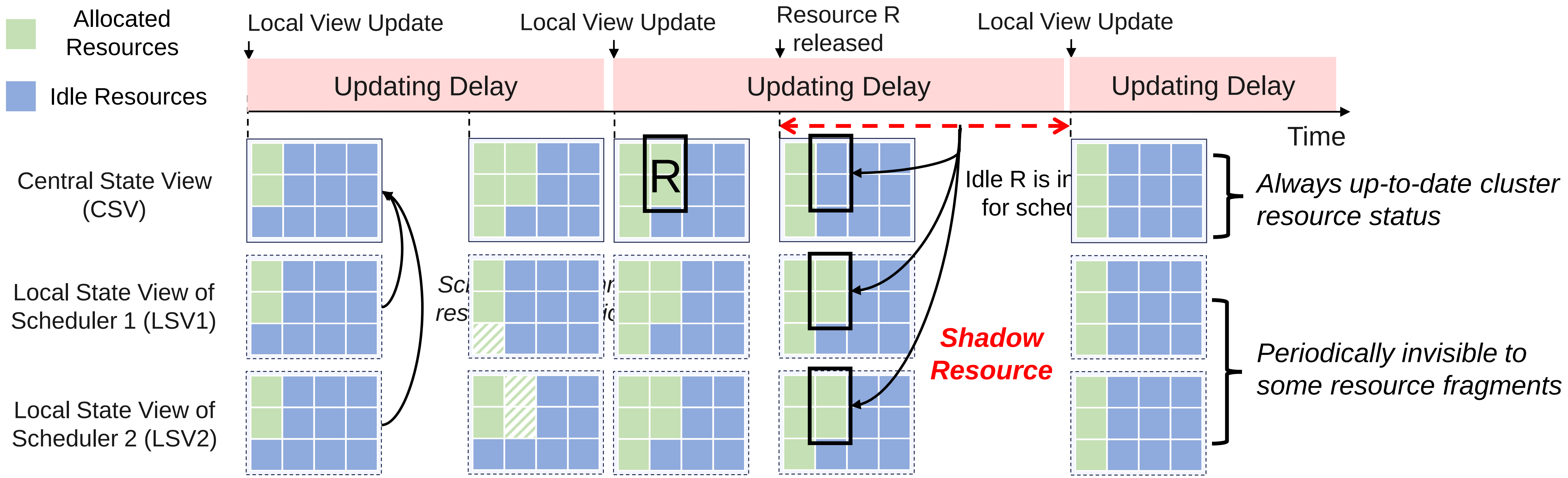
PART TWO

Motivation

SJTU

Introduction to Shadow Resources

The resource states of distributed schedulers are stale within updating delays!



Shadow resources are those that are not visible to the distributed schedulers in their resource views when they can actually be used for allocation.

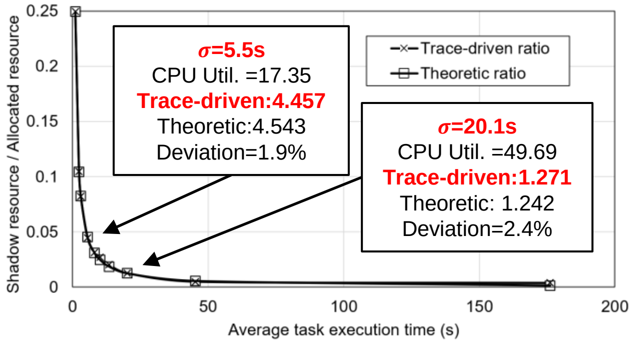
Three Observations about Shadow Resources

Shadow resources are considerable and precious, but hard to exploit them.

- **Theoretical quantitative analysis of shadow resources**
  - *Proportional to the amount of allocated resource in the cluster.*
  - *Inversely proportional to the average execution time of all tasks.*
  - *Roughly account for **3% ~ 12.5%** resources in the cluster!*

$$E(X) = \frac{d_u \times r_{run}}{2\sigma}$$

- **More severe with the advance of lightweight cloud-native tasks**
  - *Microservice and serverless have shorter execution time.*
  - *We validate the trend with industrial trace-driven experiments.*



- **Two challenges hinders the utilization of shadow resources**
  - *How to mine and manage shadow resources agilely and efficiently?*
  - *How to allocate and utilize shadow resources flexibly and transparently?*

We need to enhance the limited resource visibility of current shared-state design!

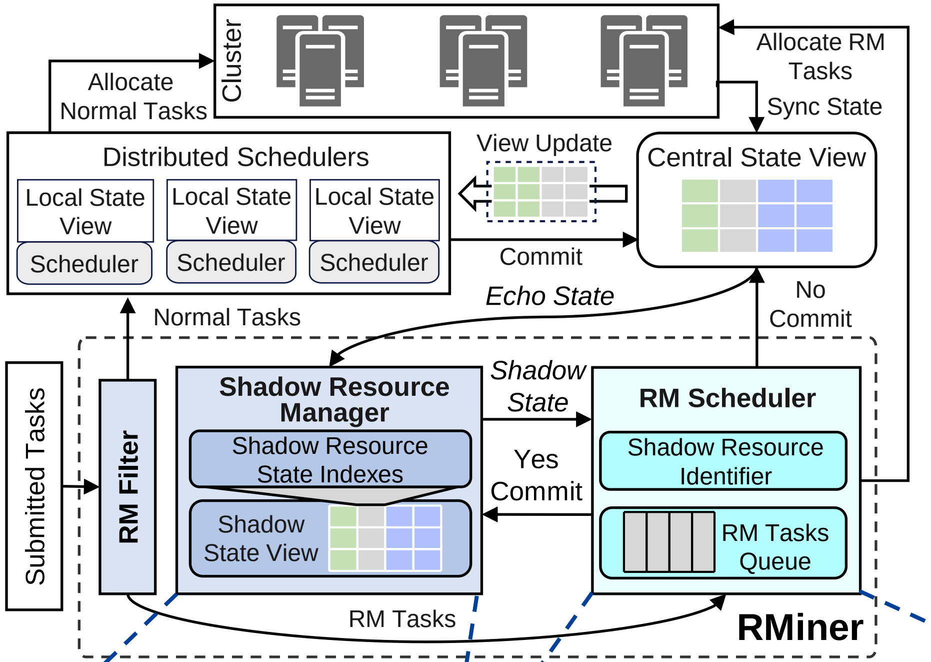
上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

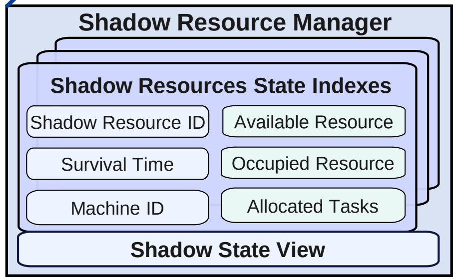
PART THREE

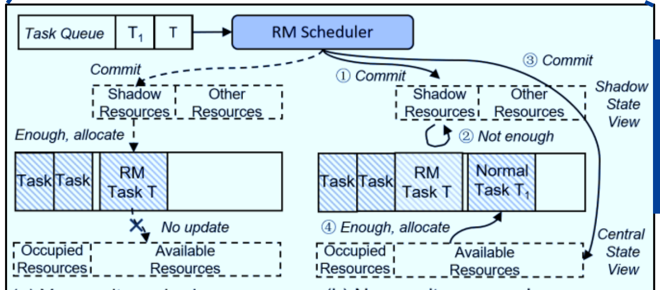
Design of RMiner

Overview of RMiner

RMiner pursues a high-performance and full-visibility scheduler system.







(a) Y-commit on shadow resource (b) N-commit on normal resource

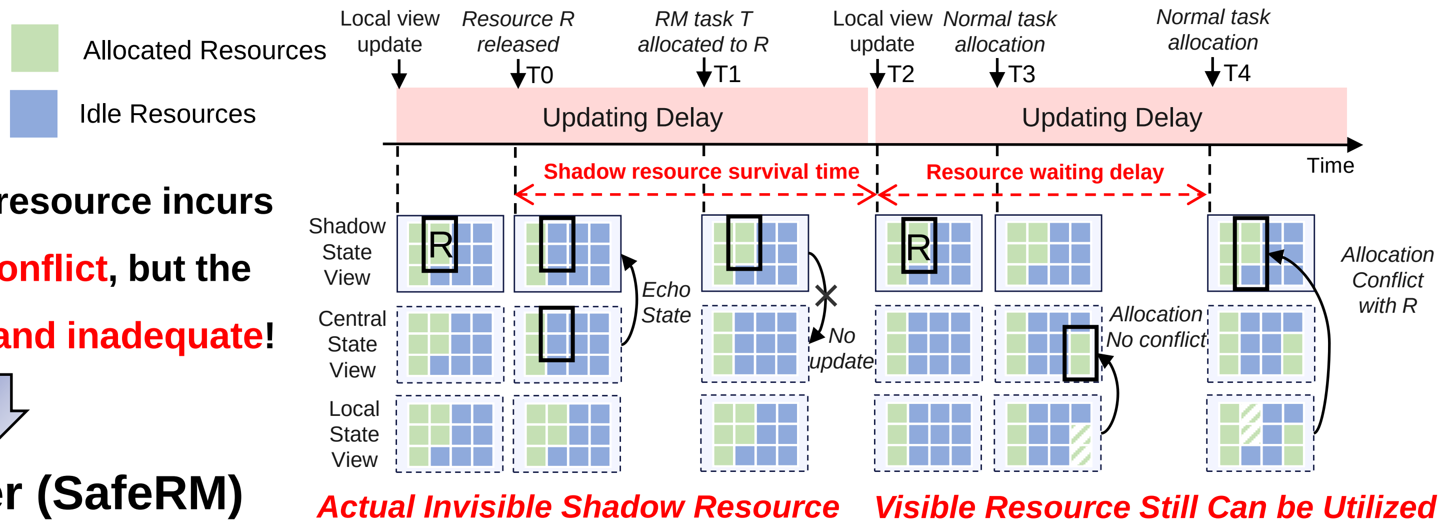
- **RMiner is built upon current shared-state schedulers.**
- **Shadow Resource Manager detects and manages shadow resources with a newly-designed index.**
- **RM Scheduler assigns shadow resources to proper tasks.**
- **RM Filter selects tasks suitable for shadow resources.**
- **We derive Intrusion Avoidance and Balanced Performance design principles for RMiner**

RMiner is composed of straightforward yet effective component designs to work.

More details in the paper!

Different Objectives of RM Scheduler

RMiner have two objectives: resource utilization and scheduling conflicts.



Utilizing shadow resource incurs no scheduling conflict, but the usage is fleeting and inadequate!

⇩

Safe RMiner (SafeRM)

Utilizing shadow resource is aggressive, but possibly incurs conflicts with normal scheduling!

⇩

Smart RMiner (SmartRM)

	SafeRM Mode	SmartRM Mode
Shadow resource time	U_d	$U_d + W_d$
RM filter policy	SJF	LJF
RM scheduling policy	Min Conflict	Max Utilization
Task eviction policy	Migrate - Kill	Kill - Migrate

+ U_d is shadow resource survival time and W_d is resource waiting delay.
+ SJF denotes shortest job first and LJF denotes lowest-priority job first.

- **SafeRM utilizes shadow resources with conflicts as few as possible and lower the priority of resource utilization.**
- **SmartRM pursues maximized utilization via using resources when they are visible and gives proper solutions for conflicts.**

RMiner could adapt to different system design considerations for all.

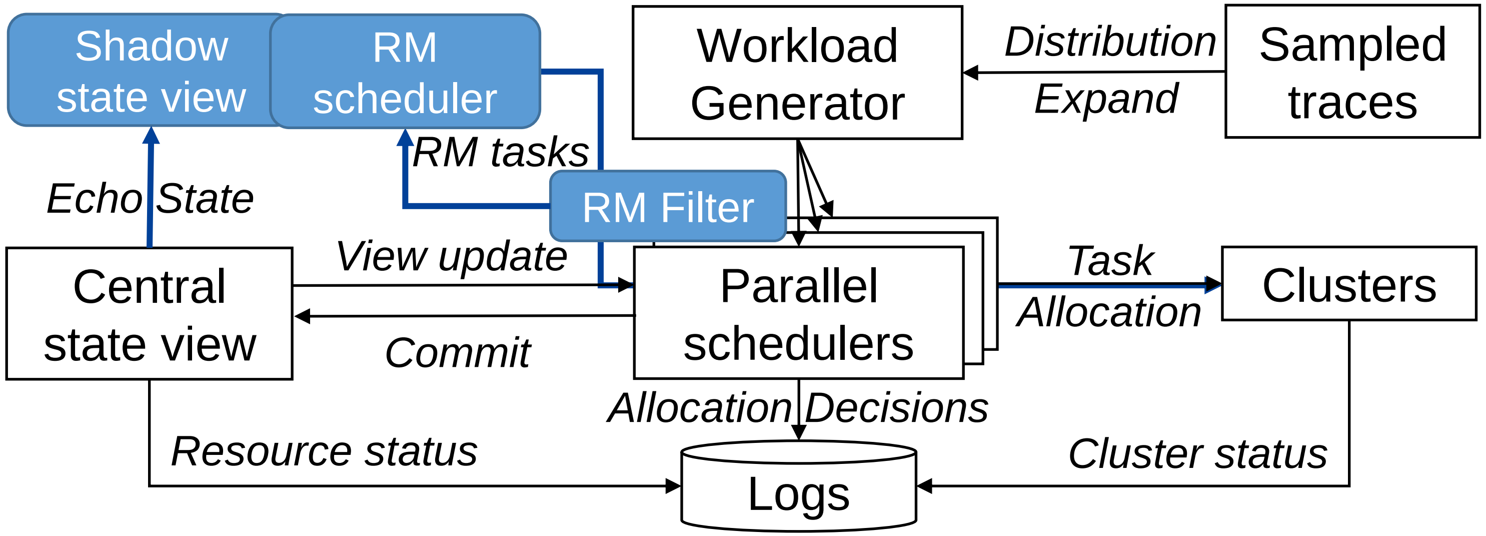
上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

PART FOUR

Evaluations

Evaluation Setups

We thoroughly analyze RMiner on the industrial simulator driven by cluster traces.



Traces	Alibaba Trace	Google Trace
Average task execution time	Sampled (4.94)	5
Average task resource demand	Sampled (1.03/64)	Sampled (0.01)
Average size of jobs	Sampled (12)	10
avgJobInterarrvialTime	1.43 (1x) - 0.7 (2x)	

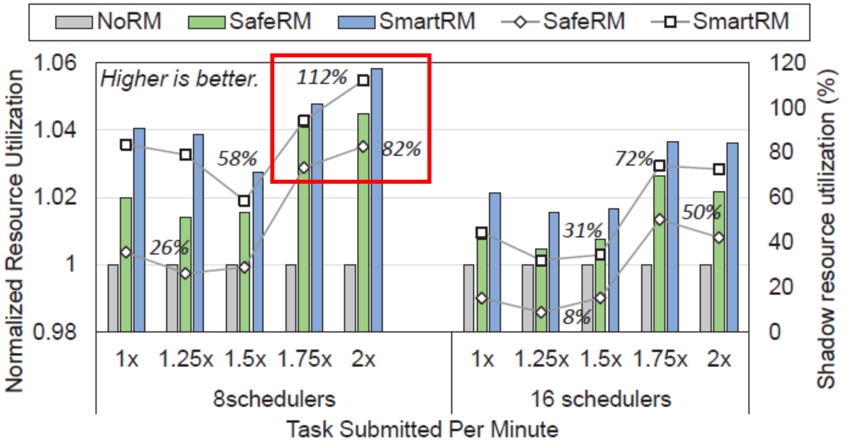
- **We modify Google cluster simulator[1] to integrate shadow resource management and scheduling functionalities.**
- **We mimic a 1500-nodes cluster with 64 CPUs and 16 memory slots.**

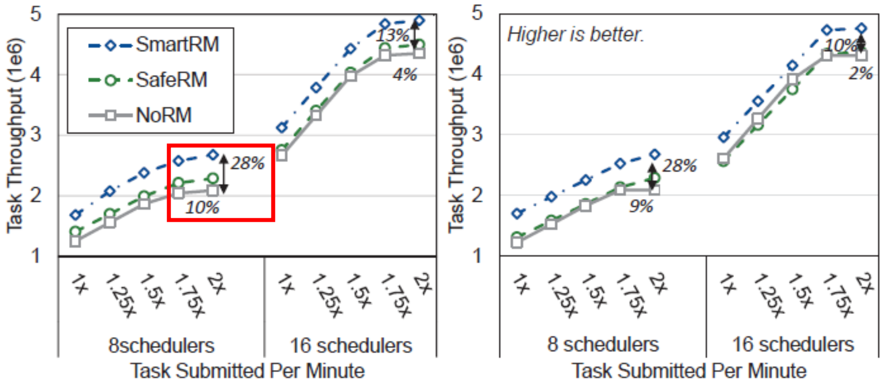
- **We adopt two independent industrial traces to drive the simulation[2][3].**
- **We generate an input stream containing 1 million jobs based on trace patterns.**
- **We compare two RMiners with typical shared-state scheduler architecture.**

1. 2014. Google cluster scheduler simulator. https://github.com/google/cluster-scheduler-simulator.
2. 2022. Alibaba Cluster Trace Program. https://github.com/alibaba/clusterdata.
3. 2019. ClusterData 2019. https://github.com/google/cluster-data/blob/master/ClusterData2019.md.

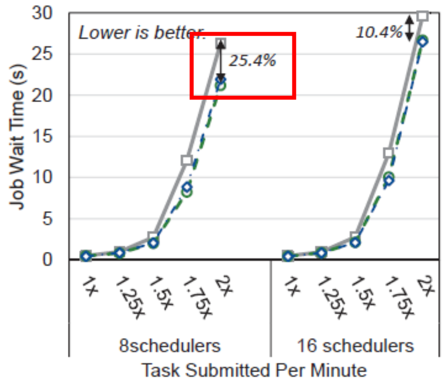
Improvements of RMiner

RMiner improves at resource utilization, throughput, and job wait time metrics.





(a) Results on Alibaba's Trace (b) Results on Google's Trace



(a) Results on Alibaba's Trace

- **SafeRM improves cluster CPU utilization by 1.5%-4%, SmartRM improves by 1.6%-5.8%.**
- **SafeRM utilizes 26%-82% shadow resources, SmartRM utilizes 58%-112% of them.**

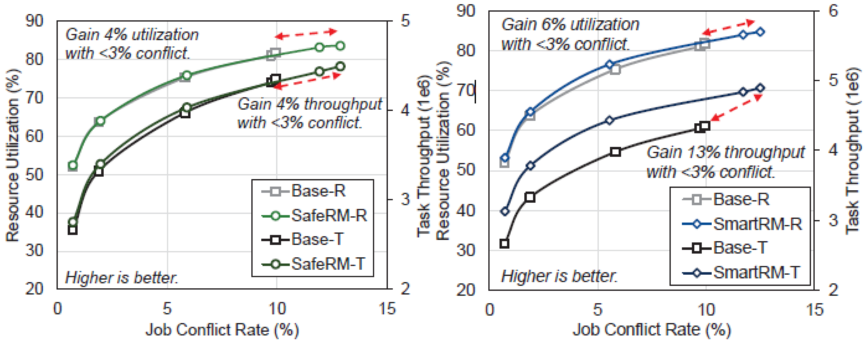
- **SafeRM achieves 4%-10% throughput improvements, SmartRM improves 13%-28%.**
- **RMiner performs better under higher workloads and less parallel schedulers.**

- **RMiner improves the waiting time between the job submitted and being scheduled by 25.4%.**

RMiner achieves multi-dimensional performance improvements via flexible utilization of shadow resources within shared-state architecture.

Detailed Analysis of RMiner

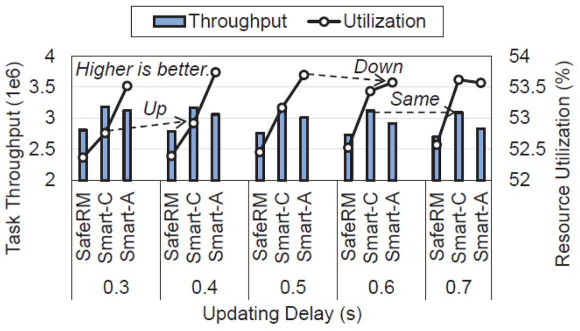
Overheads

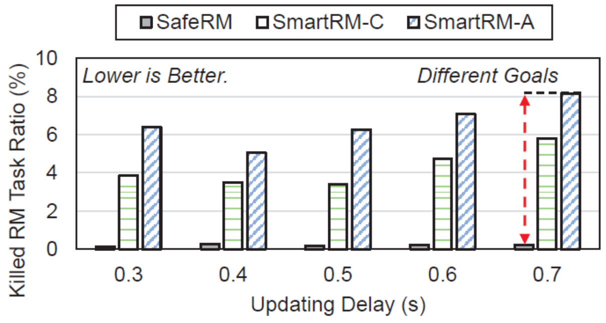


(a) 8 schedulers (b) 16 schedulers

- On average, SafeRM causes 0.5% more conflicts and SmartRM causes 0.73% more conflicts.
- SmartRM causes 3% conflict increase in the worst case for 6% utilization and 13% throughput.
- More overhead analysis in the paper.

Optimization Modes





- Different optimization modes of RMiner outperforms in respectively targeted metrics under various scenarios
- Performance of RMiner is affected by updating delay due to different design objectives and normal parallel schedulers.

RMiner achieves improvements with acceptable costs, and it can be flexibly configured for different design goals.

Conclusions

- We discover the **invisible resource opportunities** in shared-state scheduler architecture and analyze them comprehensively.
- We introduce RMiner, a **novel extension** over current architecture to mine and exploit the hidden shadow resources.
- We thoroughly analyze RMiner over an industrial cluster simulator to show the **pros and cons** of our designs.
- In the future, we plan to **integrate RMiner** into industrial schedulers and further **enhance current** mining and scheduling designs.

Thank You! Q & A

Not All Resources are Visible: Exploiting Fragmented Shadow Resources in Shared-State Scheduler Architecture

Discussion: Xinkai Wang, unbreakablewxk@sjtu.edu.cn