SHANGHAI JIAO TONG UNIVERSITY

Not All Resources are Visible

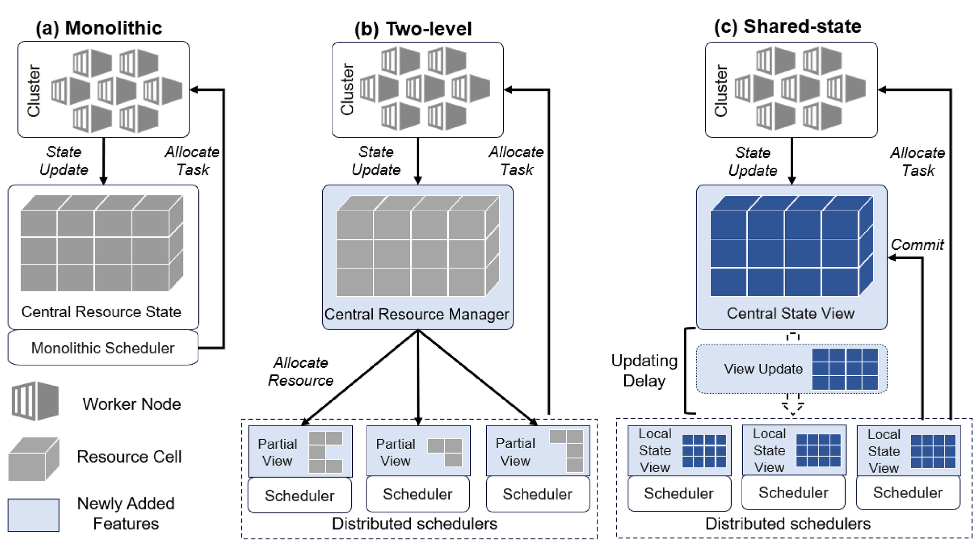
ACM Symposium on Cloud Computing

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[1]

Department of Computer Science and Engineering, Shanghai Jiao Tong University [1]Huawei Cloud

Evolution of Schedulers



Monolithic Scheduler

The only scheduler with the least scalability.

Two-Level Scheduler

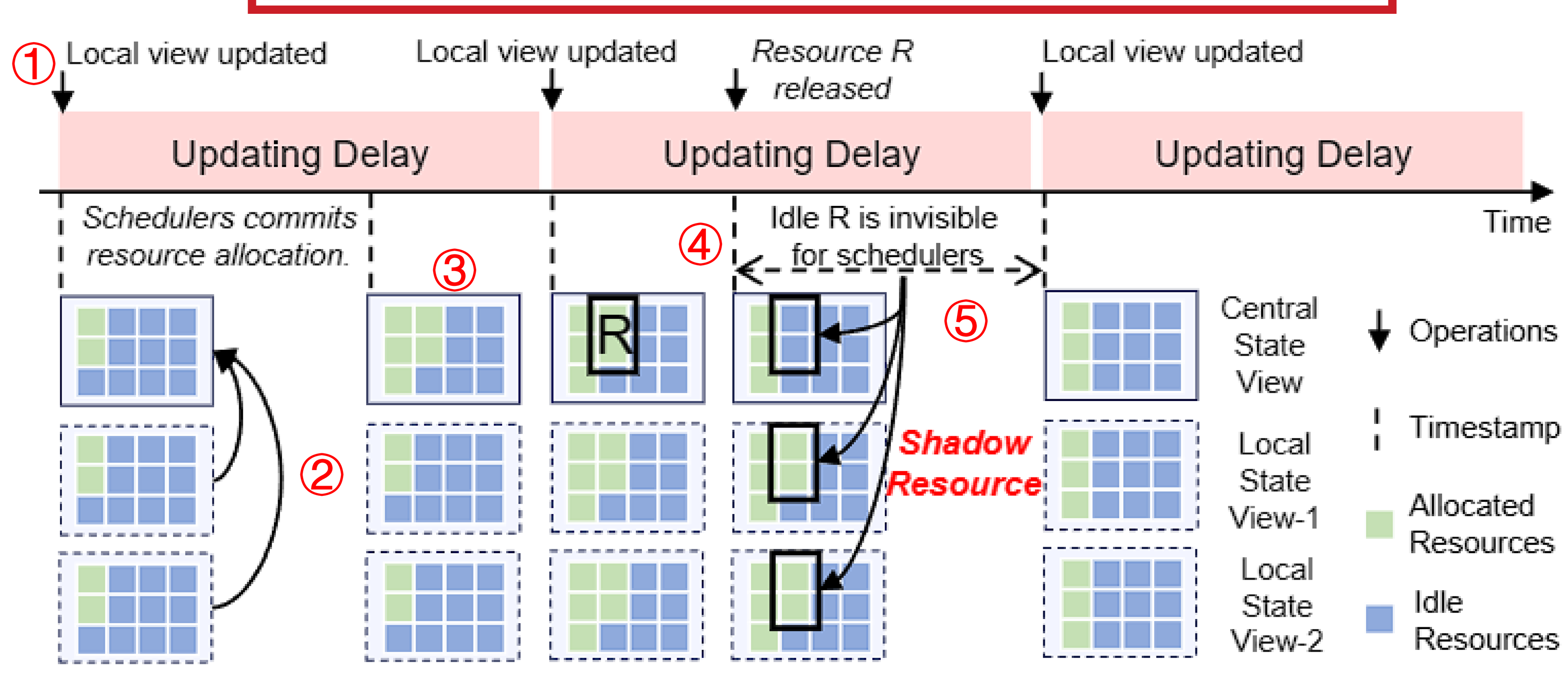
Parallel partial schedulers with low utilization.

Shared-State Scheduler

Parallel full schedulers with schedule conflicts.

Shared-state scheduler architecture has become the popular scheduling systems for large-scale clusters due to high scalability and utilization. It provides all parallel schedulers with a global view of all the resources. The central state view maintains up-to-date cluster resource status and periodically updates the local state views owned by each distributed schedulers with fixed updating delays.

Introduction of Shadow Resource



① At the start of each updating delay [5], the Central State View (CSV) updates the local resource state of each Local State View (LSV) owned by distributed schedulers.

② The distributed schedulers commit resource allocation decisions to CSV for task scheduling.

③ Within the updating delay, the latest cluster status is known by only the CSV and LSVs owned by each scheduler fall behind the actual cluster.

④ Similarly, when certain resources **R** are released, the CSV updates itself immediately while distributed schedulers are still relying on stale local state views.

⑤ Before the next local view update, each scheduler would treat idle resource **R** as allocated and cannot schedule new tasks to **R**, leading to a great waste of resources.

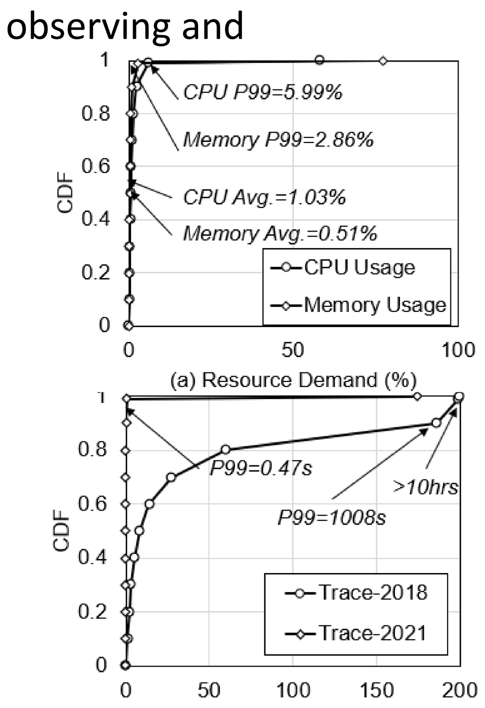
***Shadow resources** are those invisible to the distributed schedulers in their resource view when they can be used for task allocation.*

Importance of Shadow Resource

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

- **X**: Amount of shadow resources; d_u: Duration of updating delay
- r_{run}: Total allocated resources; σ: Average running time of tasks

We both theoretically model and experimentally analyze the quantitative amount of shadow resources in the cluster. Based on industrial traces [2][3], shadow resource account for about *3%~12.5%* in the cluster. Further, we validate the theoretical model via observing and recording the realistic shared-state scheduling process.



Further, shadow resources are more severe nowadays due to the spatial and temporal granularity trends of more lightweight datacenter tasks. It is important but challenging to properly utilize them.

- The **resource demand** of tasks is lower than traditional monolithic applications, leading to more fine-grained resource fragments.
- The **execution time** of tasks is decreasing due to the emergence of cloud-native technology, leading to more fleeting resource fragments.

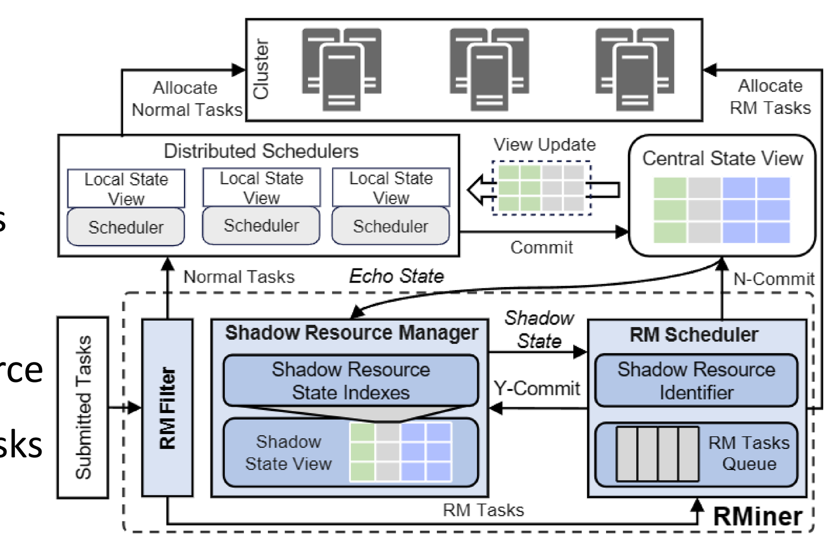
***Shadow resources** is precious and considerable for shared-state scheduling but still requires agile management and transparent utilization.*

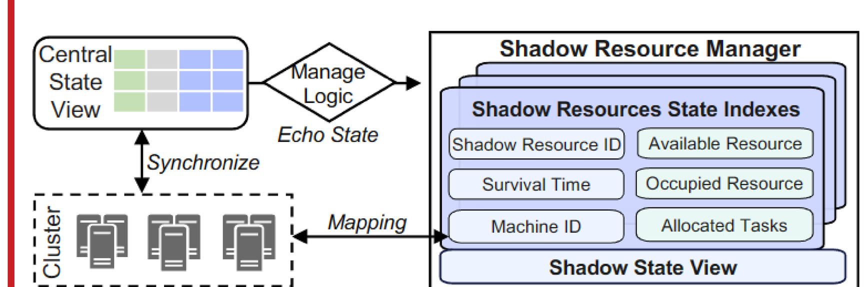
Contact: Xinkai Wang, unbreakablewxk@sjtu.edu.cn

Design of RMiner

Resource Miner (**RMiner**) is composed of three cooperated components:

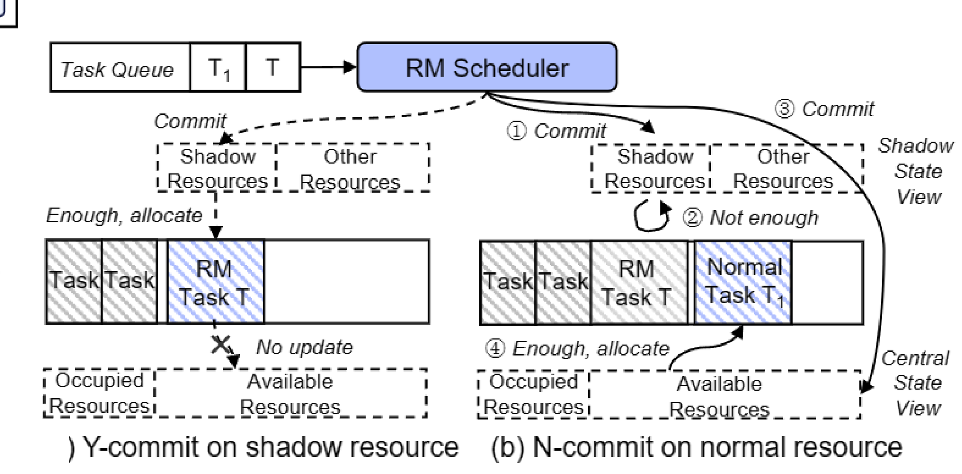
1. **Shadow Resource Manager** detects and manages up-to-date shadow resource state
2. **RM Filter** selects tasks suitable for shadow resource
3. **RM Scheduler** assign shadow resources to RM tasks

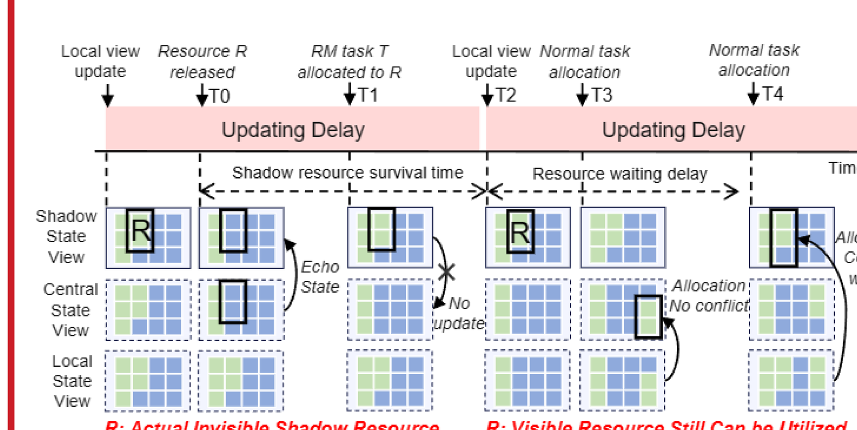




Shadow Resource Manager maintains a newly-designed data structure, shadow resource state index, to manage shadow state view.

RM Scheduler interacts with the shadow state view and central state view in different situations and has two task allocation paths to properly allocate resource to RM tasks.



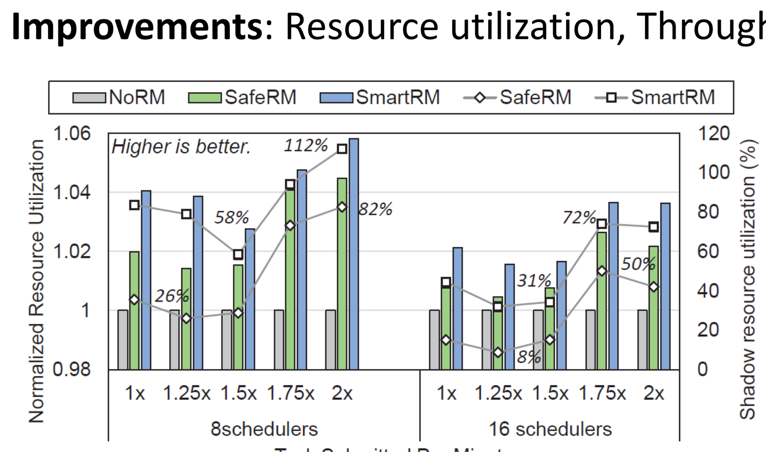


Resource Waiting Delay is the aggressive extension of shadow resources and RMiner has two diverse modes for trade-offs between maximizing resource utilization and minimizing scheduling conflicts.

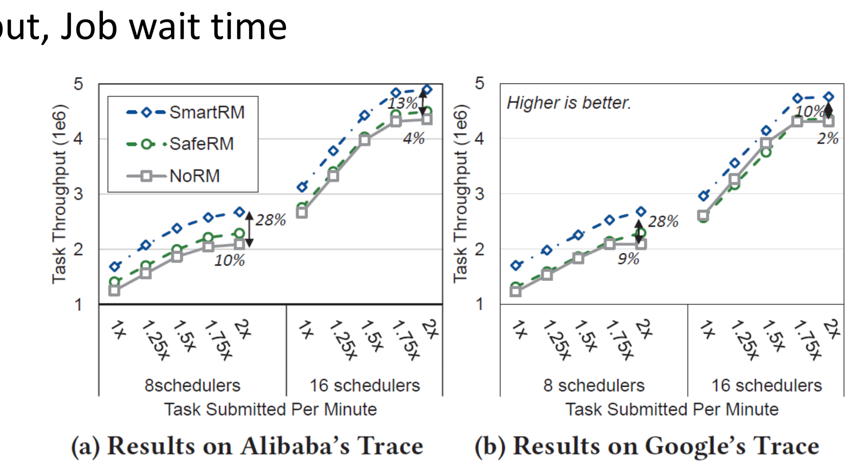
More details and design considerations in the paper!

Evaluation Results

- **Methodology**: We modify the open-source Google cluster simulator [1] and use two industrial traces [2][3] to compare the effectiveness of RMiner with Omega [4].
- **Improvements**: Resource utilization, Throughput, Job wait time



RMiner achieves ~4% utilization improvements via mining 31-112% shadow resources.



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

RMiner achieves ~13% throughput improvements

- **Overheads**: Job conflicts, Scheduling Overhead
- **More detailed results and analysis in the paper.**

Takeaways

1. There are invisible fragment resource opportunities in shared-state scheduling architecture.
2. RMiner mines and utilizes such shadow resources to enhance the resource visibilities.
3. RMiner improves cluster performance at many aspects with only minor conflict overhead.
4. In the future, we plan to integrate RMiner into realistic shared-state scheduler systems.

References

[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.

[4] Schwarzkopf, M., Konwinski, A., Abd-El-Malek, M., & Wilkes, J. (2013, April). Omega: flexible, scalable schedulers for large compute clusters. In Proceedings of the 8th ACM European Conference on Computer Systems (pp. 351-364).

[5] Feng, Y., Liu, Z., Zhao, Y., Jin, T., Wu, Y., Zhang, Y., ... & Guan, T. (2021). Scaling large production clusters with partitioned synchronization. In 2021 USENIX Annual Technical Conference (USENIX ATC 21) (pp. 81-97).