URSA: Precise Capacity Planning and Fair Scheduling based on Low-level Statistics for Public Clouds

Wei Zhang, Ningxin Zheng, Quan Chen, Yong Yang, Zhuo Song, Tao Ma, Jingwen Leng, Minyi Guo

Shanghai Jiao Tong University & Alibaba Cloud

1 Background & Motivation

2 The methodology of URSA

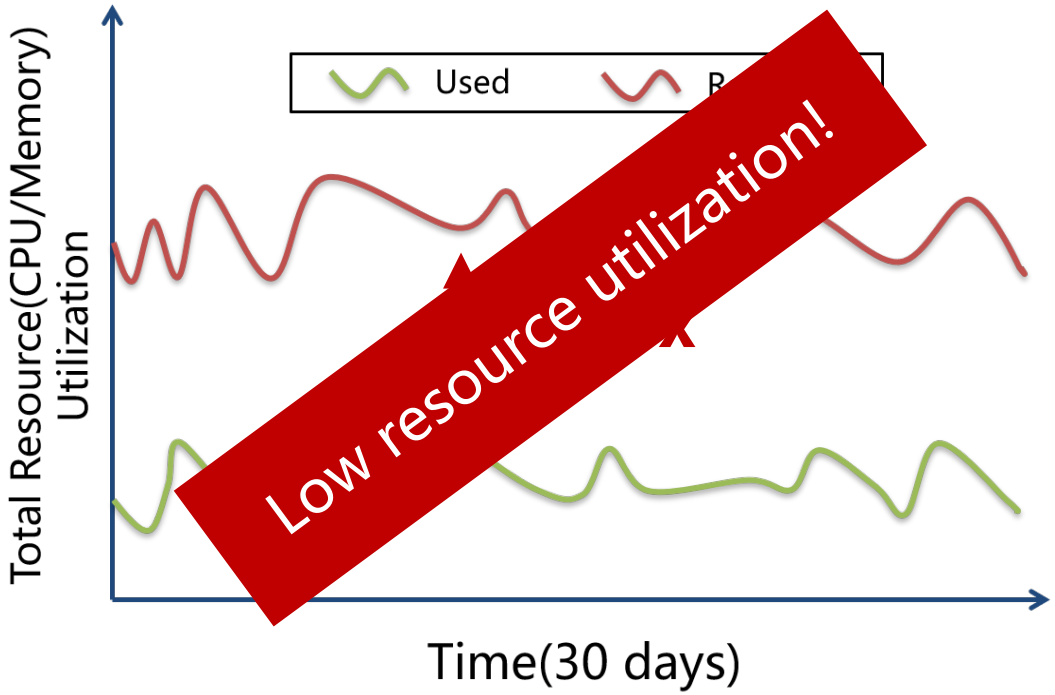
3 Evaluation

4 Conclusion

Problem：Datacenter Underutilization

dbPaaS

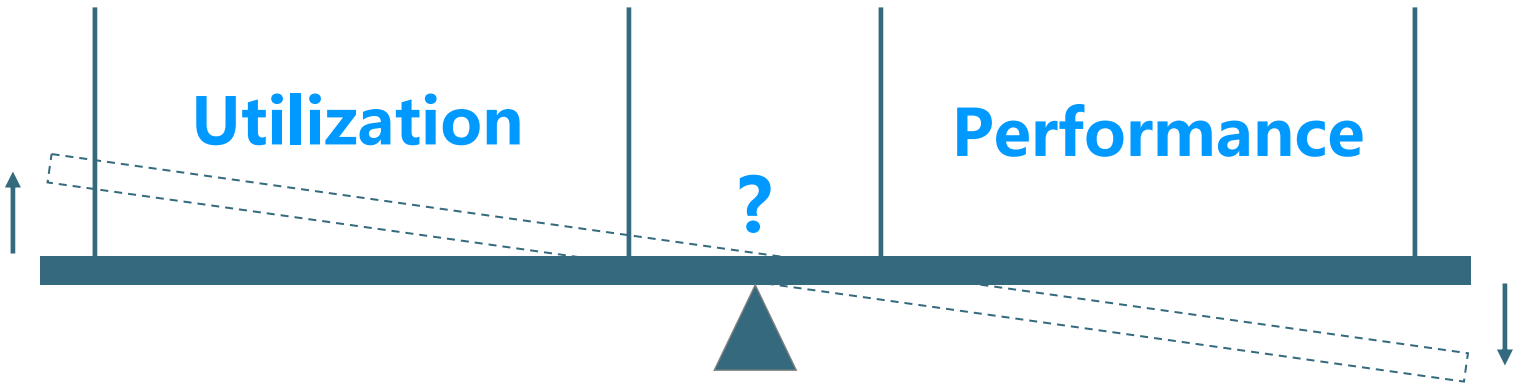
- **The excessive purchase of resources on the cloud**



Reserved vs Used Resources：Twitter: up to 5x CPU & memory overprovisioning

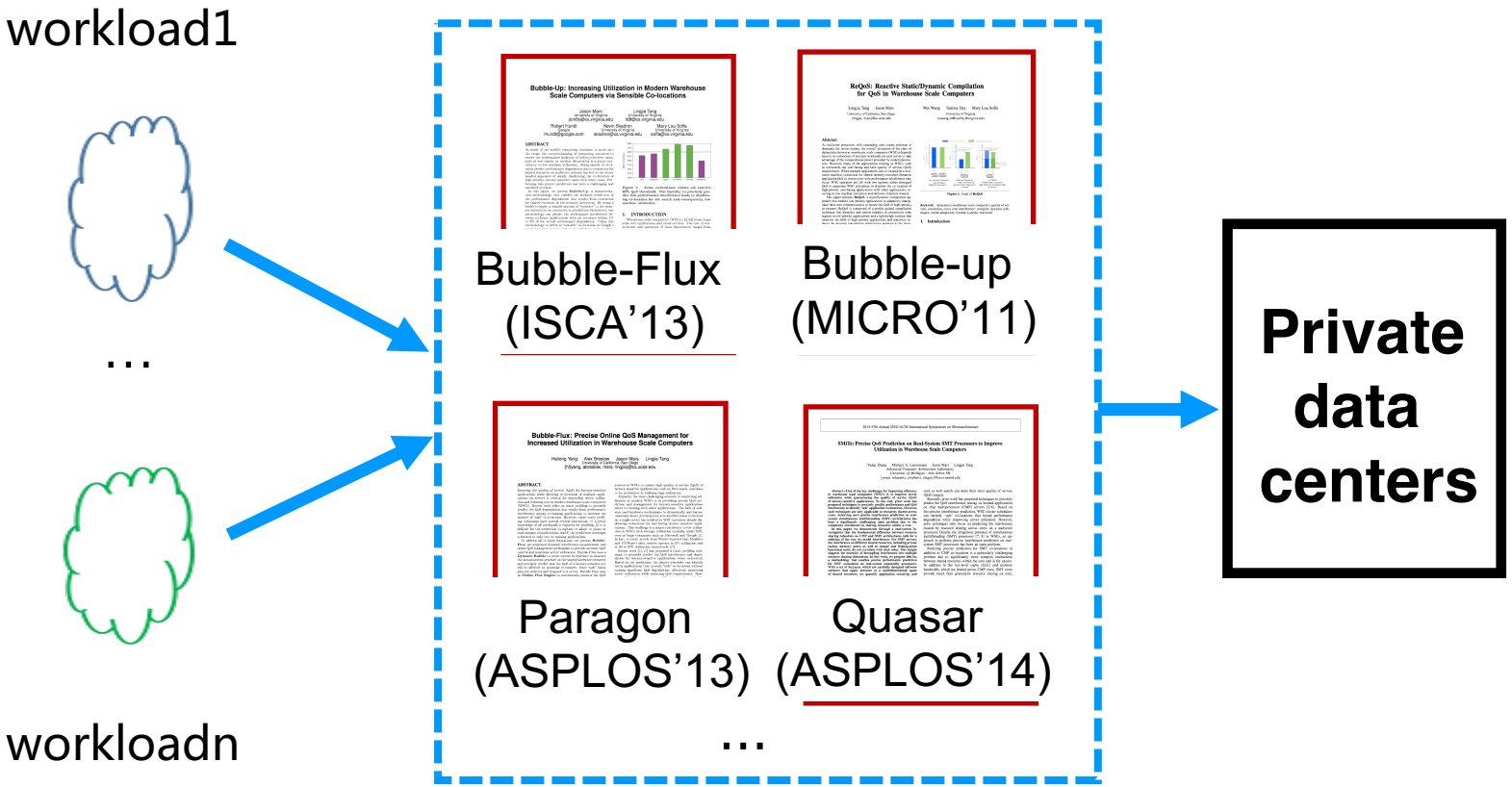
Overprovisioned reservations by users ⟹ *Capacity planning*

Problems in Capacity Planning



Improve utilization while guaranteeing the performance goals of users.

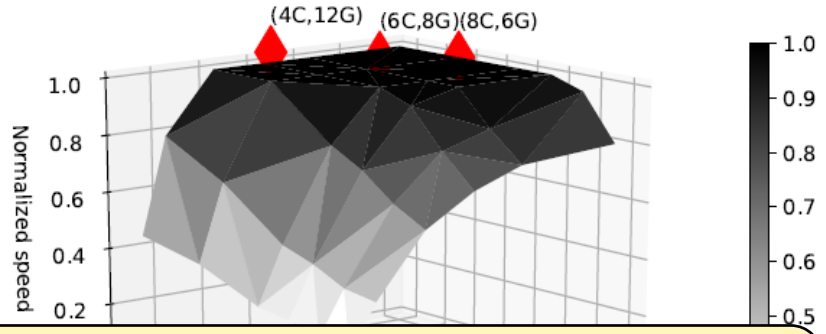
Solutions for Private Datacenter



Problems in dbPaaS public clouds

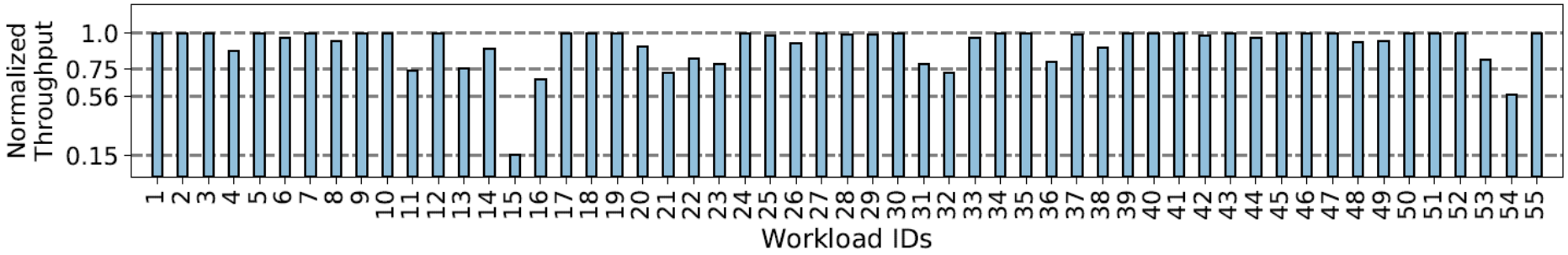
New challenges?

- Poor Resource Utilization
  - Heuristic search will get stuck in local optima
  - Extensive profiling is not applicable due to privacy problem



Prior work is not applicable for Database platform-as-a-service(dbPaaS) in public Clouds!

contention and pressure



1 Background & Motivation

2 The methodology of URSA

3 Evaluation

4 Conclusion

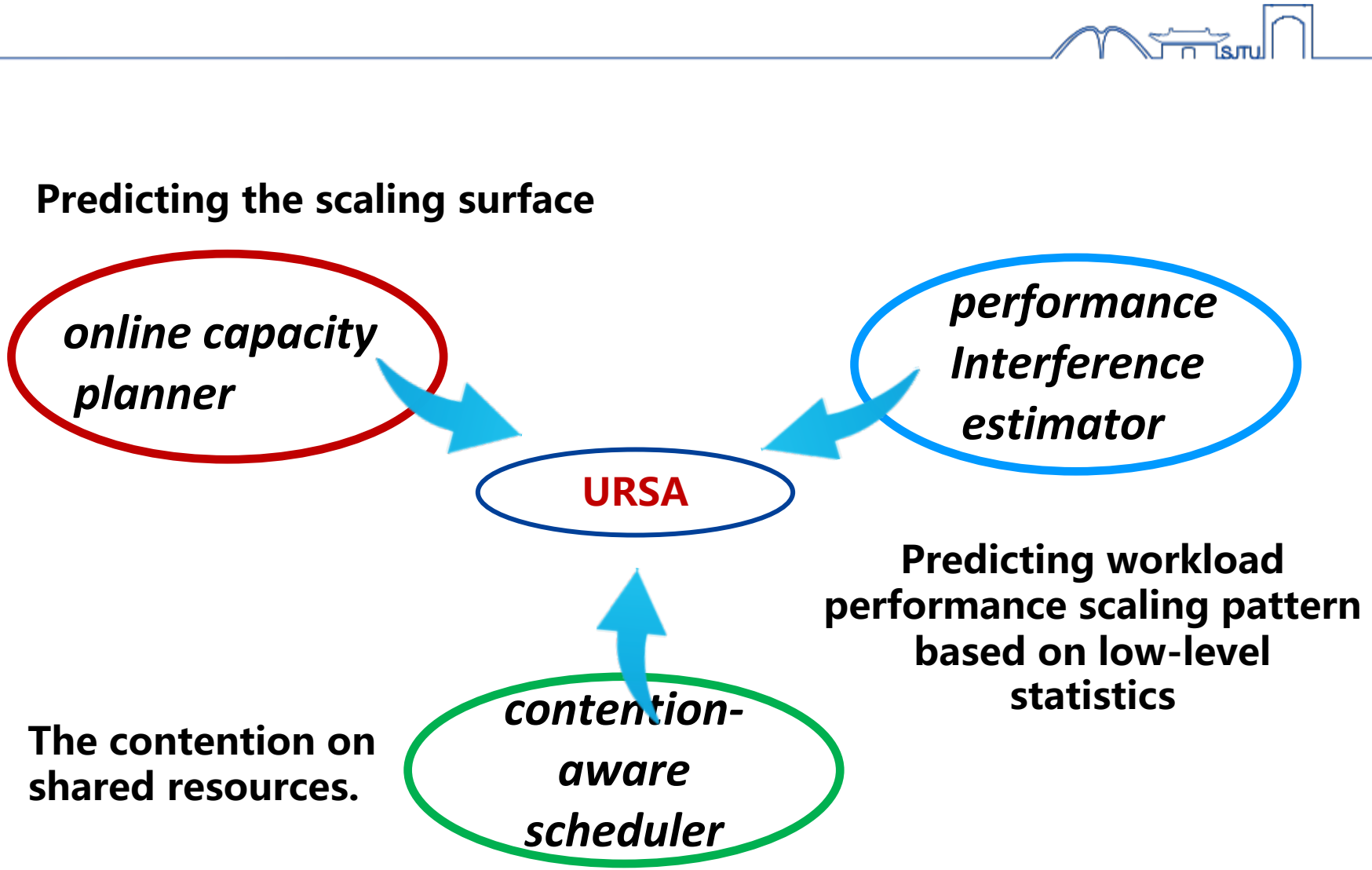
Main Idea of URSA

- **Predicting the scaling surface of the target workload based on the low level statistics and adjusting the resource specification accordingly. (*A online capacity planner*)**

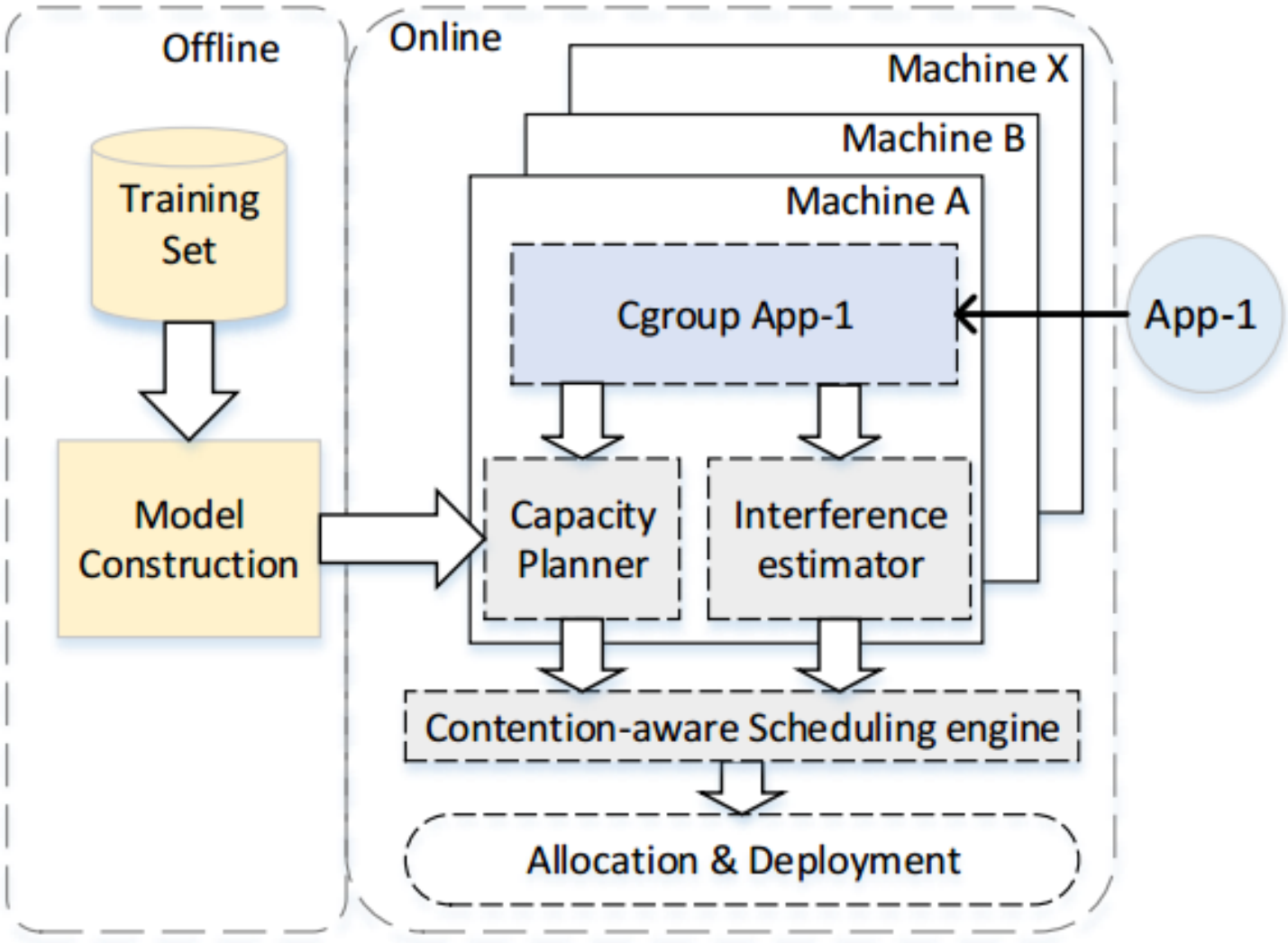
- **Quantifying the interference "pressure" and its "tolerance" to the contention on shared resources using low-level statistics. (*An performance interference estimator*)**

- **Designing a contention-aware scheduling engine at the Cloud level.**

Overview


Predicting the scaling surface
online capacity planner
performance Interference estimator
URSA
Predicting workload performance scaling pattern based on low-level statistics
The contention on shared resources.
contention-aware scheduler

The Design of URSA

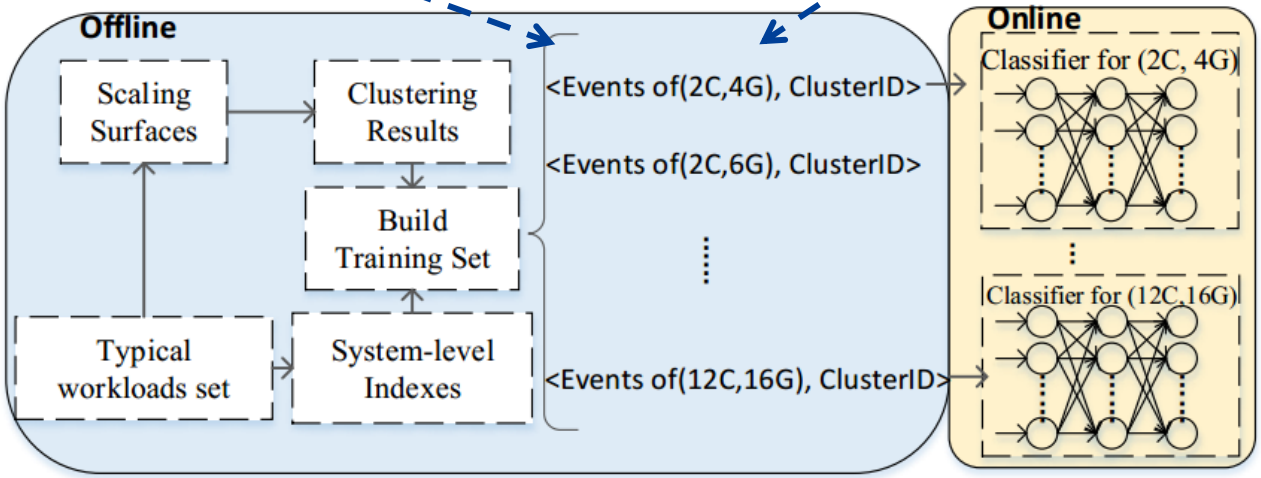


Construct capacity planner

- How to construct the capacity planner.

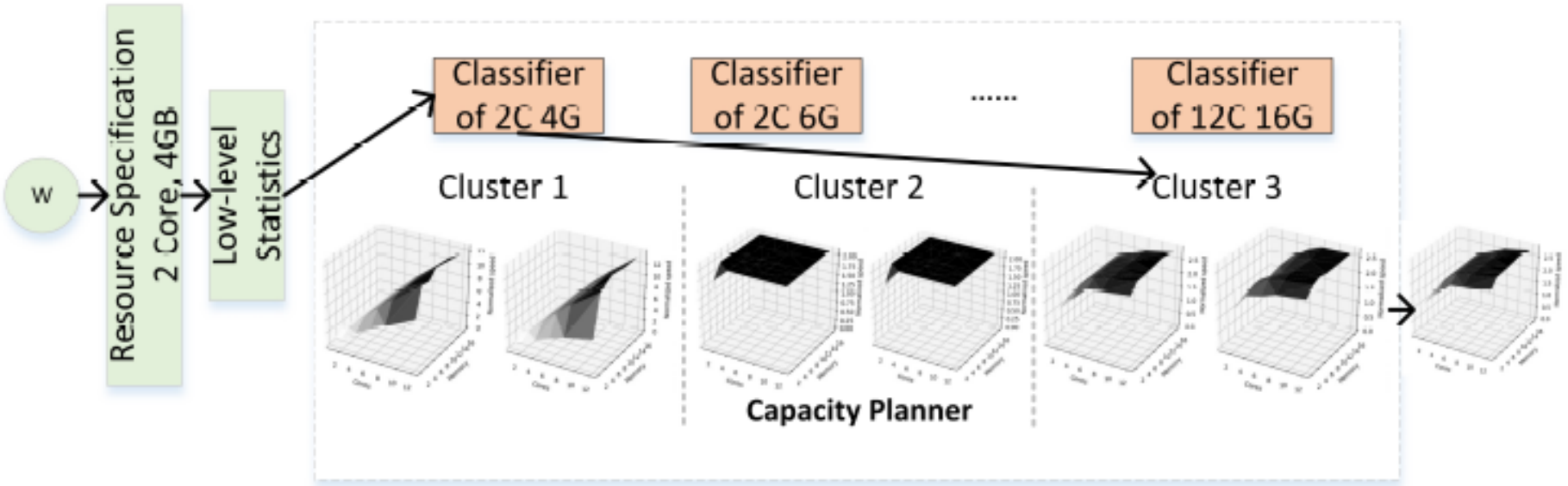
Index	Source	Index	Source
(1) IPC	perf	(9) page-fault	perf
(2) dTLB-store-misses	perf	(10) dTLB-load-misses	perf
(3) cache-misses	perf	(11) cache-references	perf
(4) node-stores	perf	(12) node-loads	perf
(5) io-read-bytes	cgroup	(13) io-write-bytes	cgroup
(6) io-serviced-read	cgroup	(14) io-serviced-write	cgroup
(7) memory usage	cgroup	(15) dirty memory	cgroup
(8) cpu usage	cgroup		

Selected system-level indexes

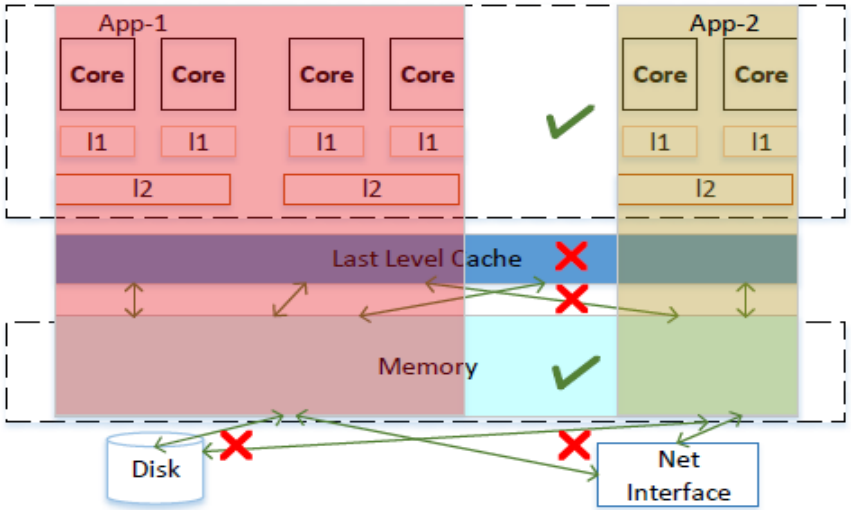


Online capacity planning

- How to perform capacity planning for an online workload.

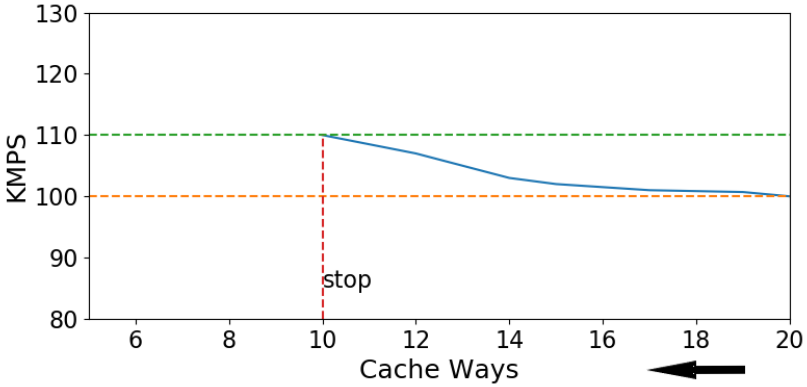


Interference estimator



- Interference due to LLC

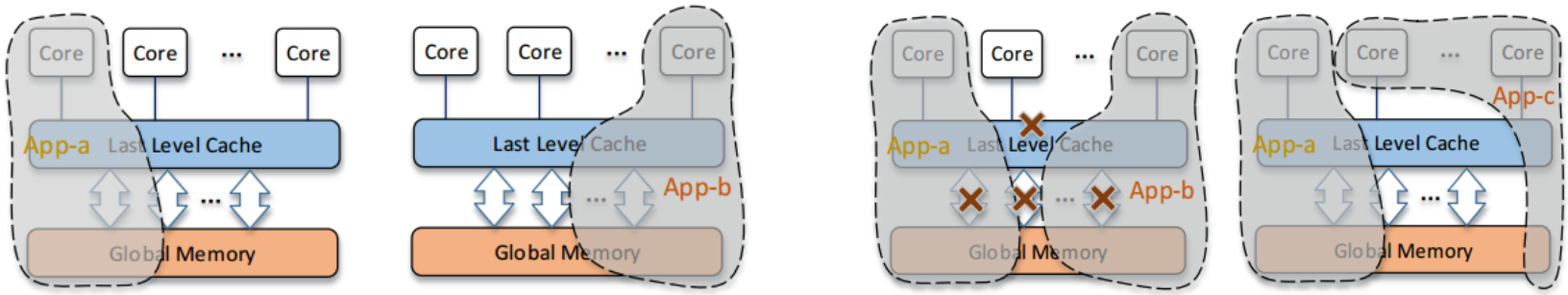
$$kmps = \frac{N_{cache-misses}}{T} \quad (1)$$



- Interference due to Memory Bandwidth

$$Pressure_{mbw} = N_{mbw} \times \frac{Usage_{mbw}}{Phy_{mbw}}$$

Contention-aware Scheduler



Based on the quantified pressures and tolerances of each database workload on all the shared resources, the contention-aware scheduling engine carefully places the workloads for enforcing the performance fairness.

$$SS = CS \times RS \qquad CS = \sum_{r=1}^{N_R} (MaxS_r \times SumP_r \times Factor^{SumP_r})$$

Each node is given a Schedule Score(SS). *CS* quantifies the contention score of the node (smaller is better) and *RS* quantifies the resource score of the node (smaller is better). For a node, *RS* is calculated to be the average percentage of the used CPUs and memory of the node. *CS* is calculated in the upon formula.

1 Background & Motivation

2 The methodology of URSA

3 Evaluation

4 Conclusion

Experimental setup

	Configuration
Hardware	CPU: Intel Xeon(R) Platinum 8163@2.50GHz Cores: 96; L3 shared cache: 32MB DRAM: 256GB; Disk: NVME SSD Network Interface Card (NIC): 25,000Mb/s
Network	25,000Mb/s Ethernet Switch
Software	DBMS: AliSQL 5.6.32 [3] Operating system: Linux with kernel 3.10.0

Benchmarks

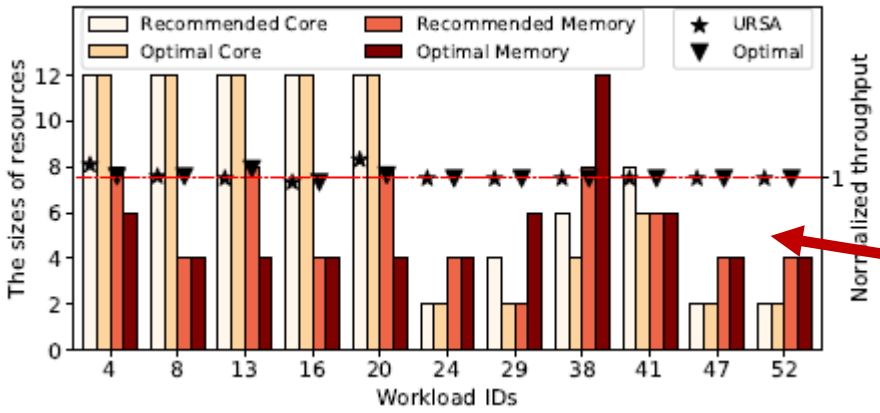
- Generating database workloads using two widely-used workload generators: Sysbench and OLTPBench that includes YCSB , TPC-C, LinkBench and SiBench workloads.
- We adjust the configurations of Sysbench, YCSB, TPC-C, LinkBench, SiBench, and generate 11 variations for each of them. The 55 workloads are randomly divided into a training set containing 44 workloads and a validation set containing 11 workloads.

Evaluation

- **Effectiveness of the Capacity Planning**
  - Scenario 1: Achieving Performance Target.

18/22 is the optimal resource specification

  - Scenario 1: Cutting Down Rent Cost.

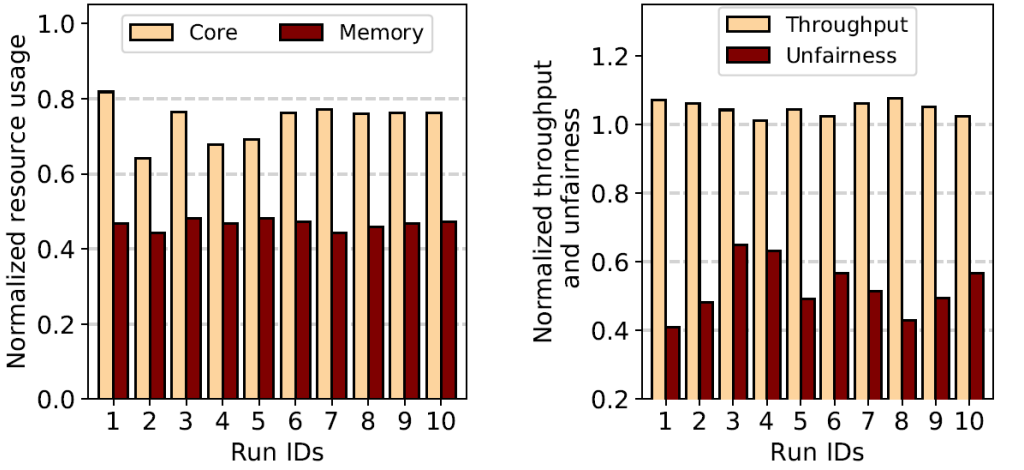


5/11 is the optimal resource specification

Evaluation

- **Effectiveness of improving Resource utilization and Fairness**

$$unfaireness = \frac{\max\limits_{1<\forall i<N_W} SD_i - \min\limits_{1<\forall i<N_W} SD_i}{\max\limits_{1<\forall i<N_W} SD_i}$$



(a) *Resources usage* (b) *Throughput and unfairness*

Overhead

The main overhead of URSA is from scheduling.

URSA identifies the appropriate node for a workload on our 7-node Cloud in 0.12ms using a single thread.

1 Background & Motivation

2 The methodology of URSA

3 Evaluation

4 Conclusion

Conclusion

- Propose
  - Automatically suggest the just-enough resource specification that fulfills the performance requirement of dbPaaS in Public Clouds
- Our work
  - An online capacity planner
  - A performance interference estimator
  - A contention-aware scheduling engine
- Results
  - URSA reduces up to 25.9% of CPU usage, 53.4% of memory and reduces the performance unfairness between the co-located workloads by 47.6% usage without hurting their performance.

Thanks for attention! Q&A

zhang-w@sjtu.edu.cn