Tarcil: Reconciling Scheduling Speed and Quality in Large Shared Clusters

Christina Delimitrou[1], Daniel Sanchez[2]
and Christos Kozyrakis[1]
[1]Stanford University, [2]MIT

SOCC – August 27th 2015

Executive Summary

- Goals of cluster scheduling
  - High decision quality
  - High scheduling speed

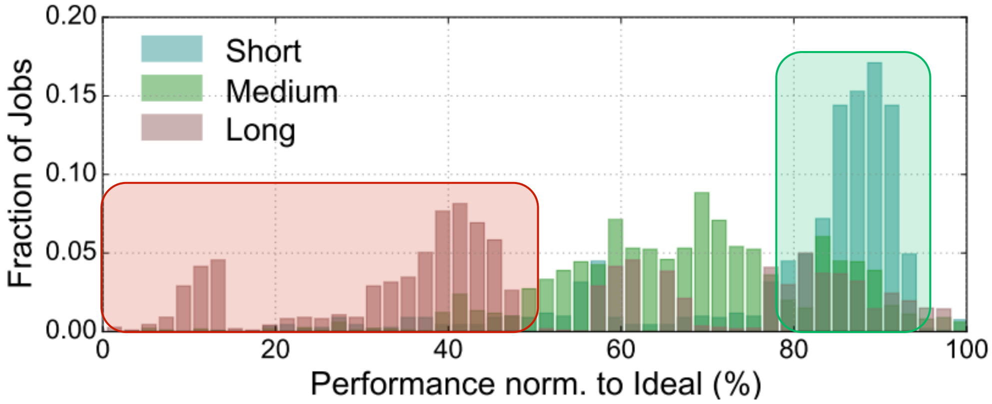
High performance
High cluster utilization

- Problem: Disparity in scheduling designs
  - Centralized schedulers → High quality, low speed
  - Sampling-based schedulers → High speed, low quality

- Tarcil: Key scheduling techniques to bridge the gap
  - Account for resource preferences → High decision quality
  - Analytical framework for sampling → Predictable performance
  - Admission control →High quality & speed
  - Distributed design → High scheduling speed

Motivation

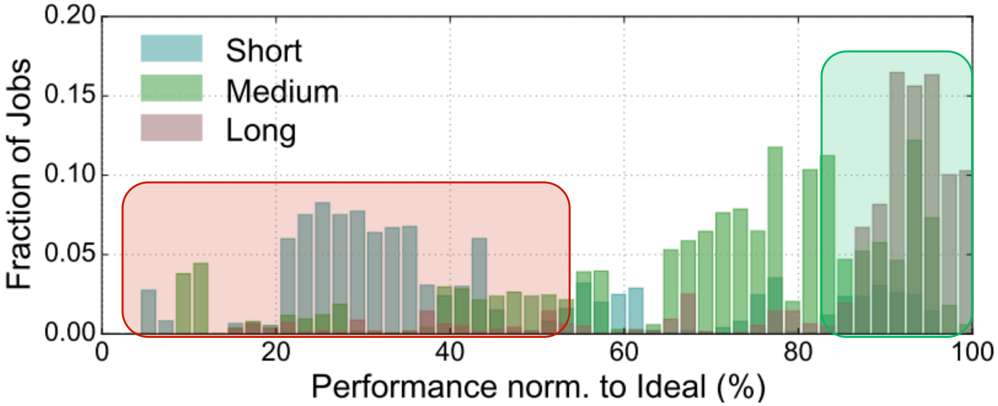
- Optimize scheduling speed (sampling-based, distributed)



Good: Short jobs

Bad: Long jobs

- Optimize scheduling quality (centralized, greedy)



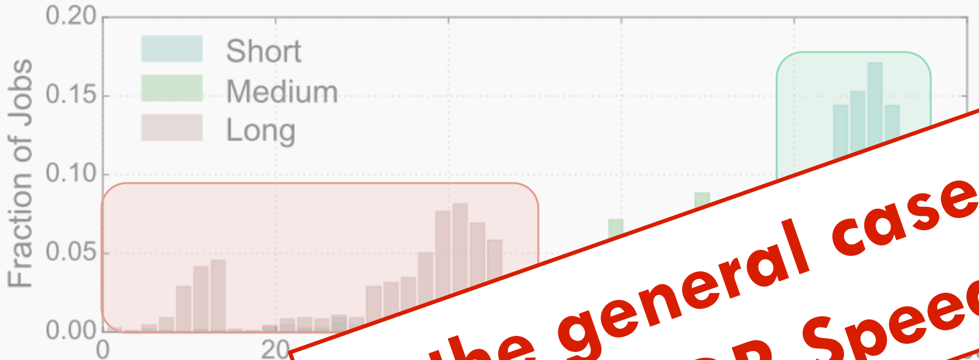
Good: Long jobs

Bad: Short jobs

Short: 100msec, Medium: 1-10sec, Long: 10sec-10min

Motivation

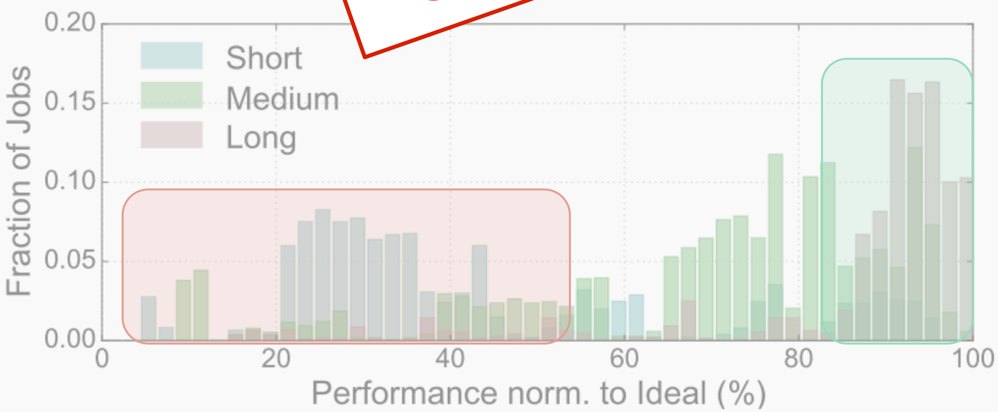
- Optimize scheduling speed (sampling-based, distributed)



Good: Short jobs
Bad: Long jobs

In the general case: Quality OR Speed

- Optimize sched (centralized, greedy)



Good: Long jobs
Bad: Short jobs

Short: 100msec, Medium: 1-10sec, Long: 10sec-10min

Key Scheduling Techniques at Scale

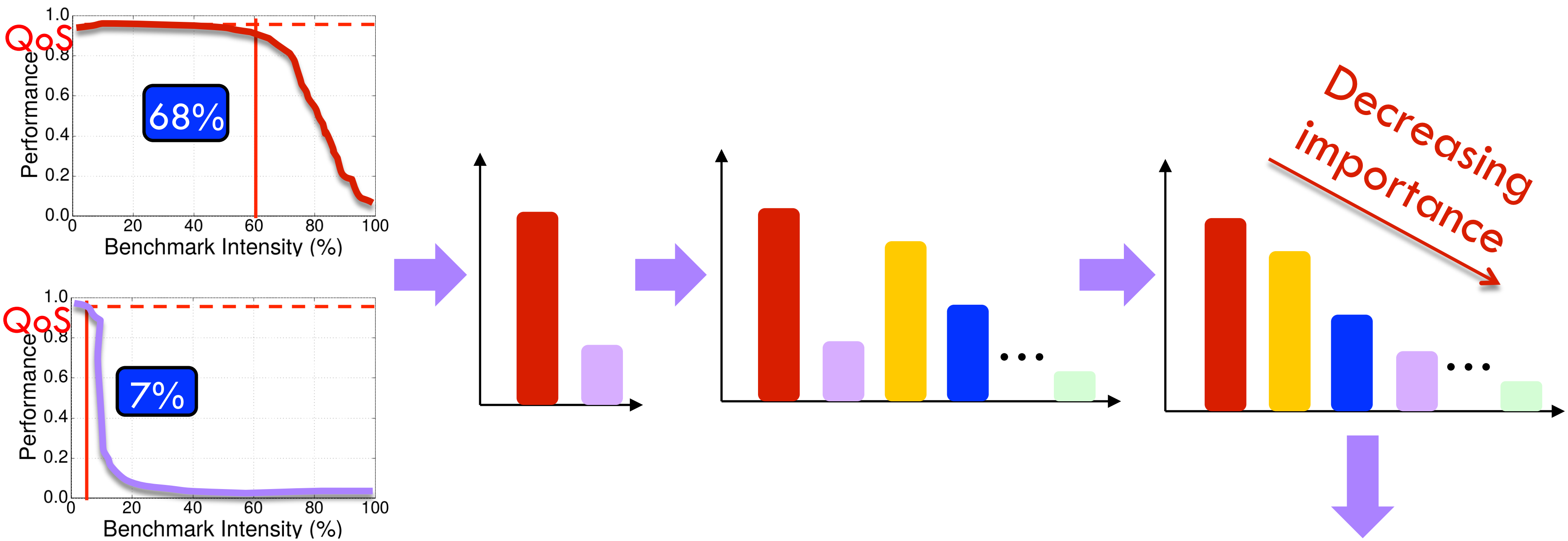
1. Determine Resource Preferences

- Scheduling quality depends on: interference, heterogeneity, scale up/out, …
  - Exhaustive exploration → infeasible
  - Practical data mining framework[1]
  - Measure impact of a couple of allocations → estimate for large space

[1] C. Delimitrou and C. Kozyrakis. Quasar: Resource-Efficient and QoS-Aware Cluster Management. In ASPLOS 2014.

Example: Quantifying Interference

- Interference: set of microbenchmarks of tunable intensity (iBench)



Measure tolerated & generated interference

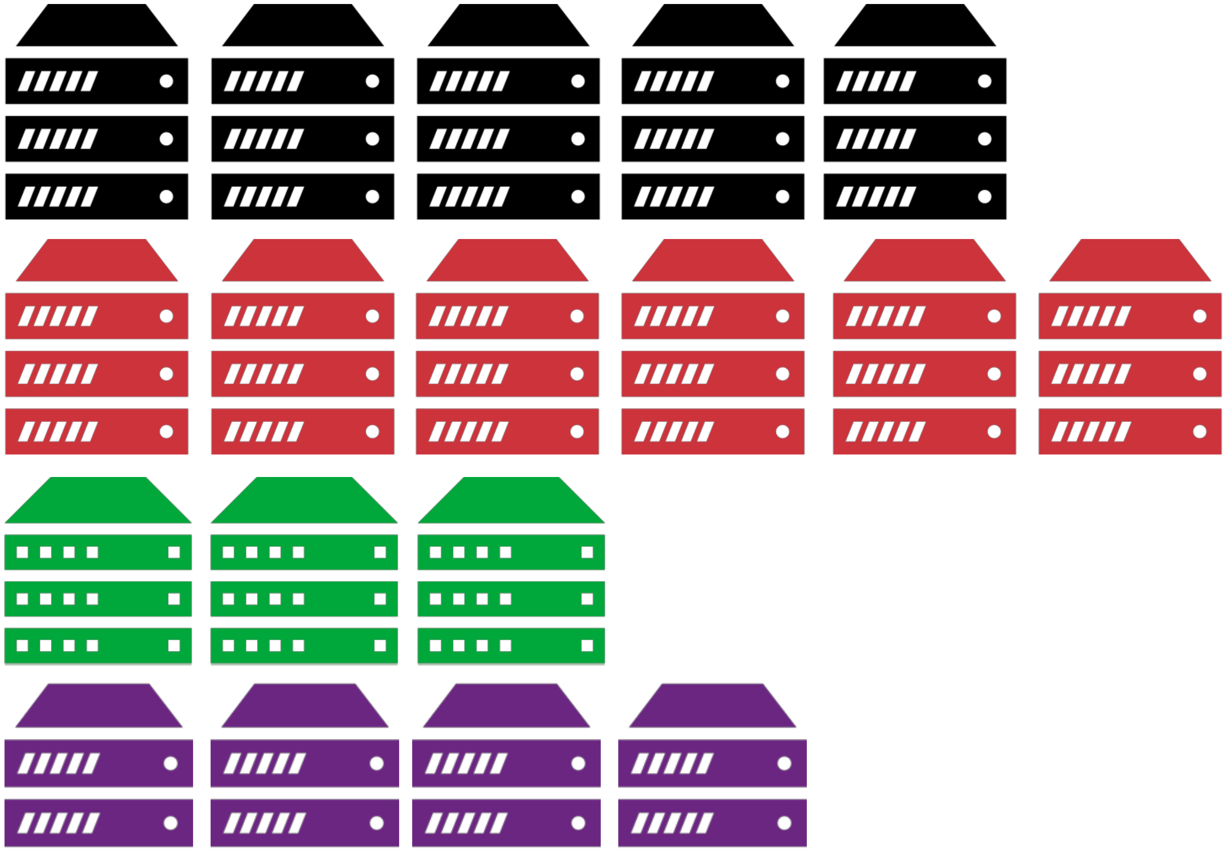
Data mining: Recover missing resources

Resource Quality Q

[1]C. Delimitrou and C. Kozyrakis. Quasar: Resource-Efficient and QoS-Aware Cluster Management. In ASPLOS 2014.

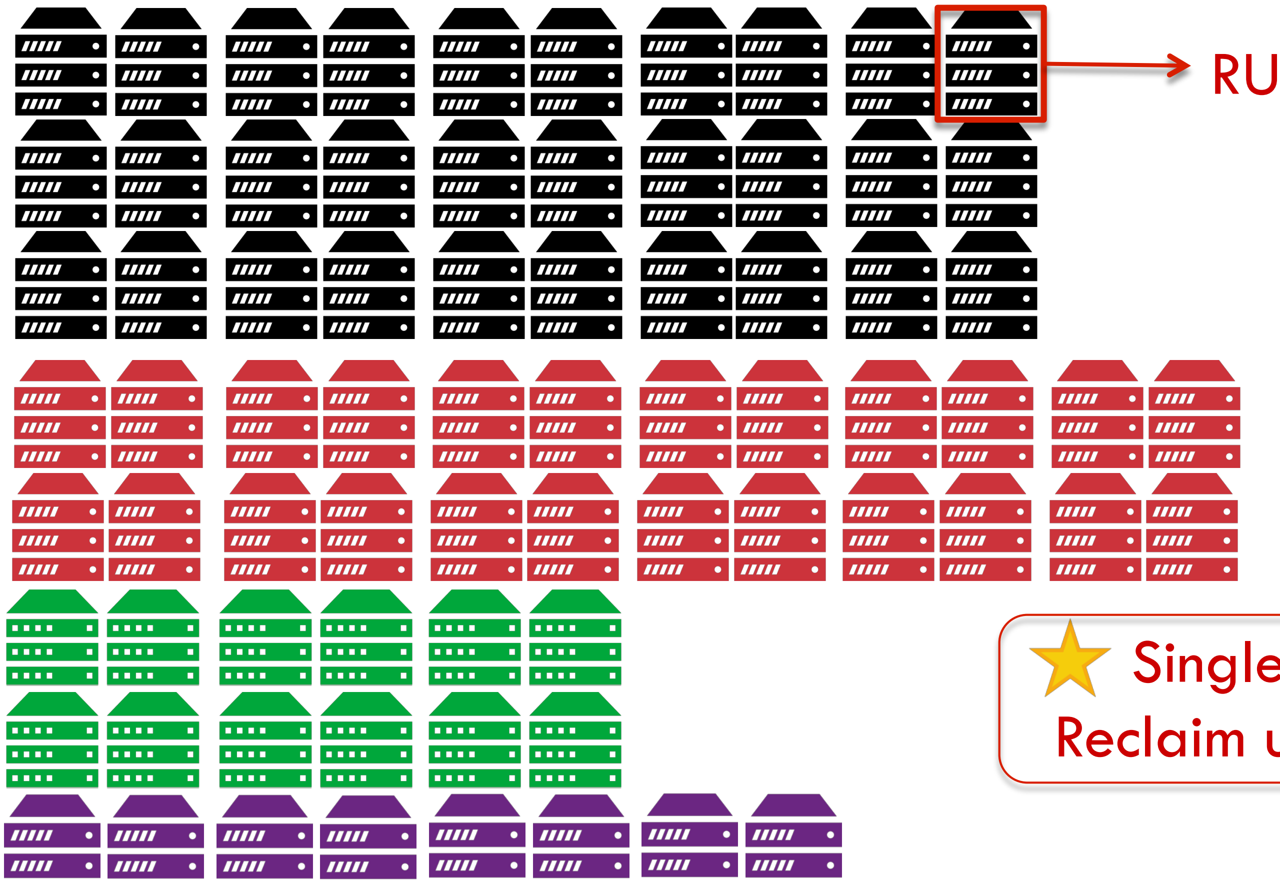
2. Analytical Sampling Framework

- Sample w.r.t. required resource quality



2. Analytical Sampling Framework

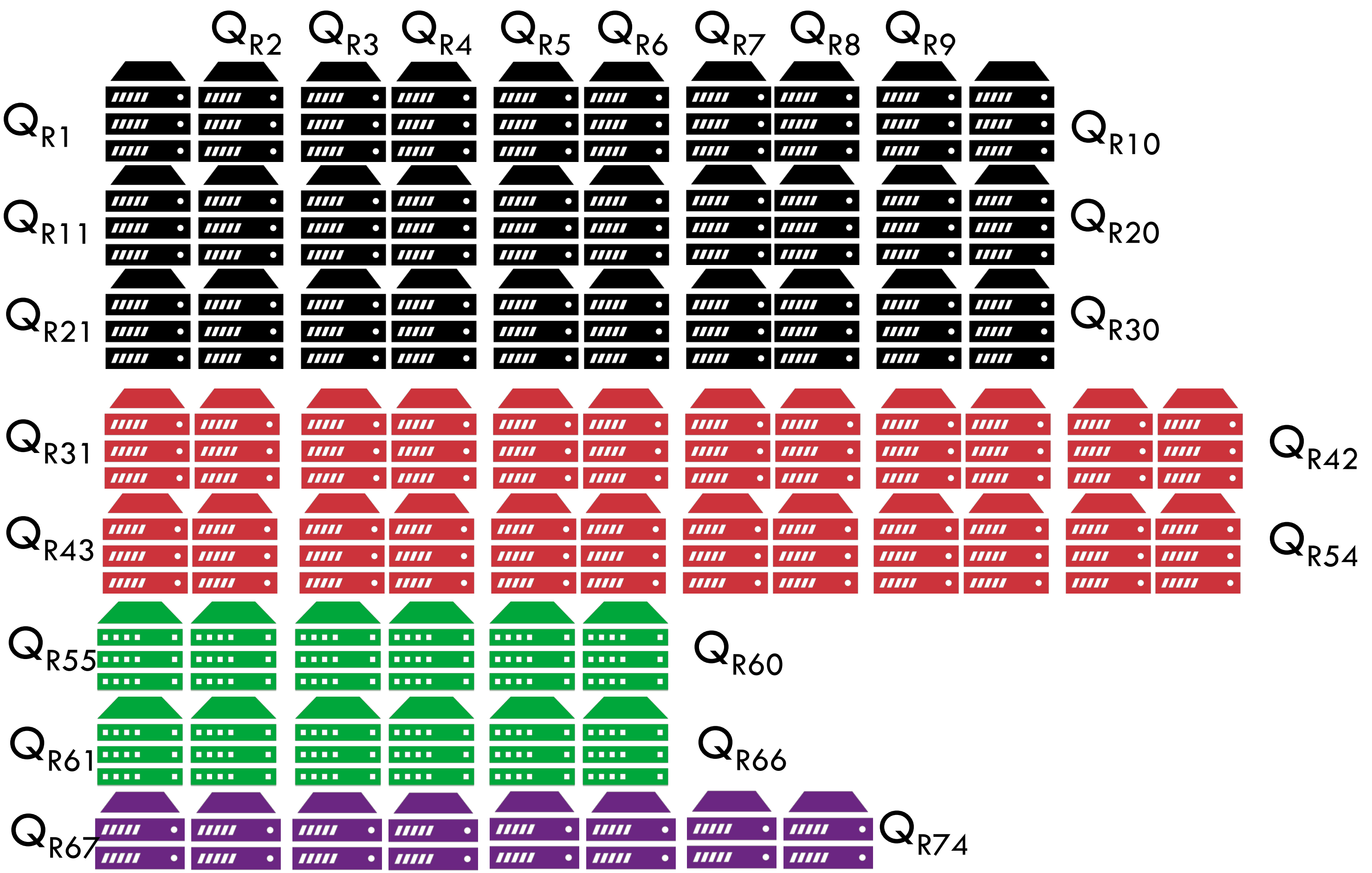
- Fine-grain allocations: partition servers in Resource Units (RU) → minimum allocation unit



★ Single-threaded apps
Reclaim unused resources

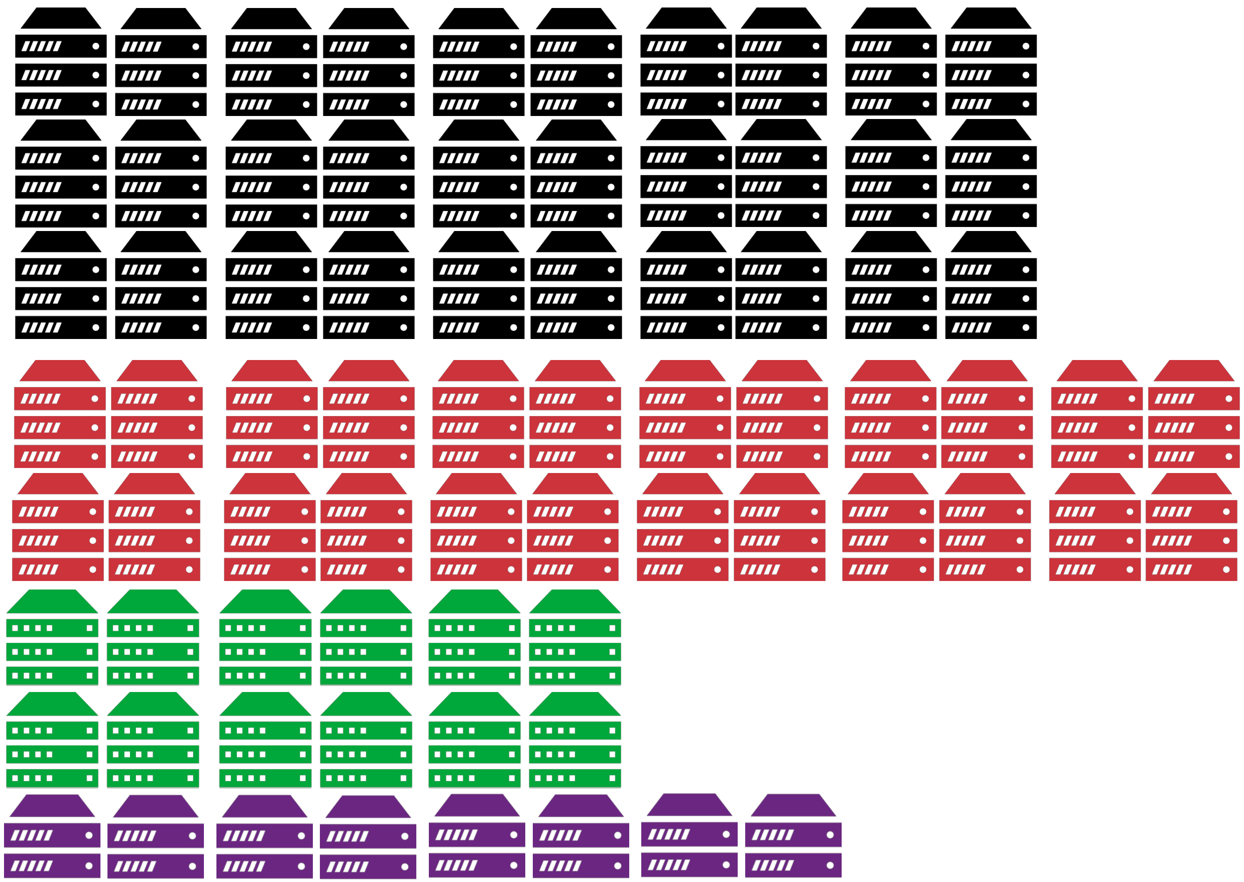
2. Analytical Sampling Framework

- Match a new job with required quality Q to appropriate RUs



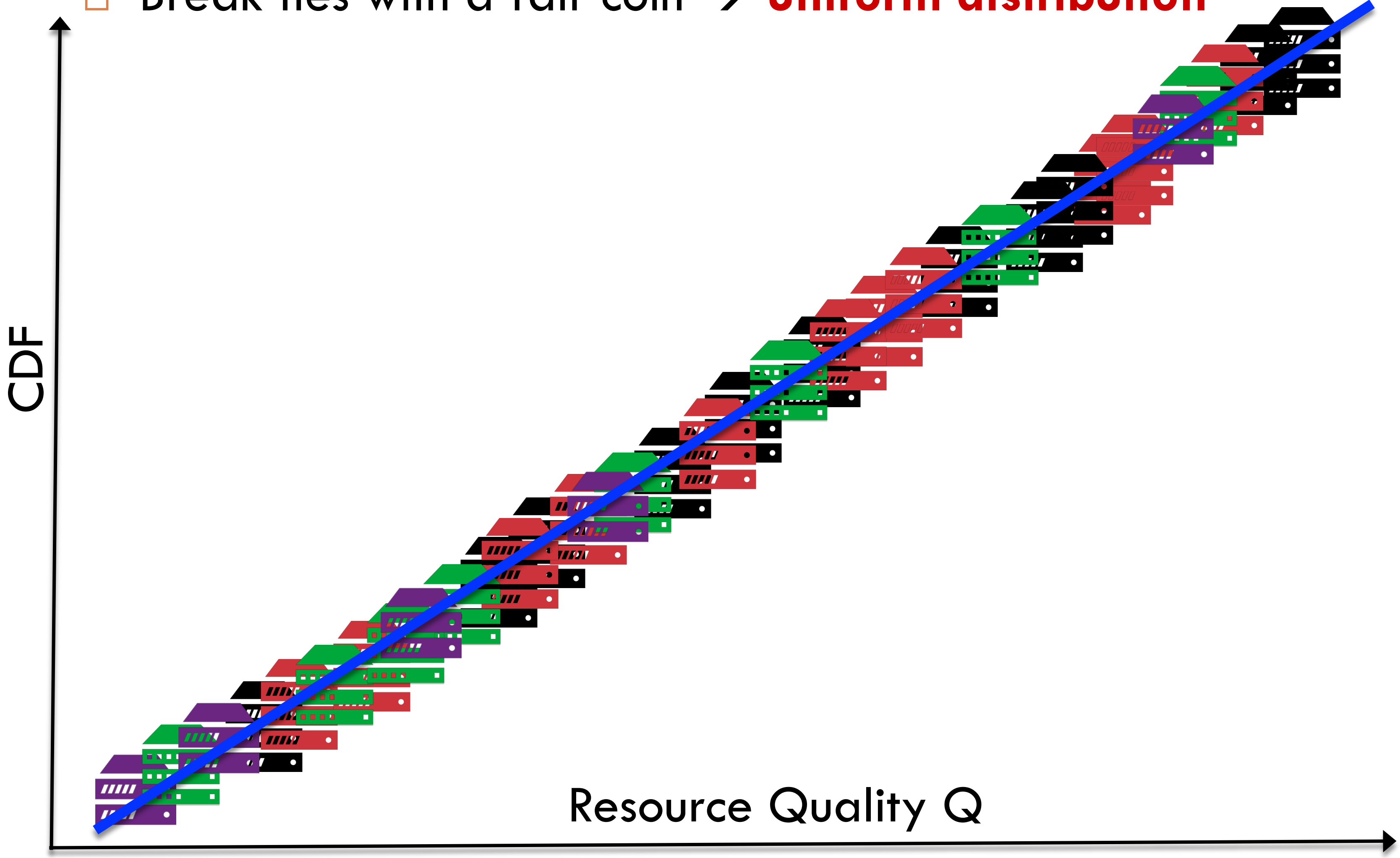
2. Analytical Sampling Framework

- Rank resources by quality



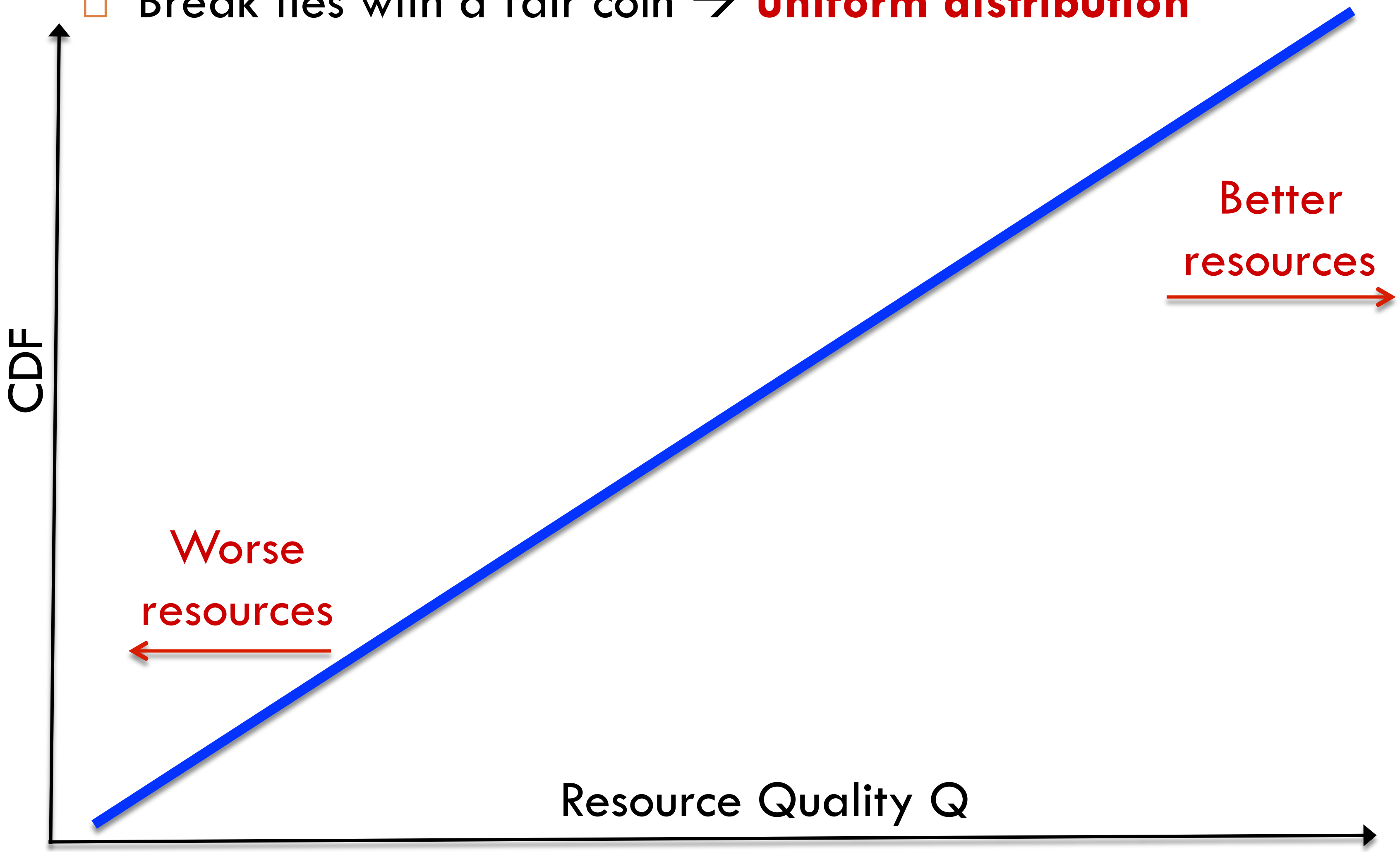
2. Analytical Sampling Framework

- Break ties with a fair coin → **uniform distribution**



2. Analytical Sampling Framework

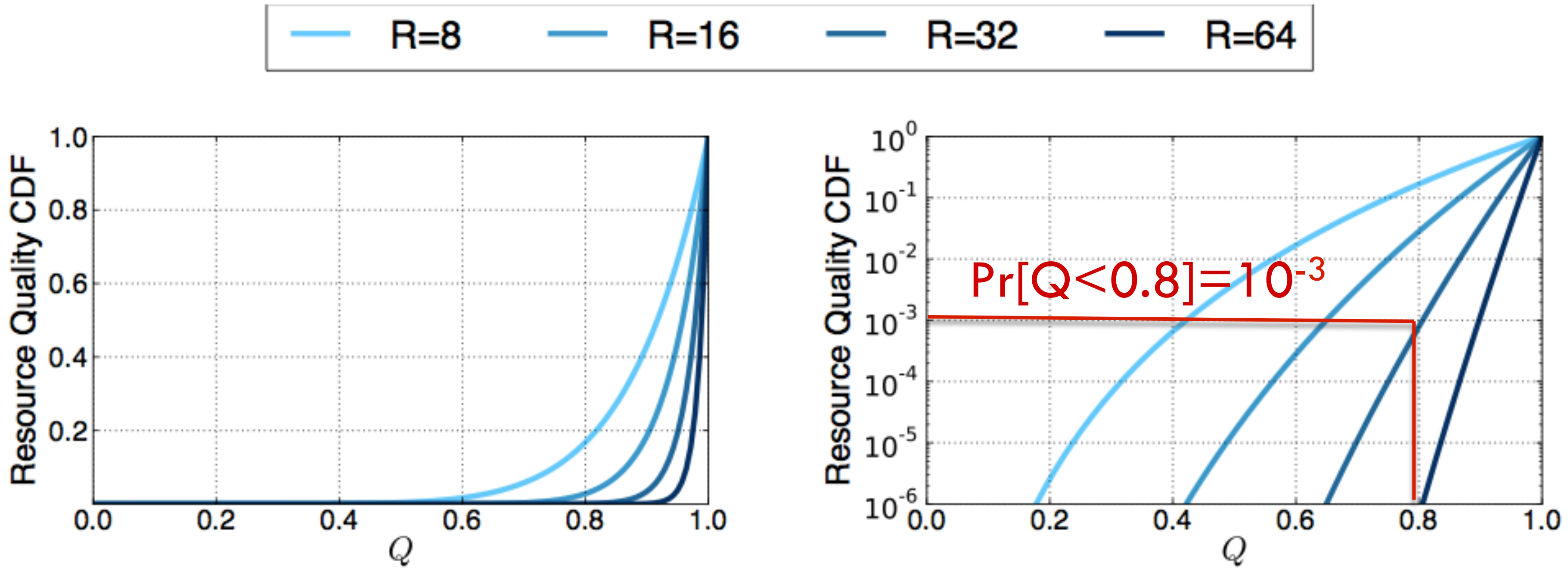
- Break ties with a fair coin → **uniform distribution**



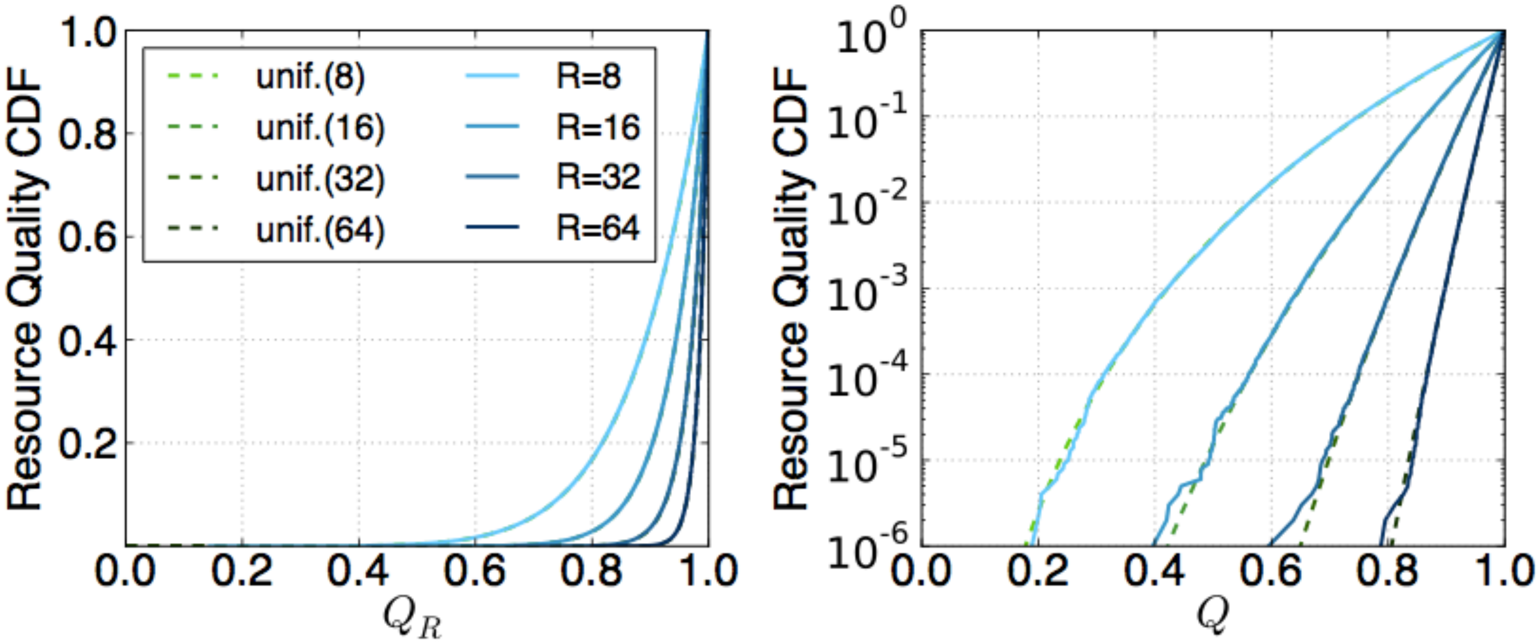
2. Analytical Sampling Framework

- Sample on uniform distribution → guarantees on resource allocation quality

$$\Pr[Q \leq x] = x^R$$



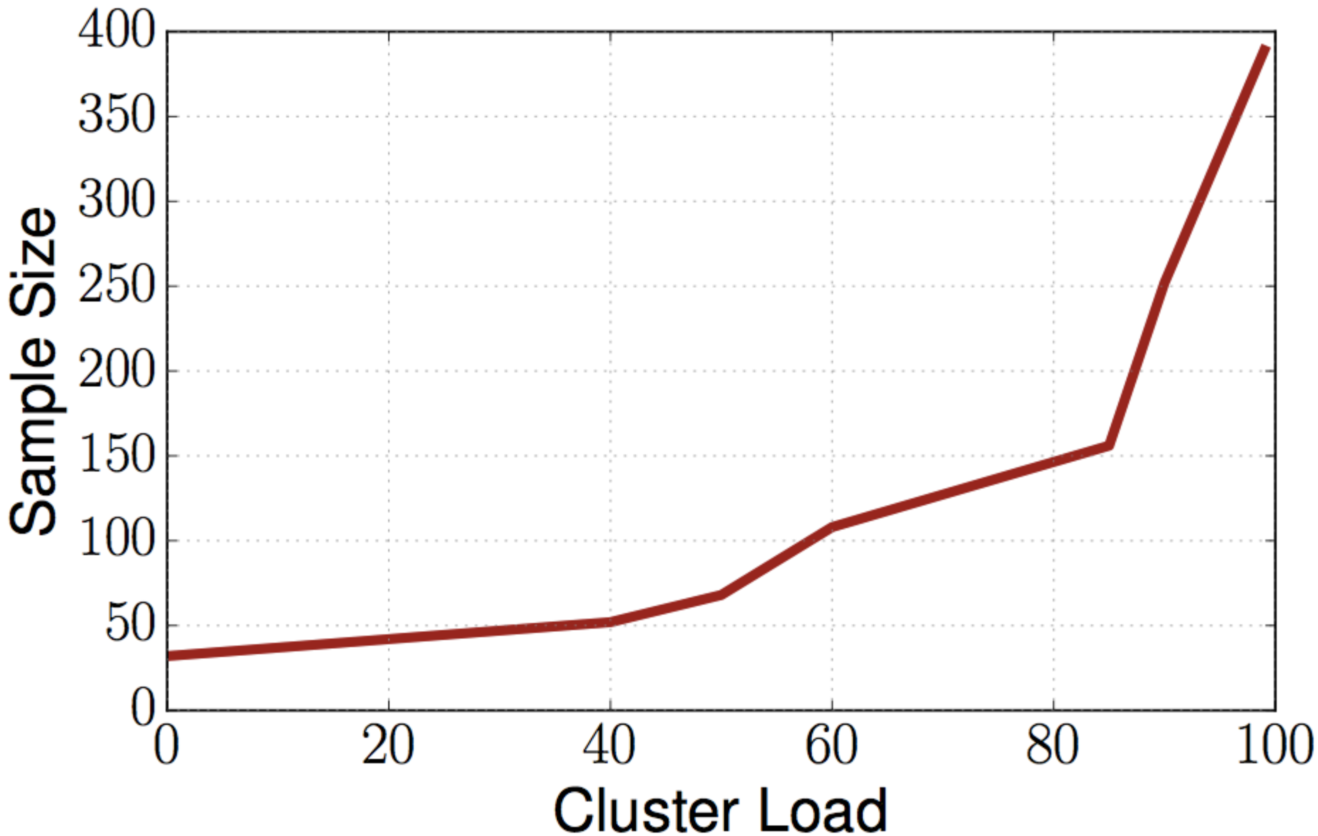
Validation



- 100 server EC2 cluster
- Short Spark tasks
- Deviation between analytical and empirical is minimal

Sampling at High Load

- Performance degrades (with small sample size)
- Or sample size needs to increase



3. Admission Control

- Queue jobs based on required resource quality
- Resource quality vs. waiting time → set max waiting time limit

Tarcil Implementation

- 4,000 loc in C/C++ and Python
- Supports apps in various frameworks (Hadoop, Spark, key-value stores)
- Distributed design: Concurrent scheduling agents (sim. Omega[2])
  - Each agent has local copy of state, one resilient master copy
  - Lock-free optimistic concurrency for conflict resolution (rare) → Abort and retry
  - 30:1 worker to scheduling agent ratio

[2]M. Schwarzkopf, A. Konwinski, et al. Omega: flexible, scalable schedulers for large compute clusters. In EuroSys 2013.

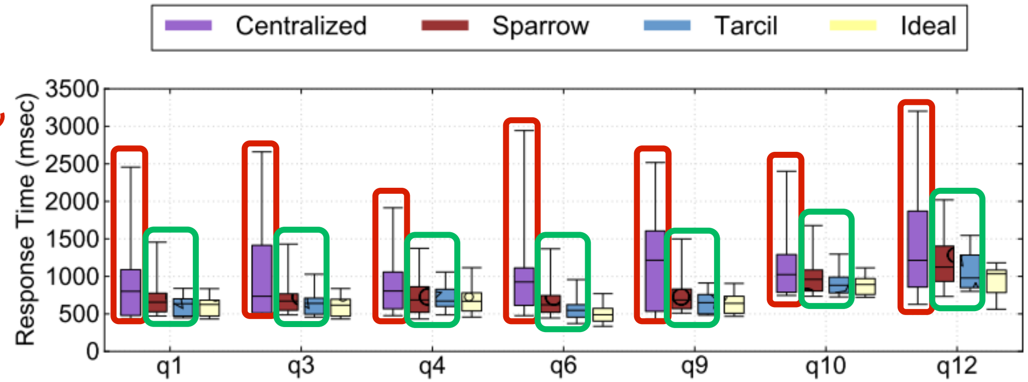
Evaluation Methodology

1. TPC-H Workload
   - ~40k queries of different types
   - Compare with a centralized scheduler (Quasar) and a distributed scheduler based on random sampling (Sparrow)
   - 110-server EC2 cluster (100 workers, 10 scheduling agents)
     - Homogeneous cluster, no interference
     - Homogeneous cluster, with interference
     - Heterogeneous cluster, with interference

- Metrics:
  - Task performance
  - Performance predictability
  - Scheduling latency

Evaluation

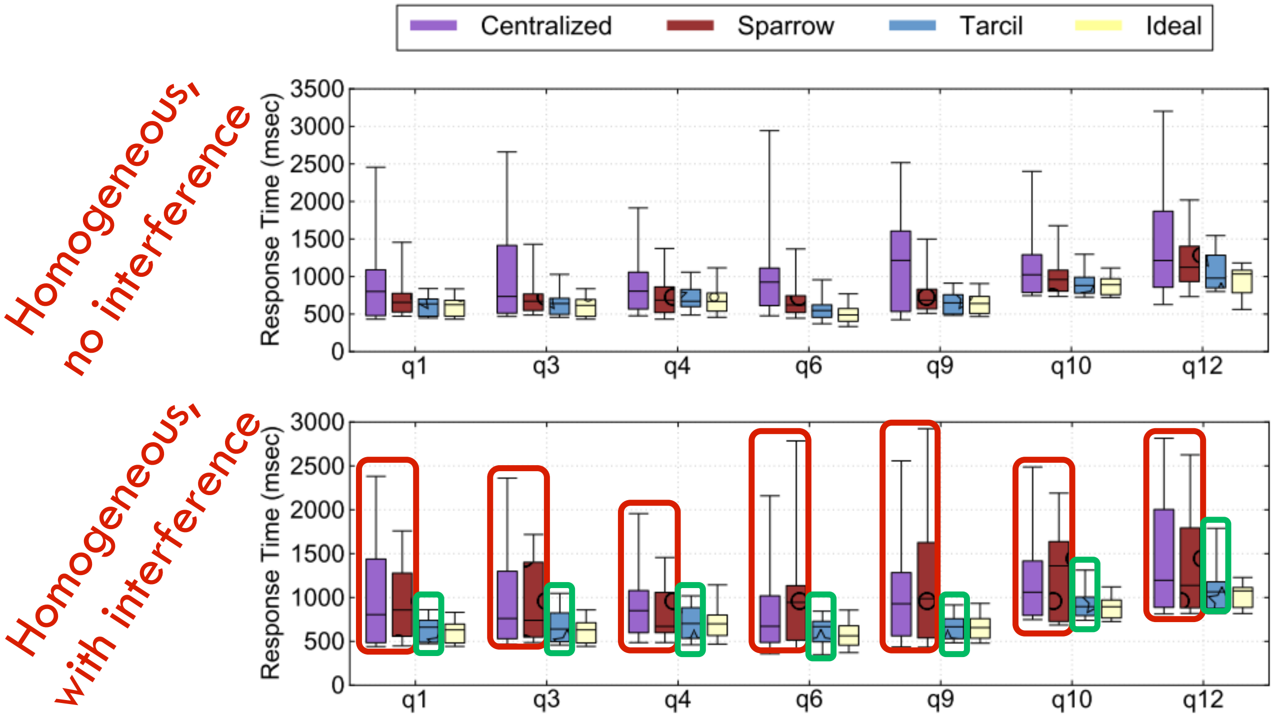
Homogeneous,
no interference



Centralized: high overheads

Sparrow and Tarcil: similar

Evaluation



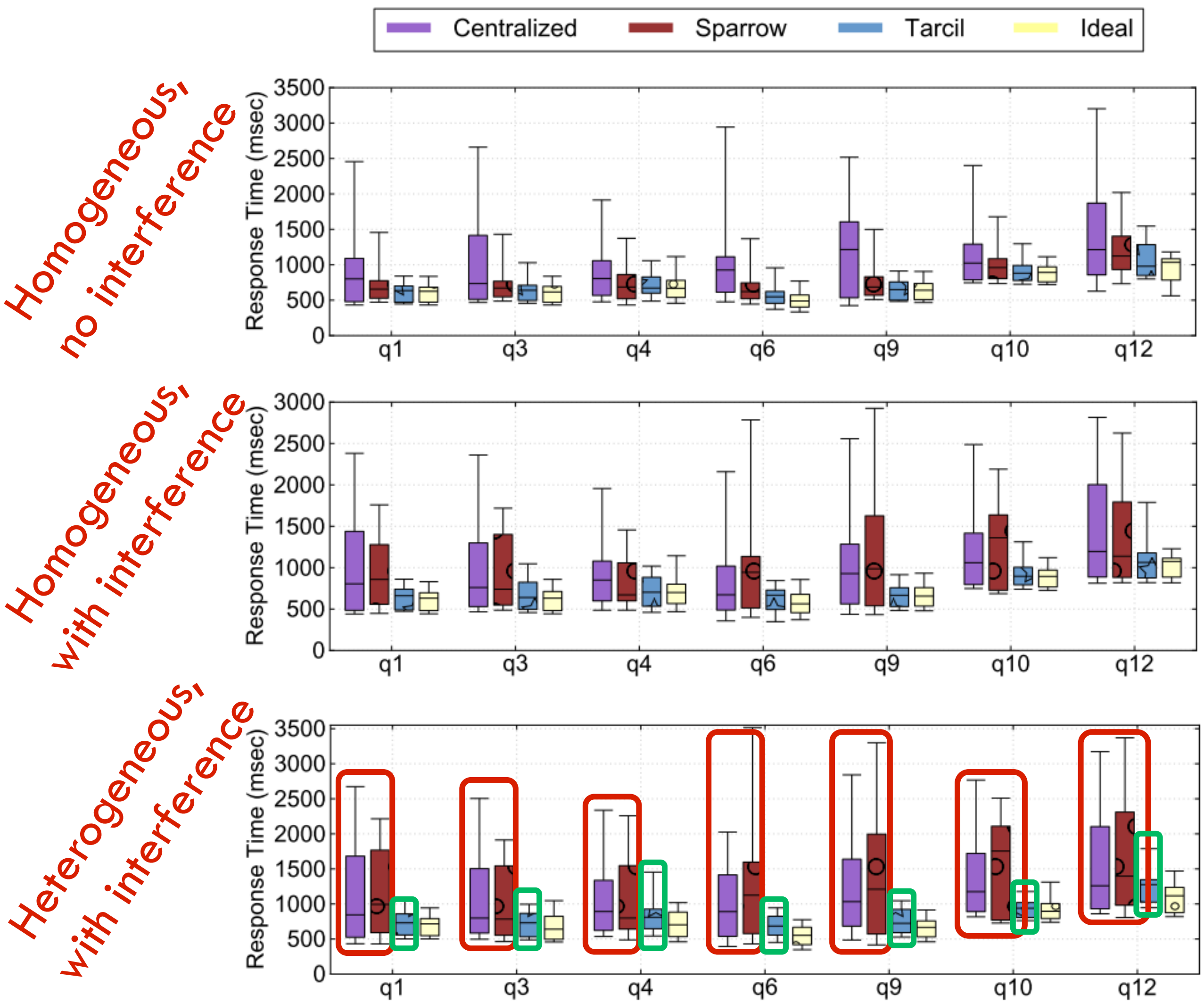
Centralized: high overheads

Sparrow and Tarcil: similar

Centralized and Sparrow: comparable performance

Tarcil: 24% lower completion time

Evaluation



Centralized: high overheads

Sparrow and Tarcil: similar

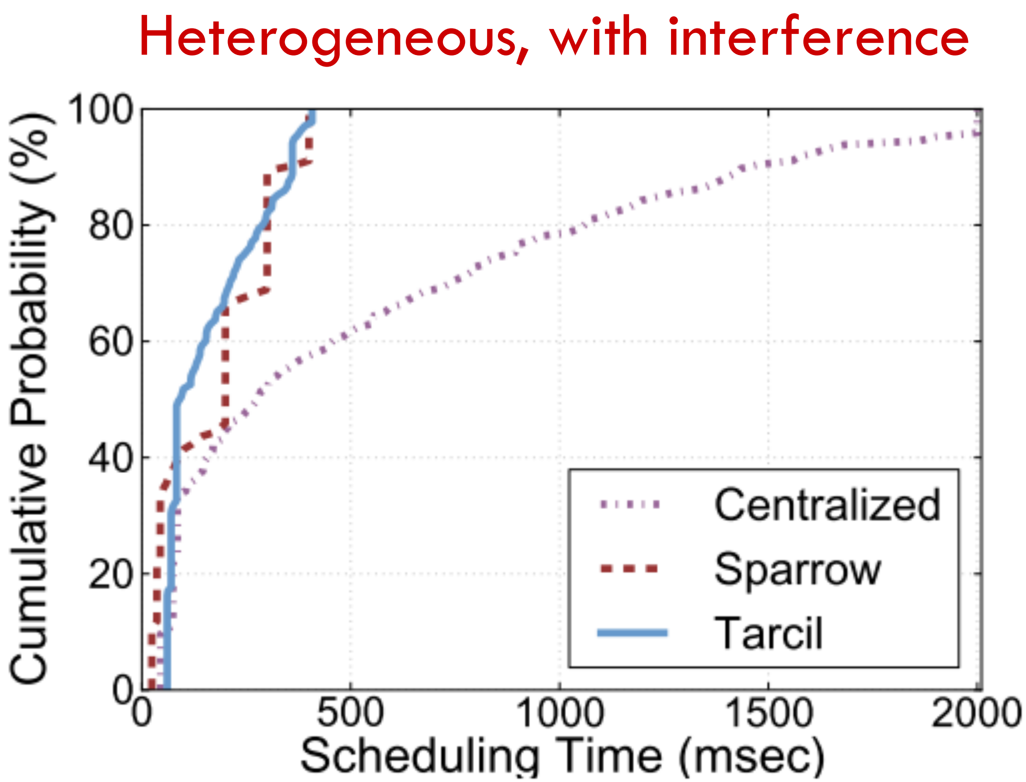
Centralized and Sparrow: comparable performance

Tarcil: 24% lower completion time

Centralized outperforms Sparrow

Tarcil: 41% lower completion time & less jitter

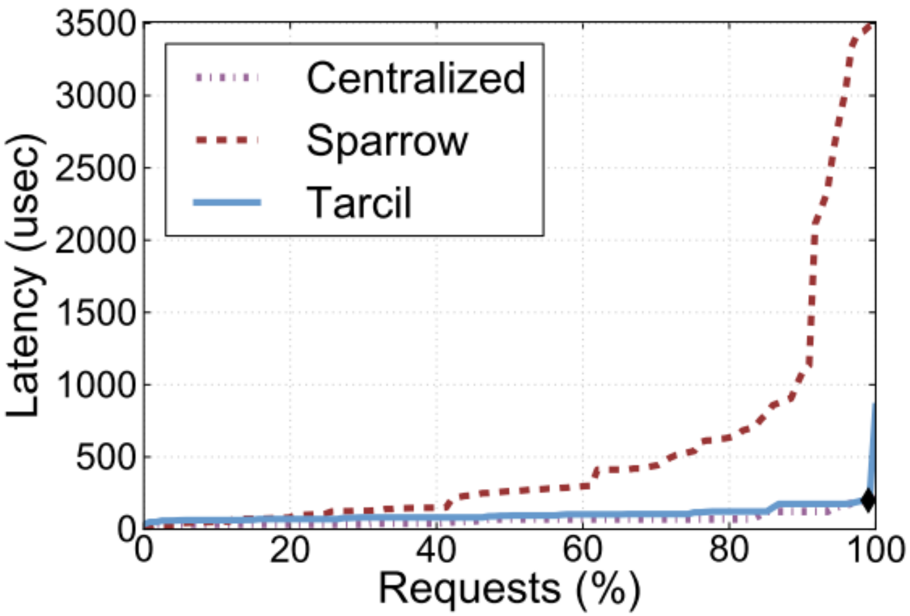
Scheduling Overheads



- Centralized: Two orders of magnitude slower than the distributed, sampling-based schedulers
- Sparrow and Tarcil: Comparable scheduling overheads

Resident Load

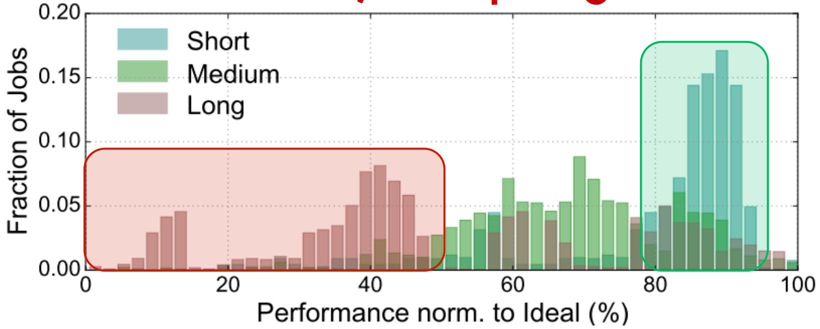
memcached



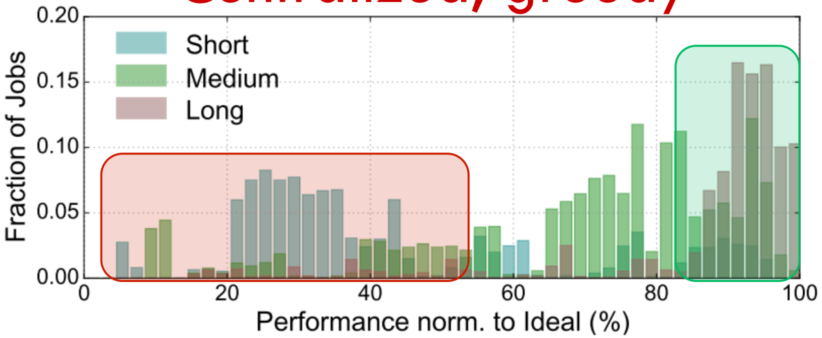
- Tarcil and Centralized account for cross-job interference → preserve memcached's QoS
- Sparrow causes QoS violations for memcached

Motivation Revisited

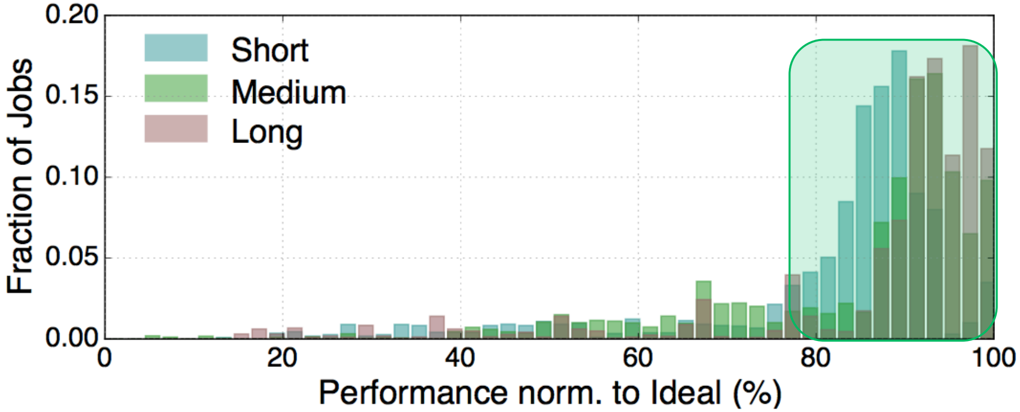
Distributed, sampling-based



Centralized, greedy



Tarcil



Short: 100msec

Medium: 1-10sec

Long:10sec-10min

More details in the paper…

- Sensitivity on parameters such as:
  - Cluster load
  - Number of scheduling agents
  - Sample size
  - Task duration, etc.
- Job priorities
- Large allocations
- Generic application scenario (batch and latency-critical) on 200 EC2 servers

Conclusions

- Tarcil: Reconciles high quality and high speed scheduling
  - Account for resource preferences
  - Analytical sampling framework to improve predictability
  - Admission control to maintain high scheduling quality at high load
  - Distributed design to improve scheduling speed

- Results:
  - 41% better performance than random sampling-based schedulers
  - 100x better scheduling latency than centralized schedulers
  - Predictable allocation quality & performance

Questions?

- Tarcil: Reconciles high quality and high speed schedulers
  - Account for resource preferences
  - Analytical sampling framework to improve predictability
  - Admission control to maintain high scheduling quality at high load
  - Distributed design to improve scheduling speed

- Results:
  - 41% better performance than random sampling-based schedulers
  - 100x better scheduling latency than centralized schedulers
  - Predictable allocation quality & performance

Questions??

Thank you