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

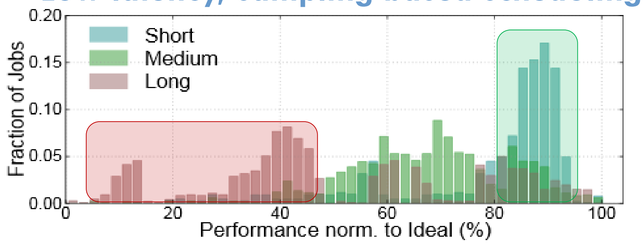
Stanford University

Motivation

Disparity between high quality and low-latency scheduling

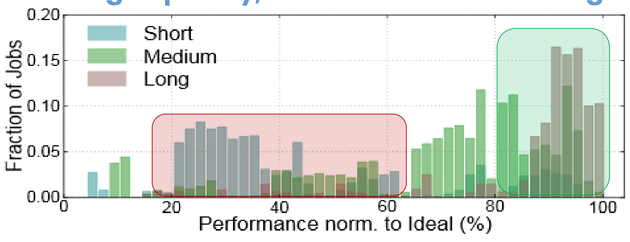
- Centralized, sophisticated → high quality, high latency
- Distributed, sampling-based → low latency, low quality

Low latency, sampling based scheduling



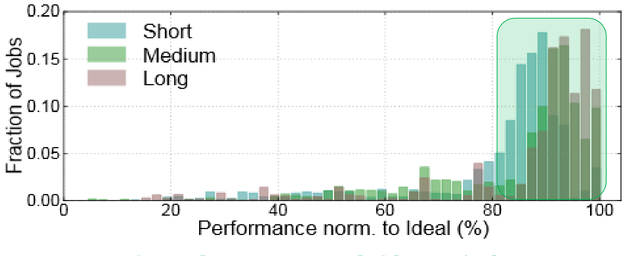
Good: Short jobs **Bad: Long jobs**

High quality, centralized scheduling



Good: Long jobs **Bad: Short jobs**

Tarcil: High quality, sampling-based scheduling



Good: Long and Short jobs

Tarcil Design

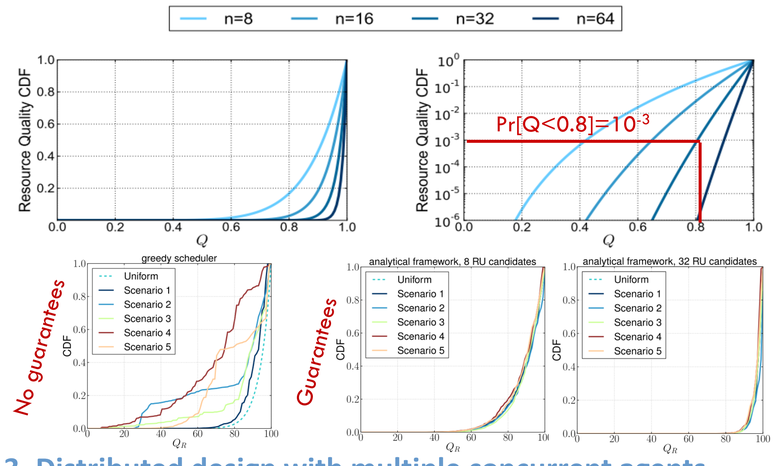
Unpredictability in resource allocation → Overprovisioning

Tarcil: Analytical sampling framework → Strict guarantees on quality of resource allocation

Key ideas:

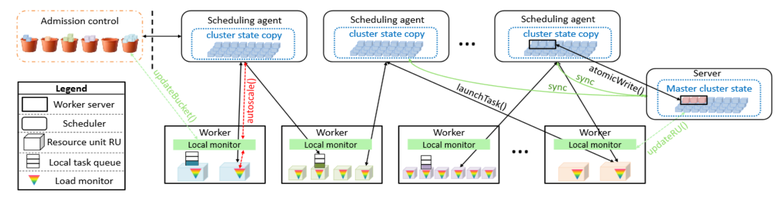
1. Accounts for resource preferences of new jobs (type & amount of required resources)

2. Quality of resource allocation (Q) → Direct function of the sample size



3. Distributed design with multiple concurrent agents

- Each agent has local copy of state, one resilient master copy
- Lock-free optimistic concurrency for conflict resolution (rare)
- Abort and retry on conflict
- Admission control to avoid machine oversubscription

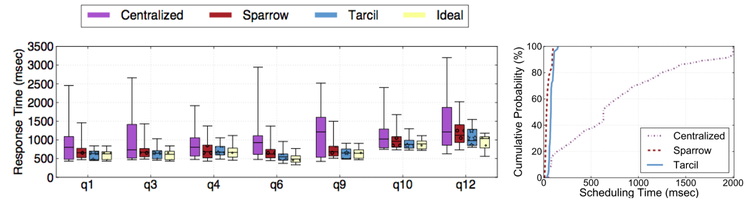


Tarcil Evaluation

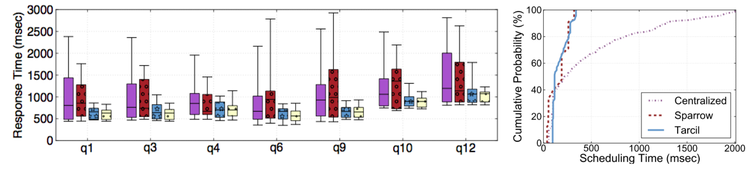
1. Short jobs:

- TPC-H queries → short Spark tasks (200-800msec)
- *Ideal*: zero scheduling delay, perfect resource selection
- 110-machine EC2 cluster

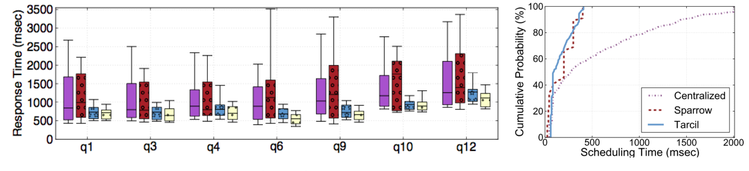
A. Homogeneous cluster, no interference



B. Homogeneous cluster, with interference



C. Heterogeneous cluster, with interference



2. Impact on resident load:

- Memcached resident load (200 usec 99th %ile latency QoS)
- 100msec homogeneous Spark tasks as incoming load