Improving Resource-Efficiency in Cloud Computing
Christina Delimitrou and Christos Kozyrakis
Stanford University

Datacenter Underutilization
- Datacenters greatly underutilized (25-35% for Google, 6-8% for Amazon EC2) → prevents scalability, increases cost
- Overprovisioned reservations
- Workload interference
- Heterogeneous platforms
- Insufficient info on apps

Utilization on a large production cluster at Twitter:
- Collected over 1 month
- Running over the Mesos scheduling system
- Serving latency-critical, user-facing workloads

Is underutilization the cluster manager’s fault?
- Conservative users: Reservations exceed usage by 5-10x
- The cluster appears to be running close to capacity

Determining the right amount/type of resources is hard!

Quasar: Resource-Efficient Cluster Management

Insight: Move the responsibility of determining the right amount/type of resources to the cluster manager

Key ideas:
1. User specifies a performance target
2. Leverage the system’s existing knowledge to minimize overheads → app classification
3. Account for allocation and assignment jointly

Profiling: sparse input signal to classification

Parallel classifications: translate performance to scale-up, scale-out, heterogeneity and interference → similarities between old and new workloads

Greedy selection: select the min amount of resources that satisfy the performance target

Total overheads: a few sec (up to 2 min for apps with long setup times)

Quasar monitors performance and adjusts allocation at runtime
- Execution in isolation
- Scale-out or scale-up
- Complete rescheduling

Quasar Evaluation
A. Two stateful latency-critical apps + best-effort workloads

B. General cloud provider case (1200 apps, 200 EC2 servers)

On-going/Future Work
Improving Performance Predictability
Unpredictability in resource allocation → overprovisioning
Analytical framework based on sampling → Strict guarantees on predictability of resource allocation
Accounts for app preferences, load changes, etc.
Simplifies user billing: based on used resources + strictness of QoS constraints (function only of sampled resources)