QoS-Aware Admission Control in Heterogeneous Datacenters

Christina Delimitrou, Nick Bambos and Christos Kozyrakis

Stanford University
Cloud DC Scheduling

- Workloads are unknown → random apps submitted for short periods
- Significant churn (app arrivals/departures) → not large long-running apps
- High variability in workloads (runtime, number of threads, etc.)
- Fast admission & scheduling decisions
Users are Interested in

- Fast Execution Time
  - The amount of time the job needs to run

- Low Waiting Time
  - The amount of time the job is waiting before it gets scheduled
Executive Summary

Problem: Admission control in large-scale cloud DCs (e.g., EC2, Azure)
- Heterogeneity $\rightarrow$ performance/efficiency
- Interference $\rightarrow$ performance loss from high interference
- High arrival rates $\rightarrow$ system can become oversubscribed

Background: Paragon is a heterogeneity and interference-aware scheduler for cloud DCs.

Limitations: In high-load scenarios demanding workloads can block easy-to-satisfy applications $\rightarrow$ head-of-line blocking $\rightarrow$ long waiting time

ARQ is an admission control protocol for cloud DCs that is:
- Application-aware: Accounts for the resource quality of each app
- QoS-aware: Queues applications s.t. their QoS guarantees are preserved
- Scalable: Scales to 10,000s of applications and servers
- Lightweight: Low and upper-bound queueing overheads
Users are Interested in

- **Fast Execution Time**: The amount of time the job needs to run
- **Low Waiting Time**: The amount of time the job is waiting before it gets scheduled

Paragon

ARQ
Background: Paragon

- **Classification:** ~Netflix Challenge
  - Small information signal about new application
  - Leverage system knowledge about previously scheduled applications
  - Collaborative filtering techniques (SVD + PQ reconstruction with SGD)
    → Scheduling recommendations: Heterogeneity + Interference

- **Greedy Scheduler:**
  - Co-schedule workloads with no/small interference on suitable hardware platforms
    → preserve QoS & improve utilization

[Diagram showing the flow of application classification, heterogeneity, interference, and scheduling recommendations with server platform and system state metrics.]
Limitations

- **Scheduling in FIFO order:**
  - Applications with small resource requirements get blocked behind demanding workloads → head-of-line-blocking → long queueing delays
  - Short jobs get blocked behind long jobs
  - High-priority jobs get blocked behind low-priority jobs

- **Resource-agnostic queueing of applications:**
  - Application in the head of the queue gets dispatched to first available server → not necessarily a suitable server for that workload
ARQ: Application-aware Admission Control

- **Resource Quality**: Degree of tolerated and caused interference in various shared resources (higher quality means more demanding application)

  \[ Q_i = \sum_{k} c_k \quad \text{For application } i \]

  \[ Q_j = \sum_{k} t_k \quad \text{For server } j \]

- **Resource quality-aware queueing**: Applications are queued based on the resource quality they need

- **Multi-class admission control**: Each class corresponds to apps with specific range of Qi → dispatched to servers with the required Qj

- **Preserving QoS**: Applications can be diverged to different queues to preserve their QoS (when waiting time is high)
ARQ Design

Q1: [90,100]
Q2: [80,90]
Q3: [70,80]

Q10: [0,10]

Higher quality resources
ARQ Design

Q1: [90,100]
Q2: [80,90]
Q3: [70,80]
Q10: [0,10]
ARQ Design

Q1: [90,100]
Q2: [80,90]
Q3: [70,80]
Q10: [0,10]
ARQ Design

Q1: [90,100]
Q2: [80,90]
Q3: [70,80]
Q10: [0,10]
ARQ Design

Q1: [90,100]
Q2: [80,90]
Q3: [70,80]
Q10: [0,10]
ARQ: Queue Switching -- Utilization

Q1: [90,100]
Q2: [80,90]
Q3: [70,80]
Q10: [0,10]

If no applications in higher queue diverge up \(\rightarrow\) suboptimal utilization but maintains QoS
ARQ: Queue Switching -- QoS

Q1: [90,100]
Q2: [80,90]
Q3: [70,80]
Q10: [0,10]

If server available diverge to lower queue → some QoS degradation
Switching between Queues

- Statistically analyze per-pool freed-server-time $\rightarrow$ distribution fitting (represent using known distributions)
- Updated every time a new server is freed
- From CDFs of per-pool freed-server-time compute the optimal switching point between queues
Switching between Queues

- Optimization function:
  - Find switching time $t$ s.t.:
    
    $$
    \text{maximize } \text{Prob[server is freed]},
    $$
    
    $$
    \text{subj. total waiting time preserves QoS}
    $$

- Solving the optimization problem is fast ($\sim$ msec) and scalable ($O(n)$) even for large numbers of applications and servers.
Methodology

- **Workloads:**
  - Single-threaded: SPEC CPU2006
  - Multi-threaded: PARSEC, SPLASH-2, BioParallel, Minebench, Specjbb
  - Multiprogrammed: 4-app mixes of SPEC CPU2006 workloads
  - I/O-bound: Hadoop + data mining (Matlab)

- **Small scale:**
  - 40 servers, 10 server configurations (Xeons, Atoms, etc.)
  - 178 applications used in four *workload scenarios*:
    - *Low load, high load and oversubscribed*

- **Large scale:** 1,000 EC2 servers, oversubscribed scenario (8,500 apps)
Evaluation: Small Scale

- Paragon + ARQ preserves QoS for 95% of workloads $\rightarrow$ 94% without ARQ
- Average performance is 99.6% of optimal
Evaluation: Small Scale

Paragon + ARQ preserves QoS for 82% of workloads → 64% without ARQ

Average performance is 98% of optimal
Evaluation: Large Scale (EC2)

- Paragon preserves QoS for 75% of workloads → 61% without ARQ
- Bounds degradation to less than 10% for 99% of workloads
Other experiments

- Workload scenario with application phases (app requirements change)
- Shortest Job First (SJF) and priorities
- Queueing overheads
- Sensitivity to parameters (e.g., number of queues, etc.)
- Distributions of server freed times
Conclusions

- ARQ leverages Paragon to classify applications in multiple queues such that QoS guarantees are preserved and utilization is maximized.
- It improves performance both for low and especially for oversubscribed workload scenarios.
- It is scalable and lightweight.
Questions??

Thank you