Problem/Motivation

- **Goal:** Meet per-application tail latency Service Level Objectives (SLOs)
- in shared networked storage infrastructures
- with bursty applications

- **Challenges:**
  - End-to-end latency is affected by all stages (storage & network)
  - Bursts affect tail latencies of workloads sharing infrastructure

PriorityMeister Controller Design

- Profiles workloads to proactively identify bottlenecks in storage & network
- Configures priorities and rate limits to meet SLOs under a worst-case model

Application specification:
- Latency SLO: Maximum acceptable latency of a request
- Throughput SLO: Total time to complete a set of requests
- History of access patterns: Trace of recent requests
- Location of client & server: Pair of IP addresses for client and server VMs

- PriorityMeister (PM) satisfies SLOs (dashed lines) across all percentiles

System Architecture

- Application specification
- QoS parameters (priority, rate limits)
- Profile data
- Storage/network enforcers

QoS Enforcement

- Storage & network QoS is enforced at end-hosts only using local state
- Priority - provides latency differentiation
- Rate limits - provides performance isolation and prevents starvation

Results

- Tail latencies for 3 co-located workloads across multiple policies
- Only PriorityMeister (PM) satisfies SLOs (dashed lines) across all percentiles

Latency differentiation

- FIFO
- Round Robin
- Priority

99th percentile latency

- Requires low latency
- Requires medium latency
- Latency insensitive

Co-located workload: Behaving Misbehaving

- Rate limiting enabled
- Rate limiting disabled

90th percentile latency

99th percentile latency

99.9th percentile latency

99.99th percentile latency