CARD: A Congestion-Aware Request Dispatching Scheme for Replicated Metadata Server Cluster

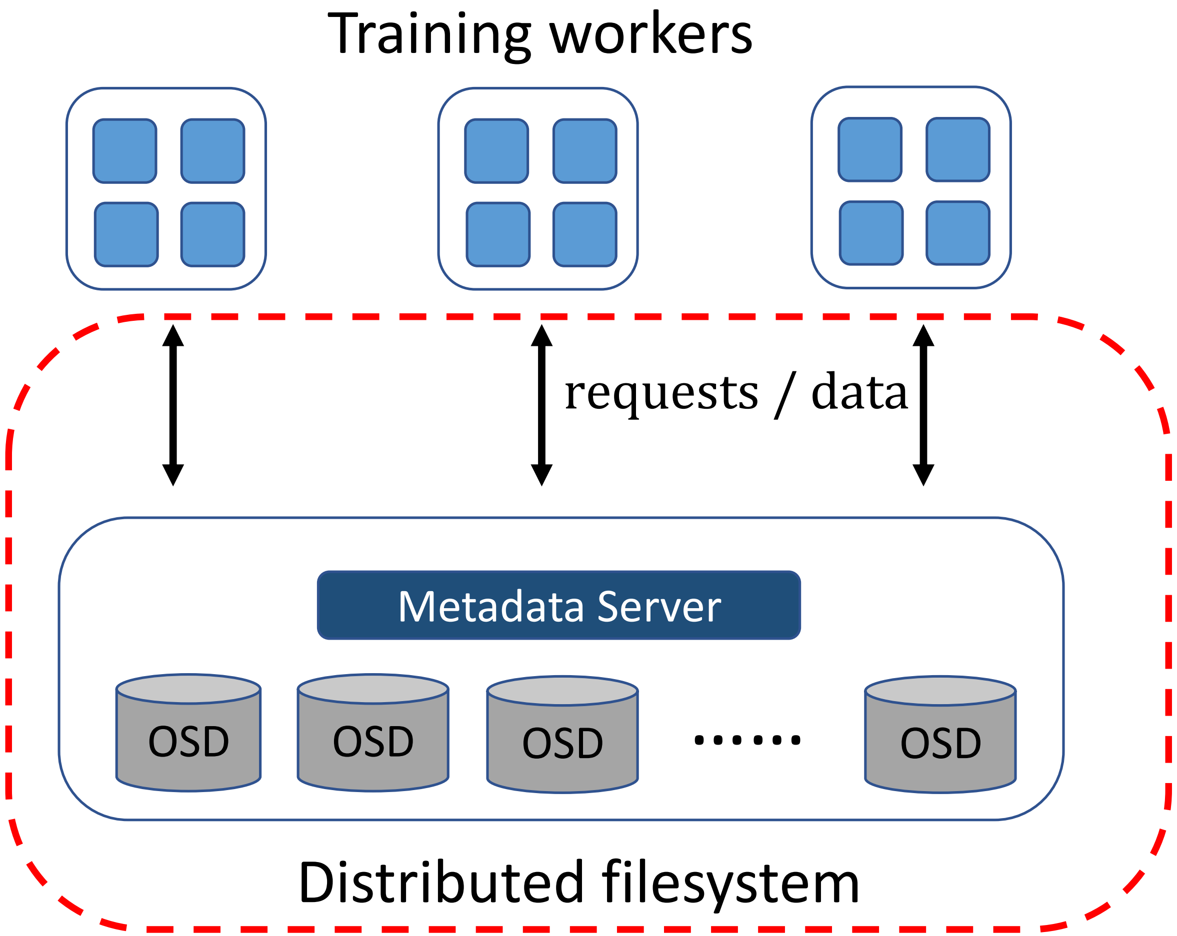
Shangming Cai, Dongsheng Wang,
Zhanye Wang and Haixia Wang
Tsinghua University

Background: Massive-scale ML in product environments

- Datasets updated hourly or daily
  - data collected and stored in an HDFS-like distributed filesystem
  - periodically offline training for online inference
- Challenges of the data-reader pipeline while training
  - extremely heavy read workloads: millions to billions of files per epoch
  - random access pattern: up-level shuffling for convergence speed

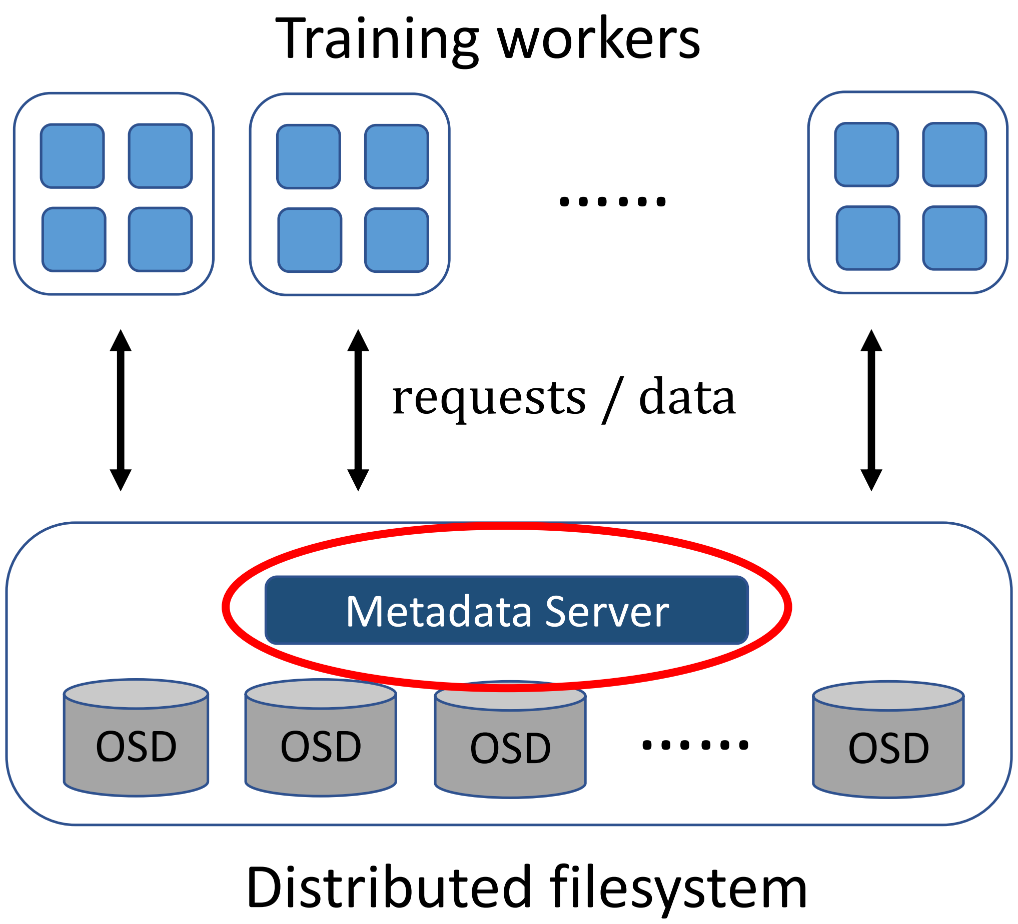
Background: Massive-scale ML in product environments

- Workers interact with a DFS
- Metadata request
  -> metadata server (MDS)
- File I/O
  -> object storage devices (OSD)



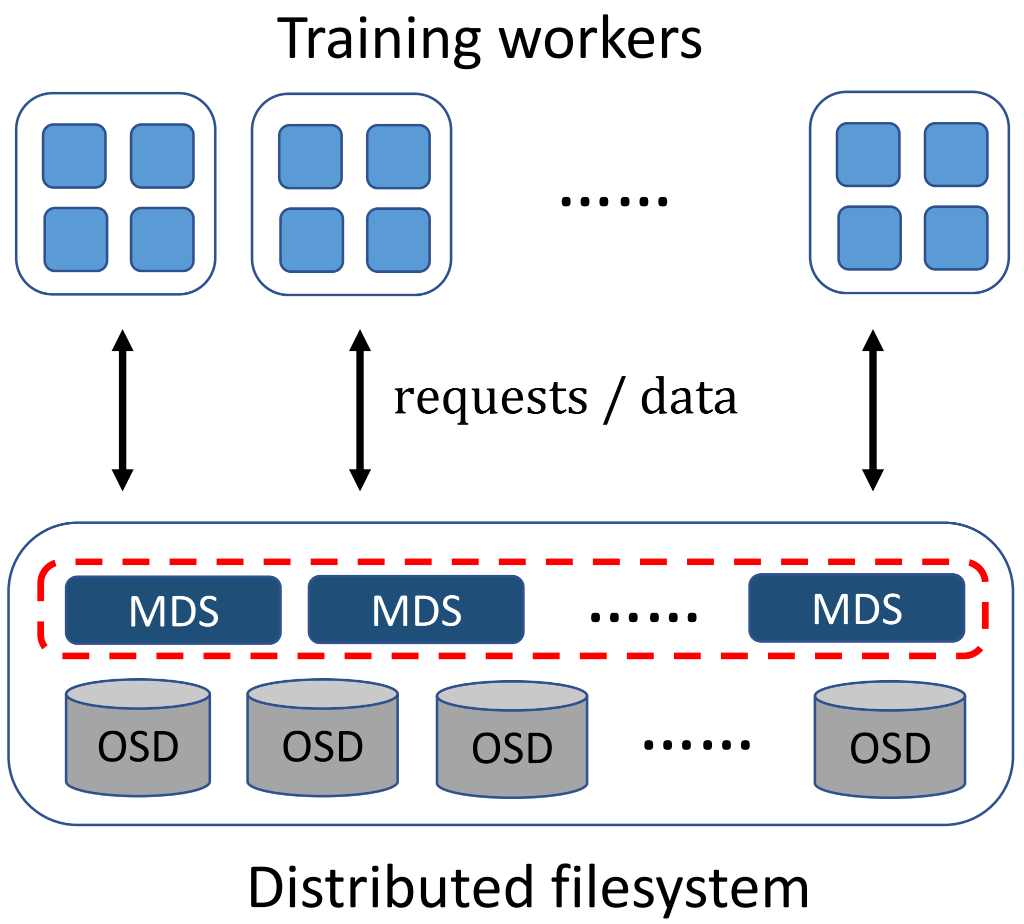
When the number of training workers grows…

- Extremely stressed workloads
- Metadata access step bottlenecks the data-reader pipeline
- Potential single point of failure on MDS



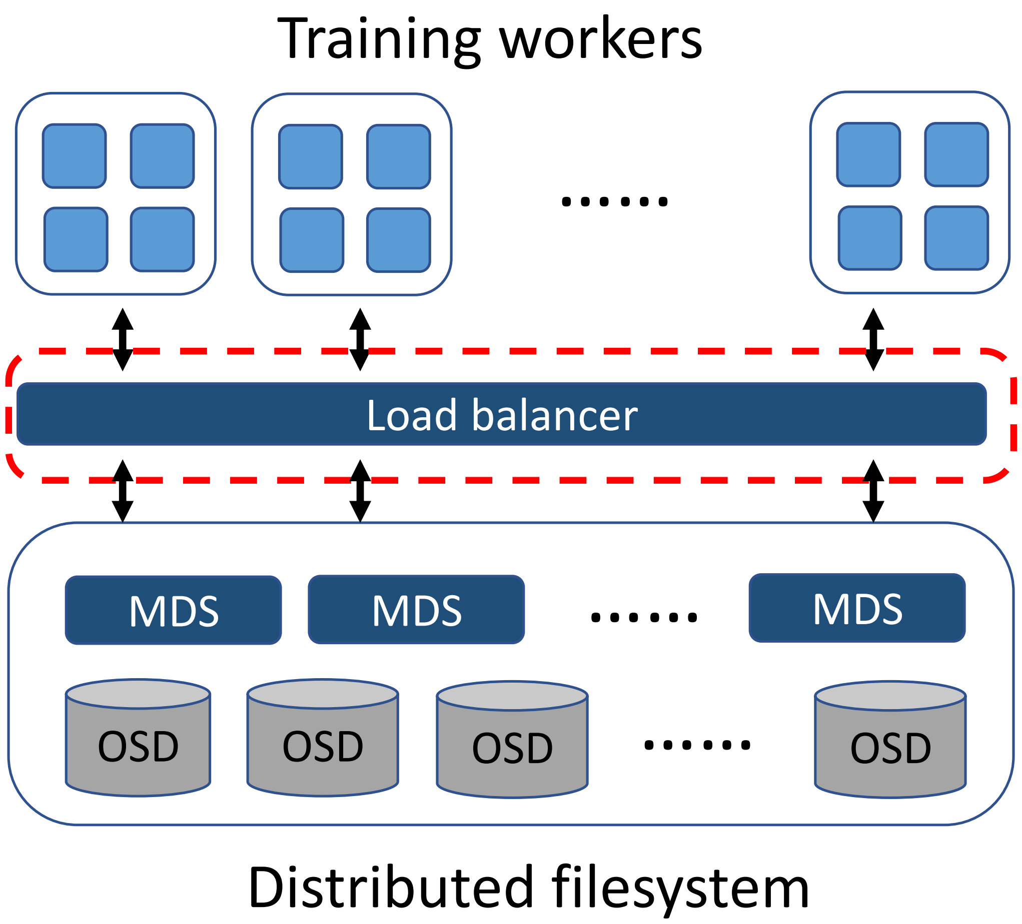
Typical industrial response: Scaling out likewise

- Concerns to be addressed:
  - Cost-effectiveness
  - Scalability
  - Run-time stability



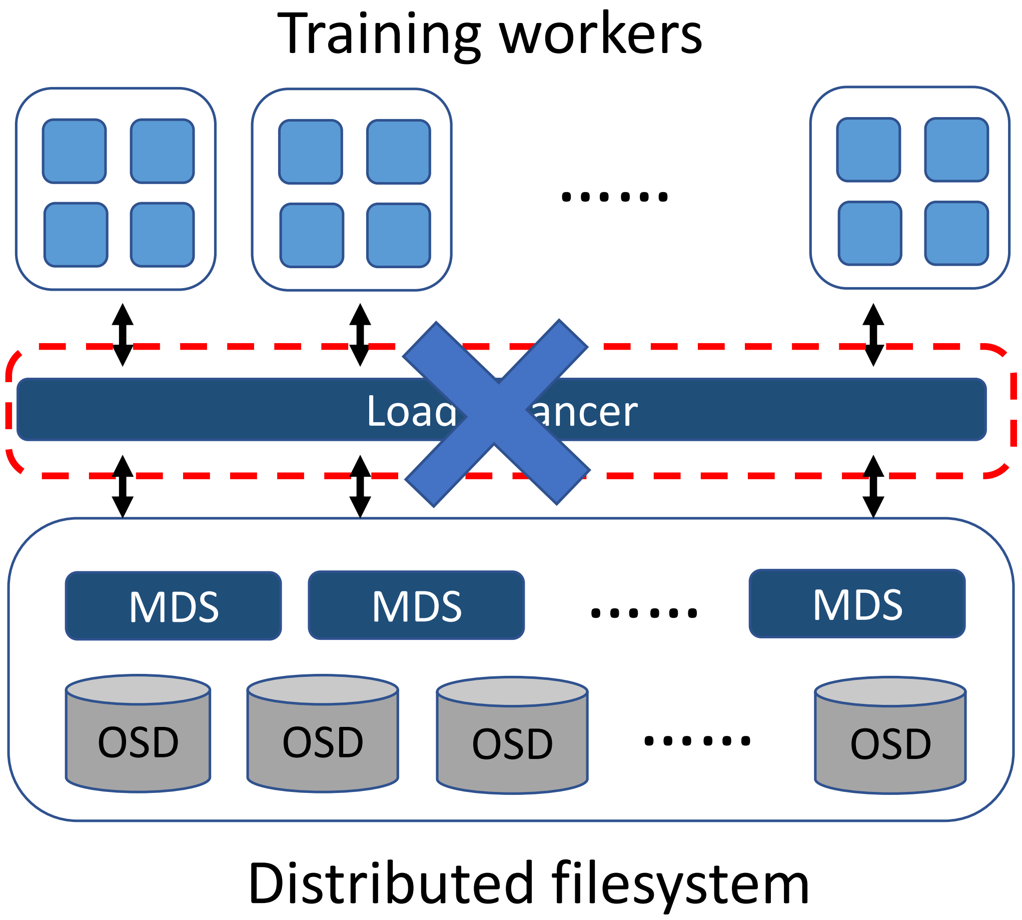
To achieve load-balance…

- A middle layer load-balancer
  - Pros:
    - good global load balancing
    - more features are optional
  - Cons:
    - load-balancer is stressed
    - reintroduce a potential single point of failure
    - not cost-effective



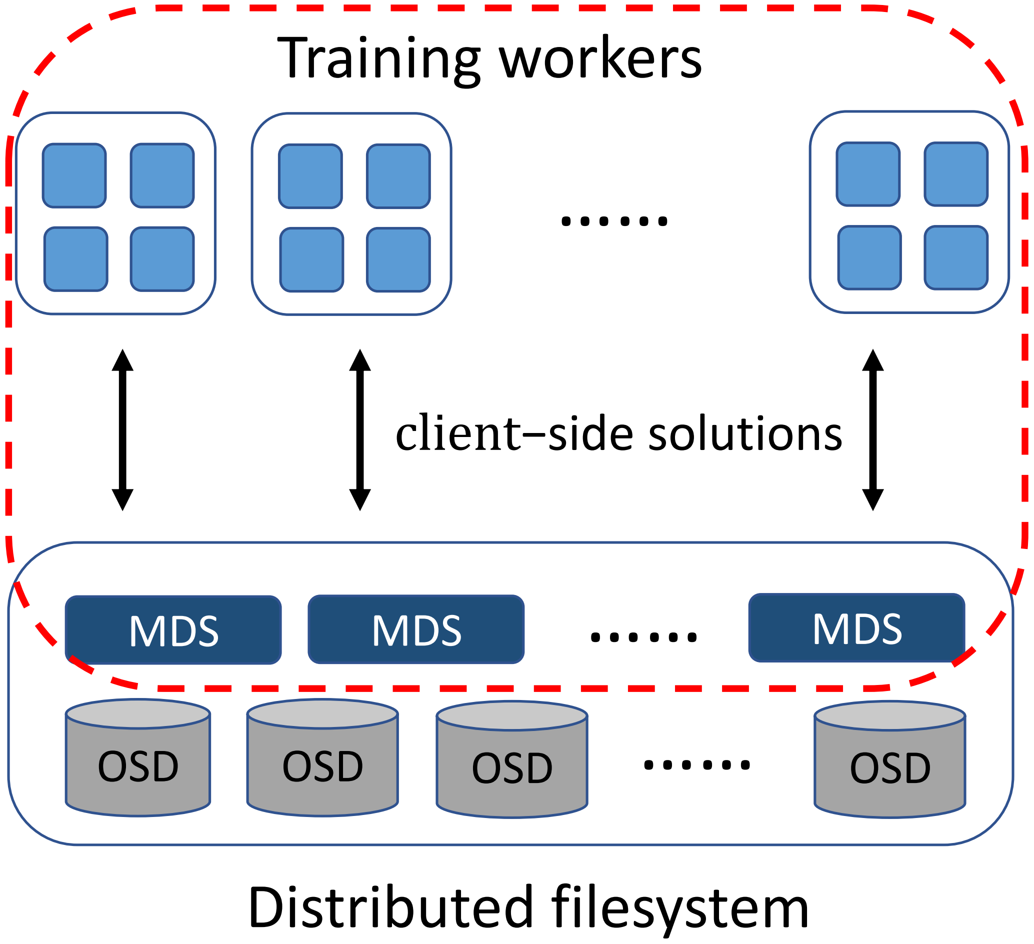
To achieve load-balance…

- A middle layer load-balancer
  - Pros:
    - good global load balancing
    - more features are optional
  - Cons:
    - load-balancer is stressed
    - reintroduce a potential single point of failure
    - not cost-effective



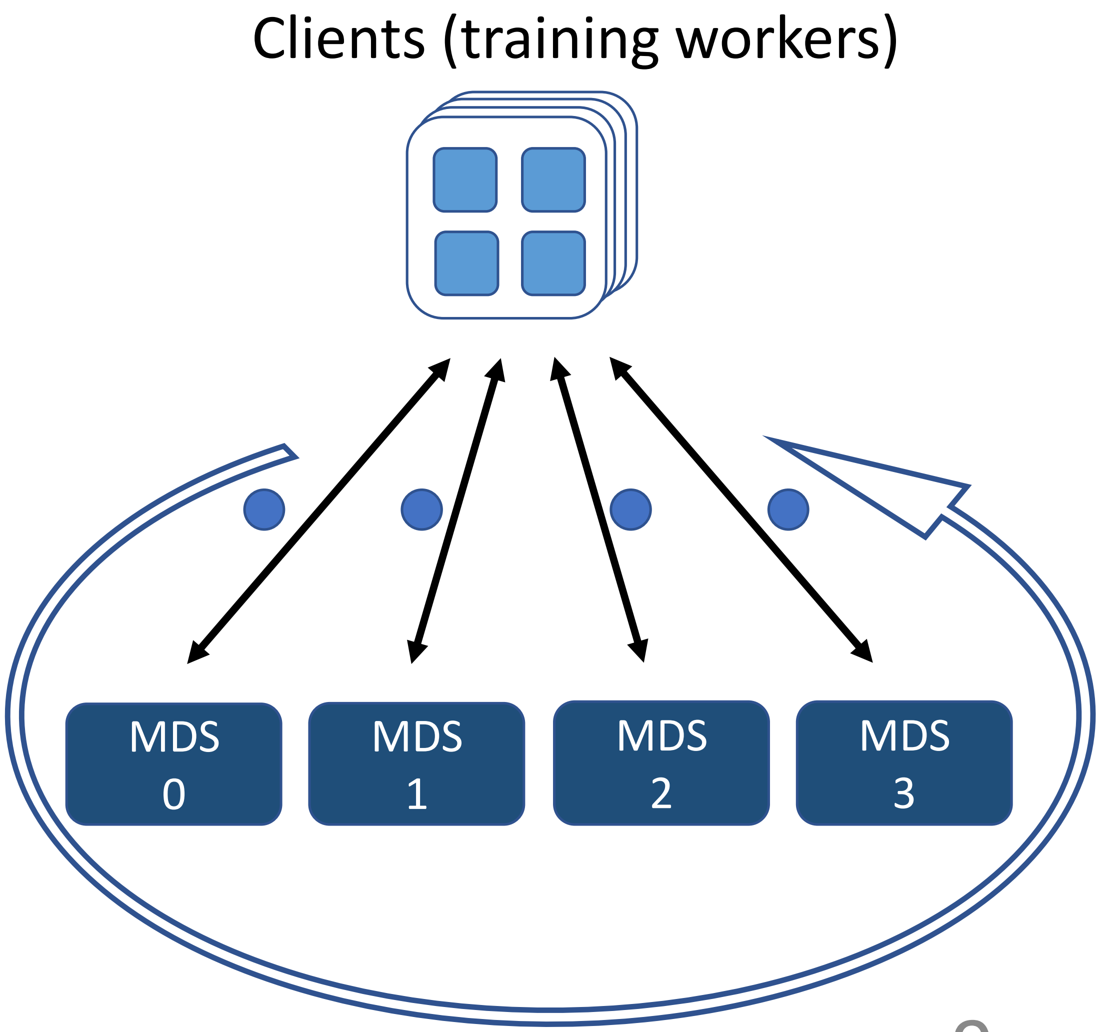
Try client-side solutions

- Easy to implement
- Cost-effective



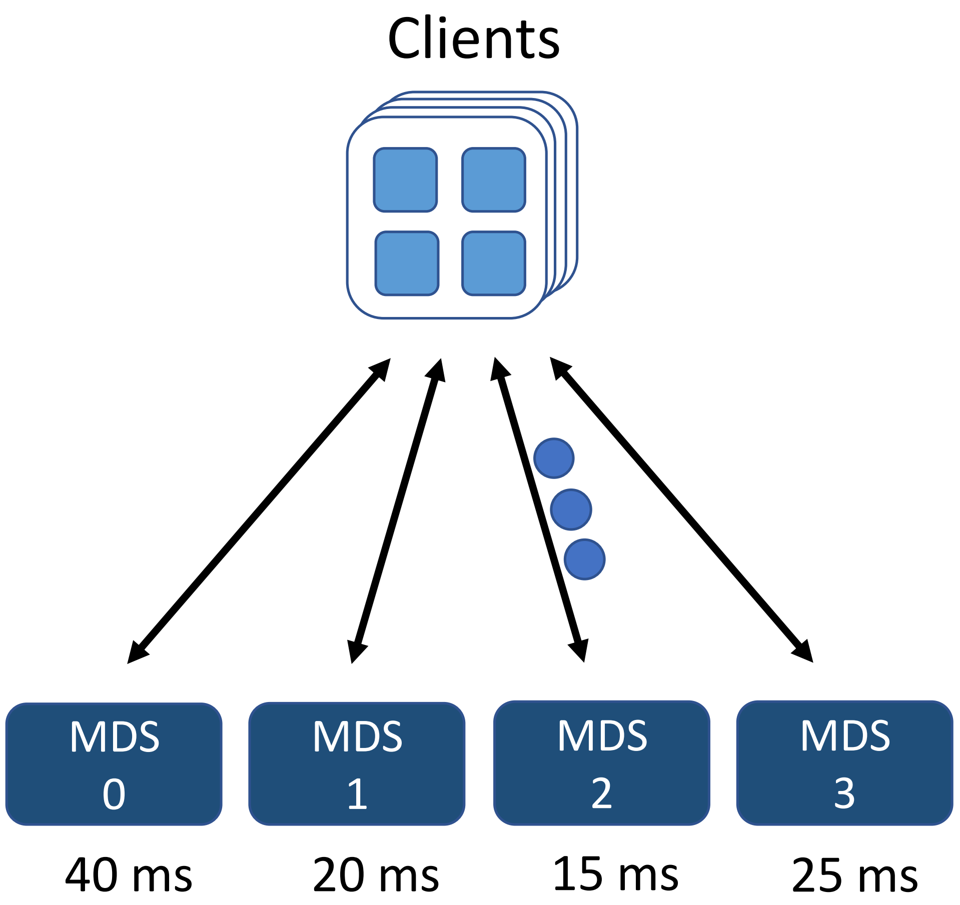
Client-side solution: Round-Robin

- Round-Robin
  - Pros:
    - simple yet effective in homogeneous environments
  - Cons:
    - inflexible and inefficient in shifting or heterogeneous environments



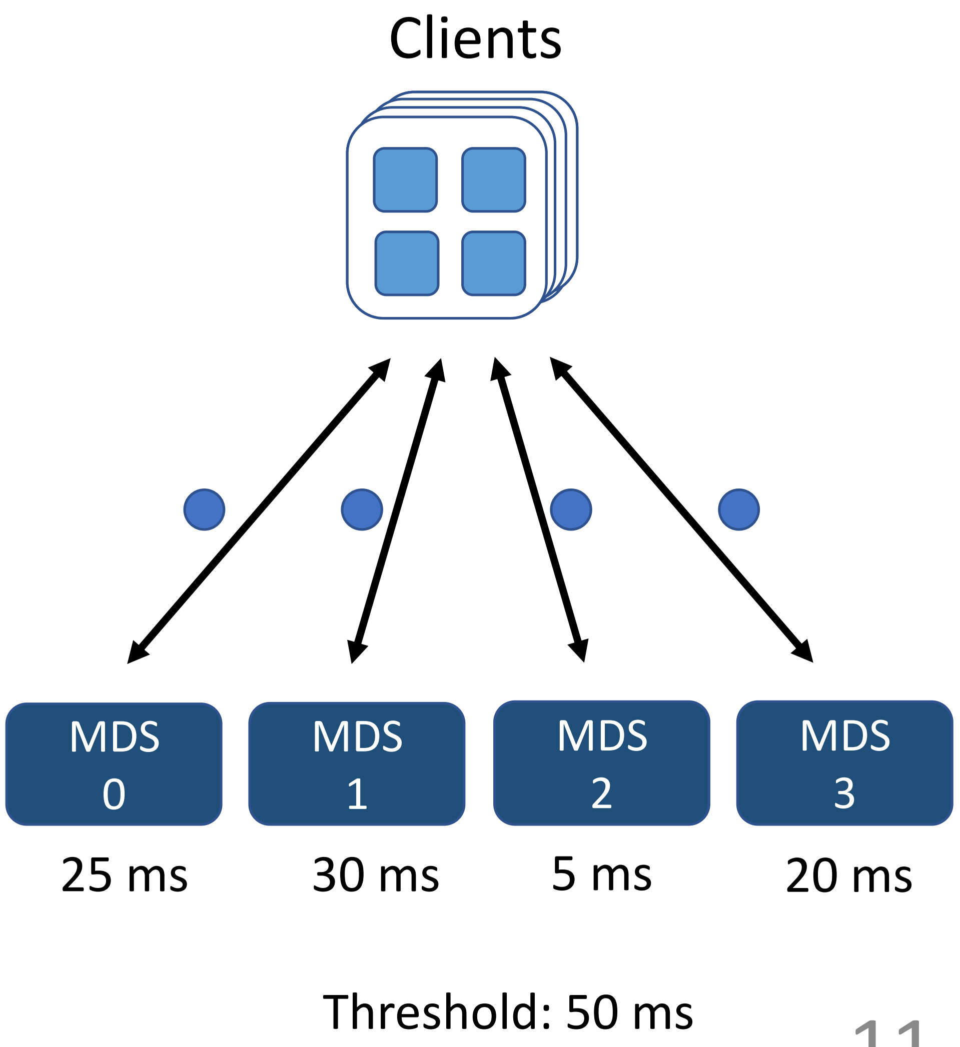
Client-side solution: Heuristic selection

- Heuristic selection
  - e.g., prefer lowest MART (moving average of response time)
  - Pros:
    - effective when facing light-weight workloads
  - Cons:
    - cause herd-behavior and load-oscillations



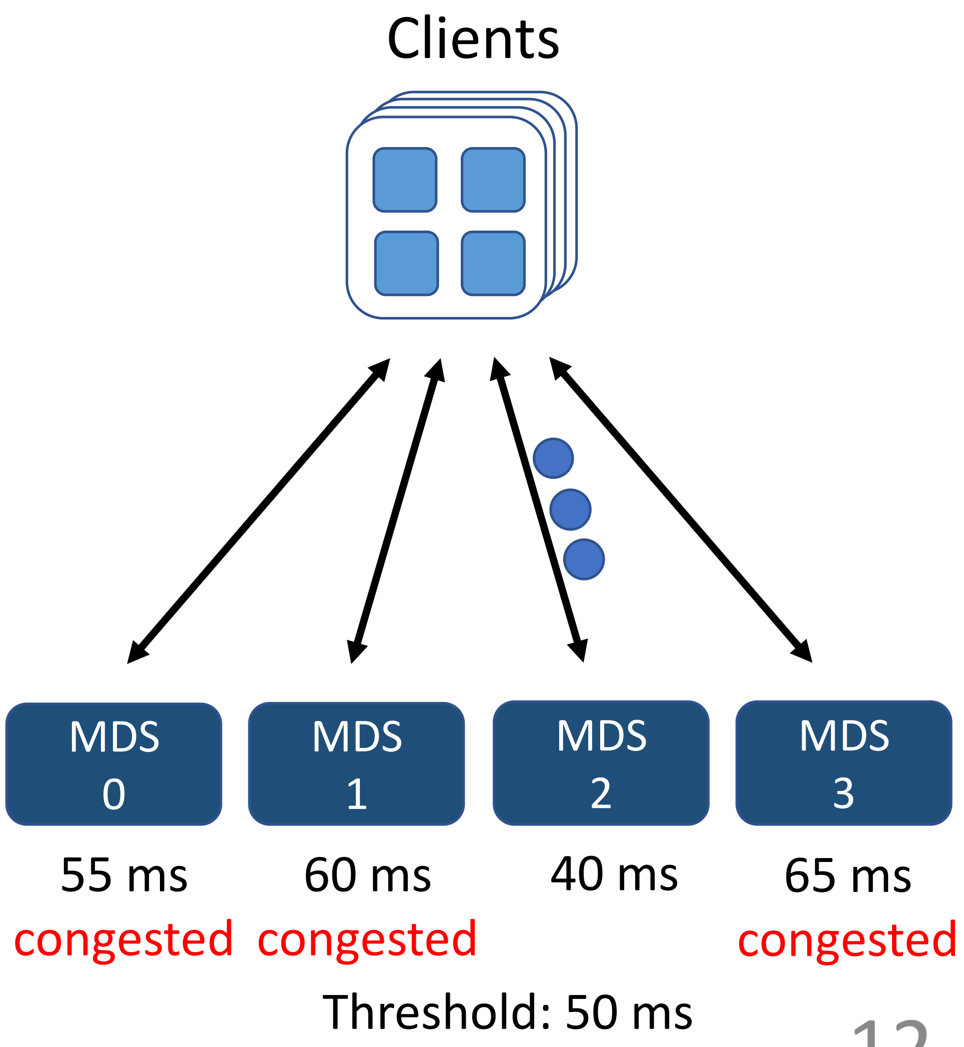
Client-side solution: Round-Robin with Throttling

- Round-Robin with throttling
  - e.g., LADS, preset a MART threshold to mark servers as congested
  - Light-weight workloads
    - = Round-Robin



Client-side solution: Round-Robin with Throttling

- Round-Robin with throttling
  - e.g., LADS, preset a MART threshold to mark servers as congested
  - Light-weight workloads
    - = Round-Robin
  - Heavy workloads
    - = Heuristic selection
    - herd-behavior and load-oscillations remain



CARD: Congestion-Aware Request Dispatching scheme

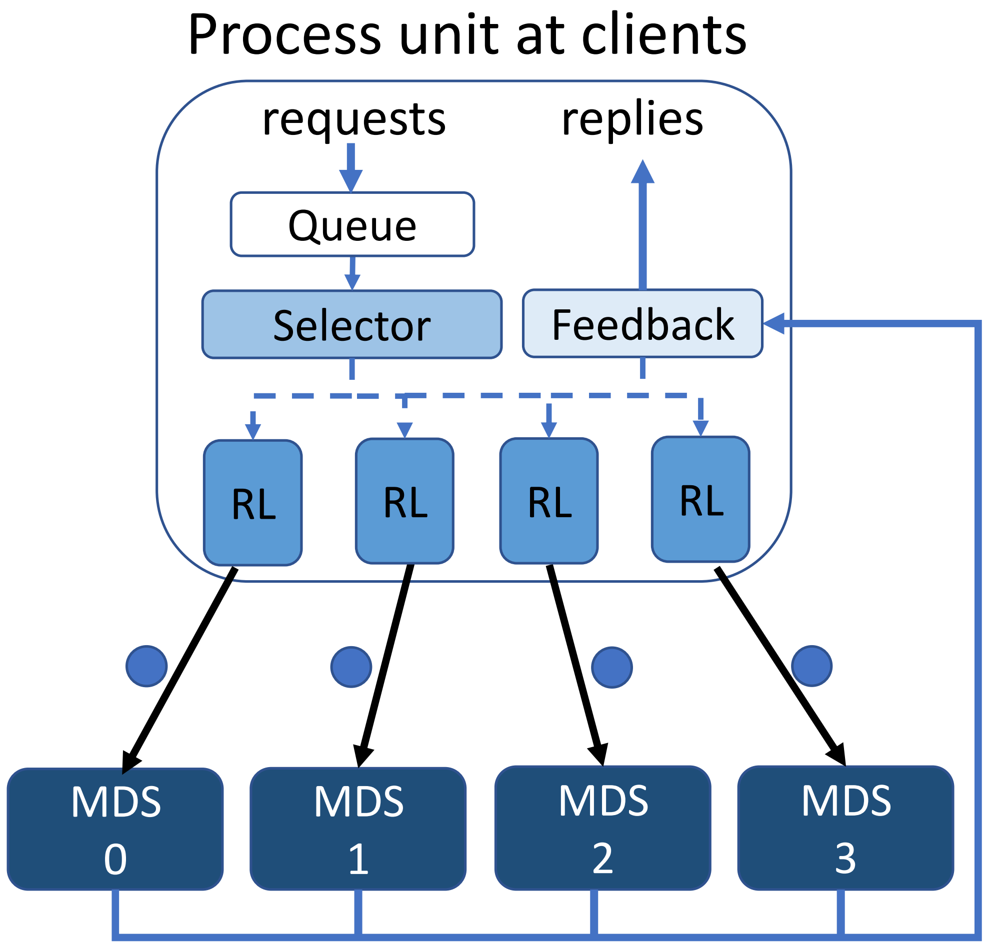
- Core idea: Round-Robin with adaptive rate-control
  - inspired by CUBIC for TCP protocol
  - counting-based implementation
  - no extra info required from servers

- Light-weight workloads
  - = Round-Robin

- Heavy workloads
  - redirect requests from overloaded MDS to underloaded MDS
  - suppress upcoming requests: if and only if all servers are overloaded

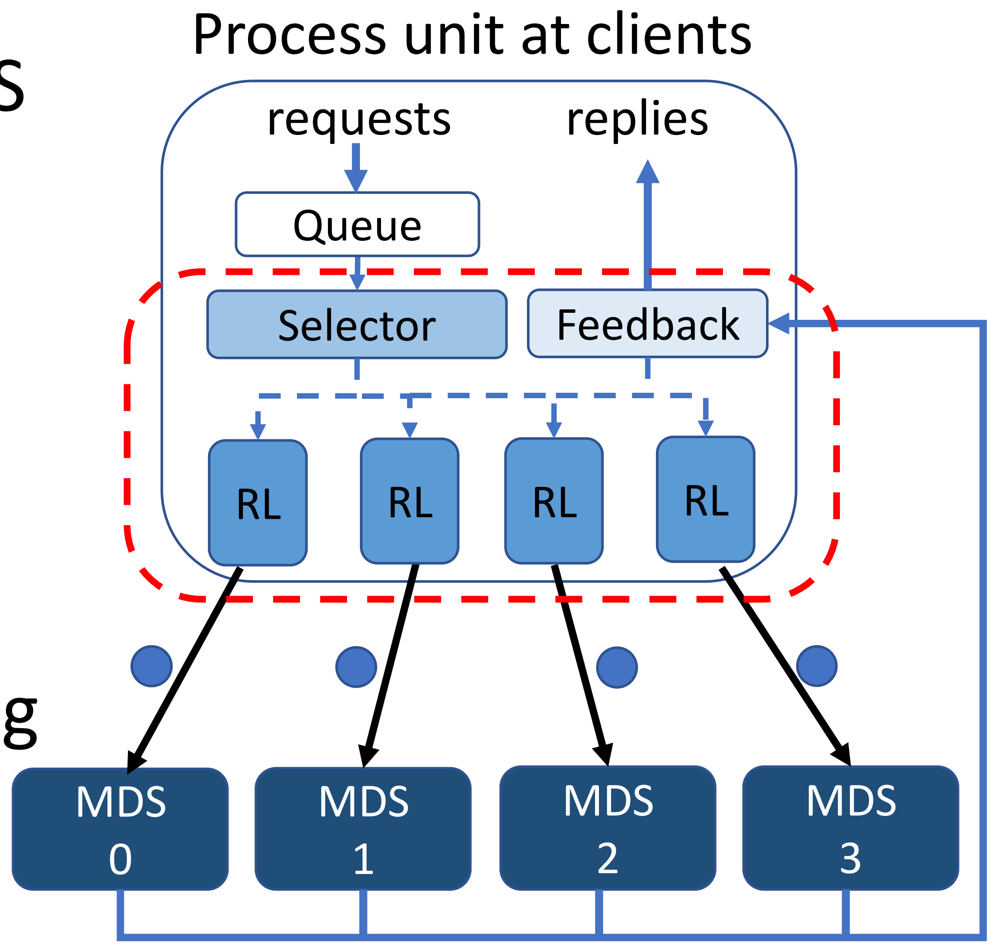
Congestion-aware rate-control mechanism

- Queue: place pending requests
- Selector: Round-Robin dispatching
- Rate-limiter: rate-control module
- Feedback: process feedbacks and forward replies



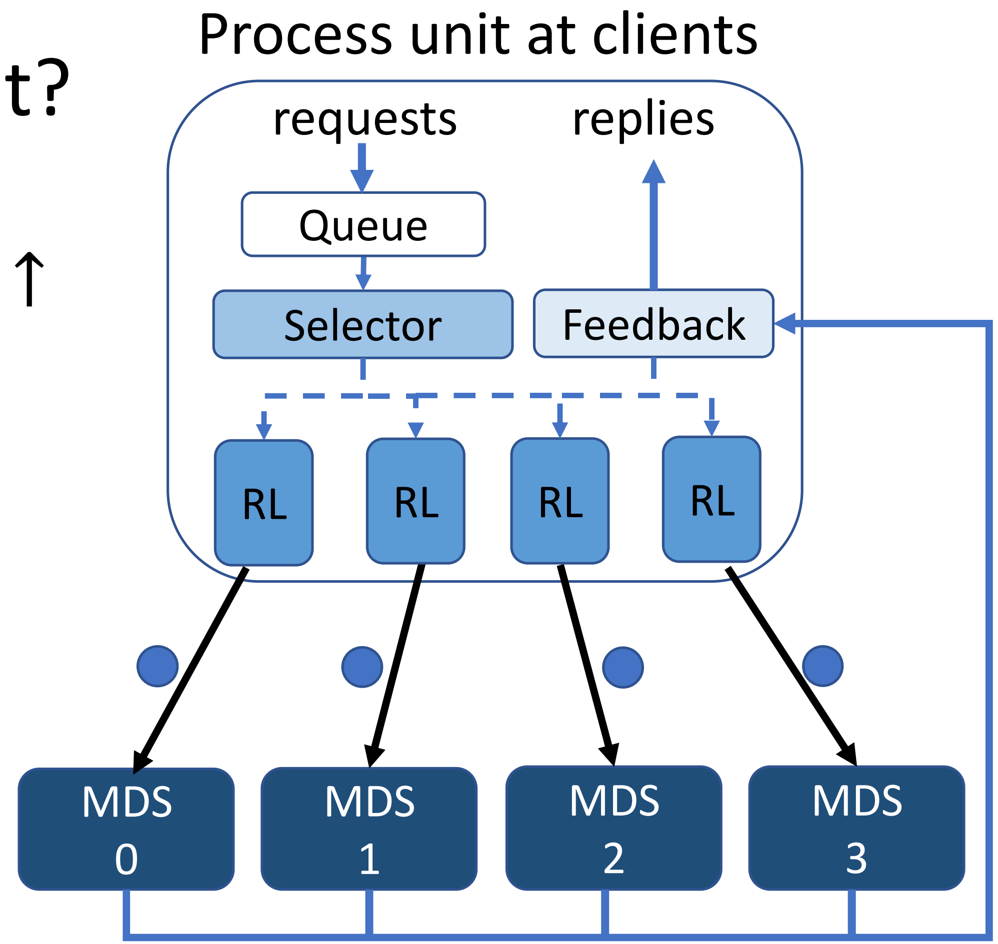
Congestion-aware rate-control mechanism

- Restrict requests routed to each MDS per δ time window
- Gradually increase the restriction according to a cubic growth function
- Feedback module computes receiving rates after each time window and forwards to RLs



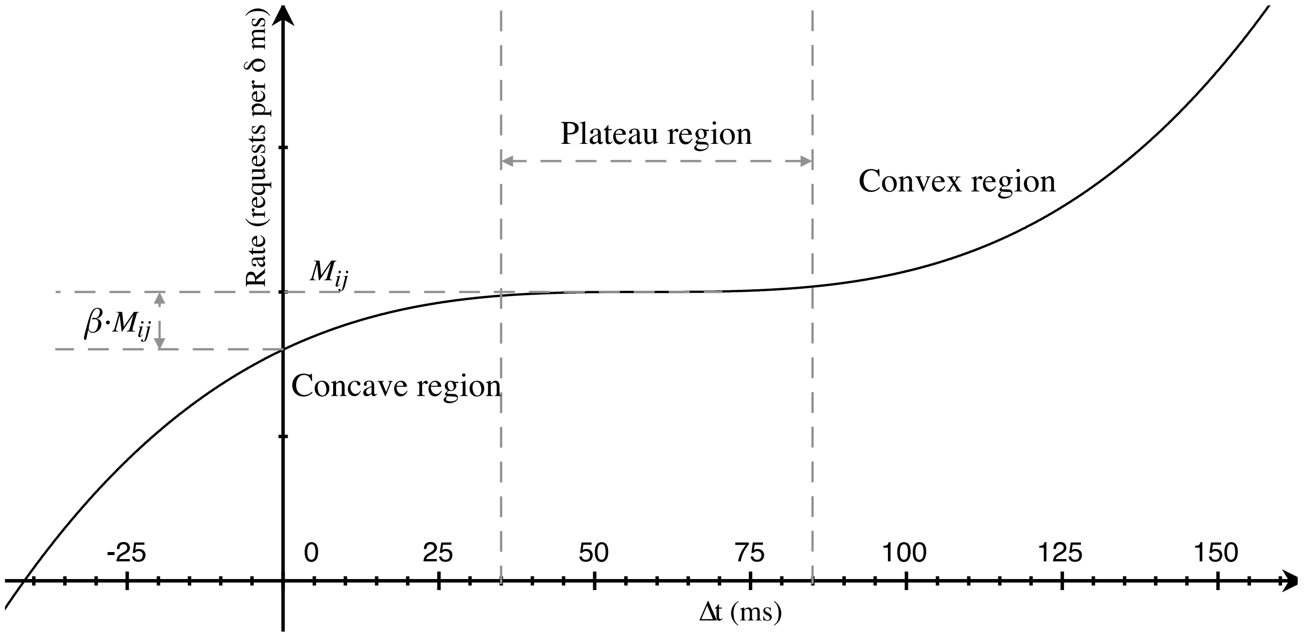
Congestion-aware rate-control mechanism

- How to identify a congestion event?
  - sending rate > receiving rate
  - elapsed time since last sending rate ↑ event > λ (a hysteresis period)
- What to do then?
  - record current sending rate as saturated sending rate
  - reduce current sending rate



The cubic growth function for the rate-control

- Δt: elapsed time since the last congestion event
- M_{ij} : saturated sending rate
  - Changed to current sending rate adaptively whenever a congestion event happens
  - Then, current sending rate reduced to $(1-\beta) \cdot M_{ij}$, and start to grow all over again accordingly



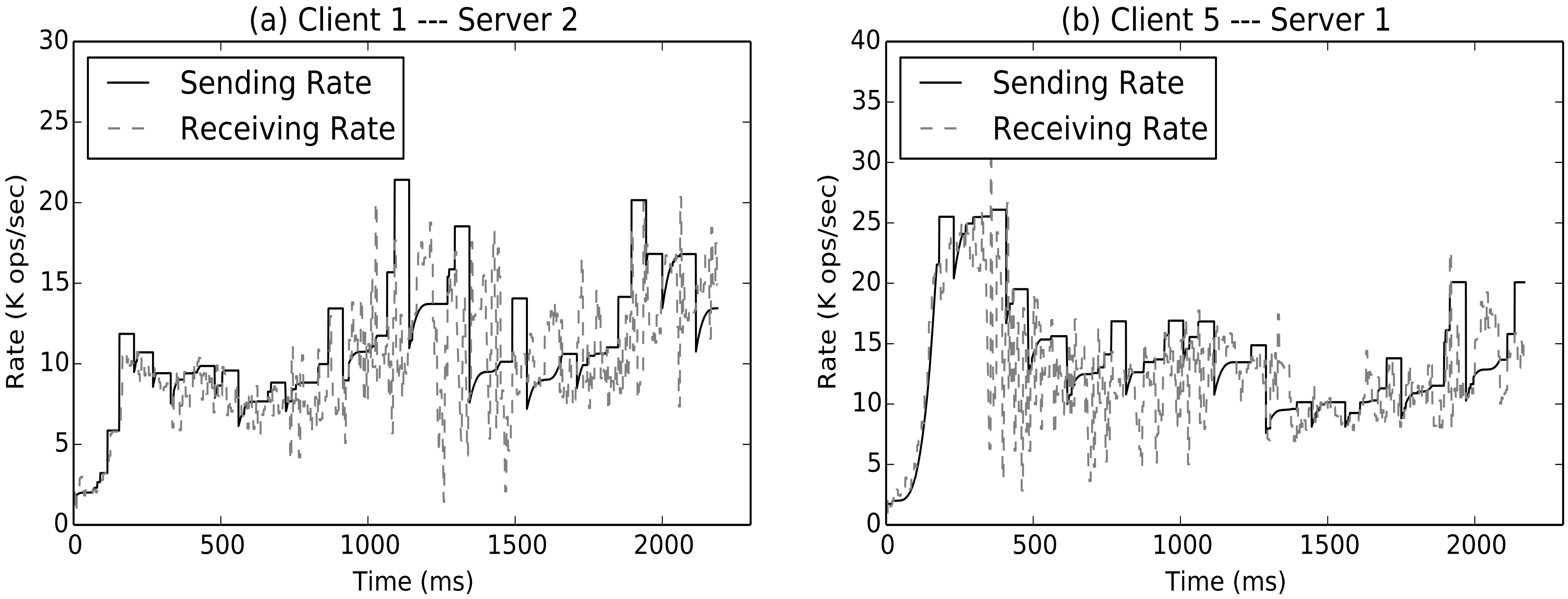
Evaluation setup

- We implemented a prototype RMSC for simulation purposes
- Up to 8 servers to measure system scalability
- Crafted descending setup for heterogeneous experiments
- 10 clients run on separate machines launching request with Poisson arrivals
- δ = 5 ms, λ = 10 ms, β =0.20
- To compare against CARD, we implemented aforementioned Round-Robin, MART and LADS as well
- Refer to the paper for more setup details

Evaluation highlights

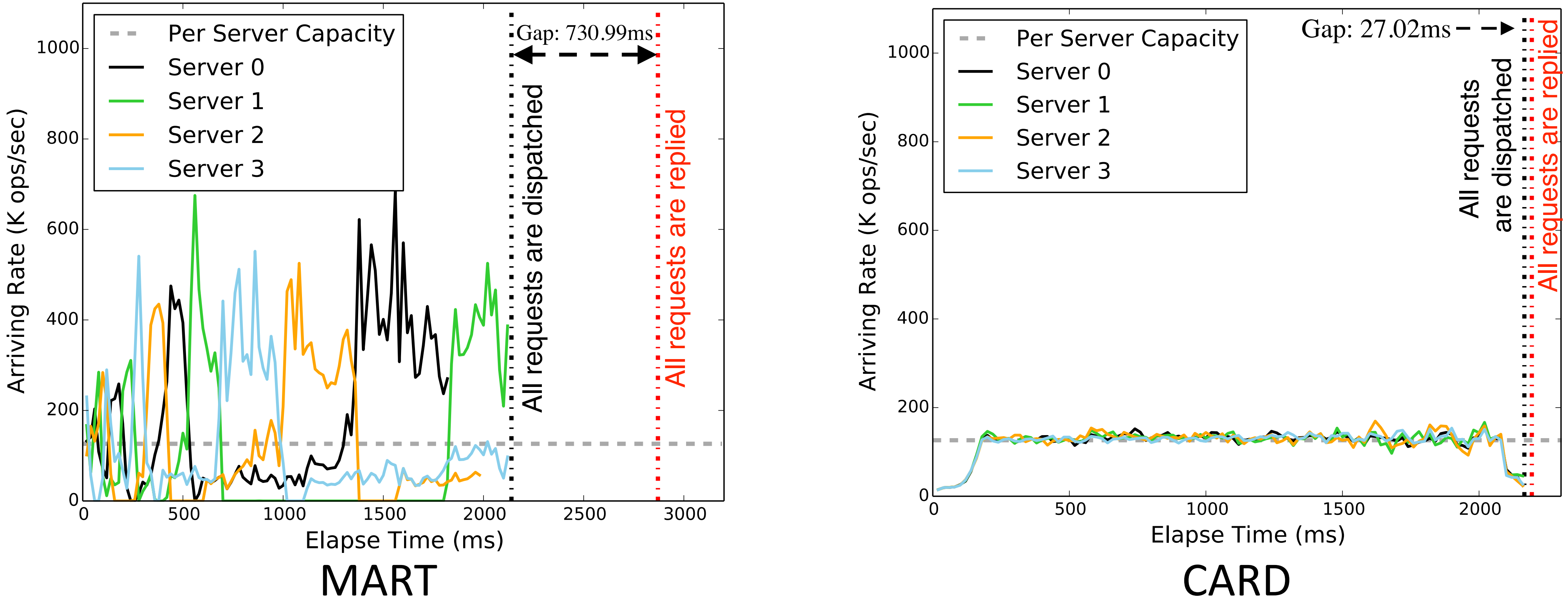
- Do CARD’s rate-control mechanism work as expected?
  - Yes, the rate-control process is effective and adaptive
  - Loads among servers are balanced under heavy workloads
- Can CARD achieve better scalability?
  - In homogeneous clusters: CARD $\approx$ Round-Robin > other strategies
  - In heterogeneous clusters: Yes, CARD > other strategies

Examples of the rate-control procedure



The sending rate from each client to each server is adjusted adaptively according to the receiving rate

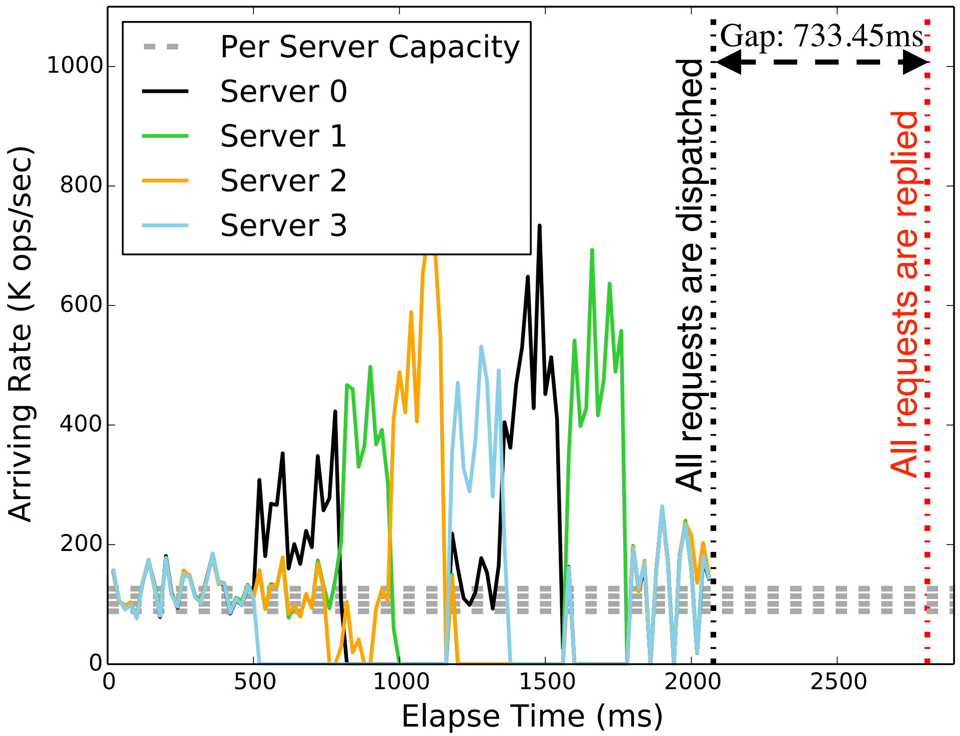
Overall arriving rates in the homogeneous cluster



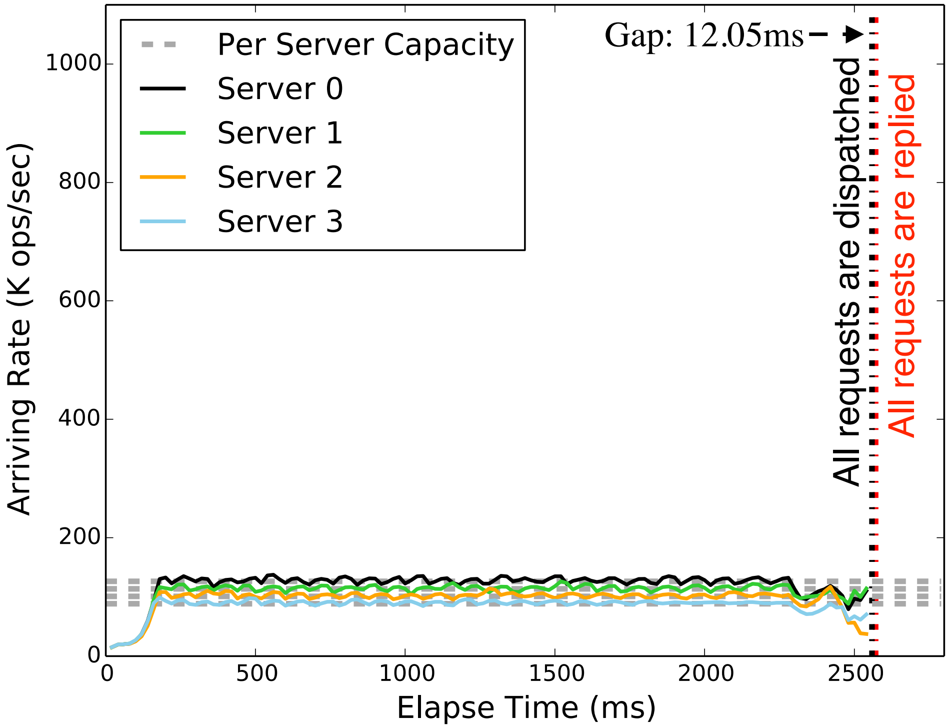
1) Heuristic selections cause severe herd behavior and load-oscillations

2) A data loading job is completed earlier when using CARD

Overall arriving rates in the heterogeneous cluster



LADS



CARD

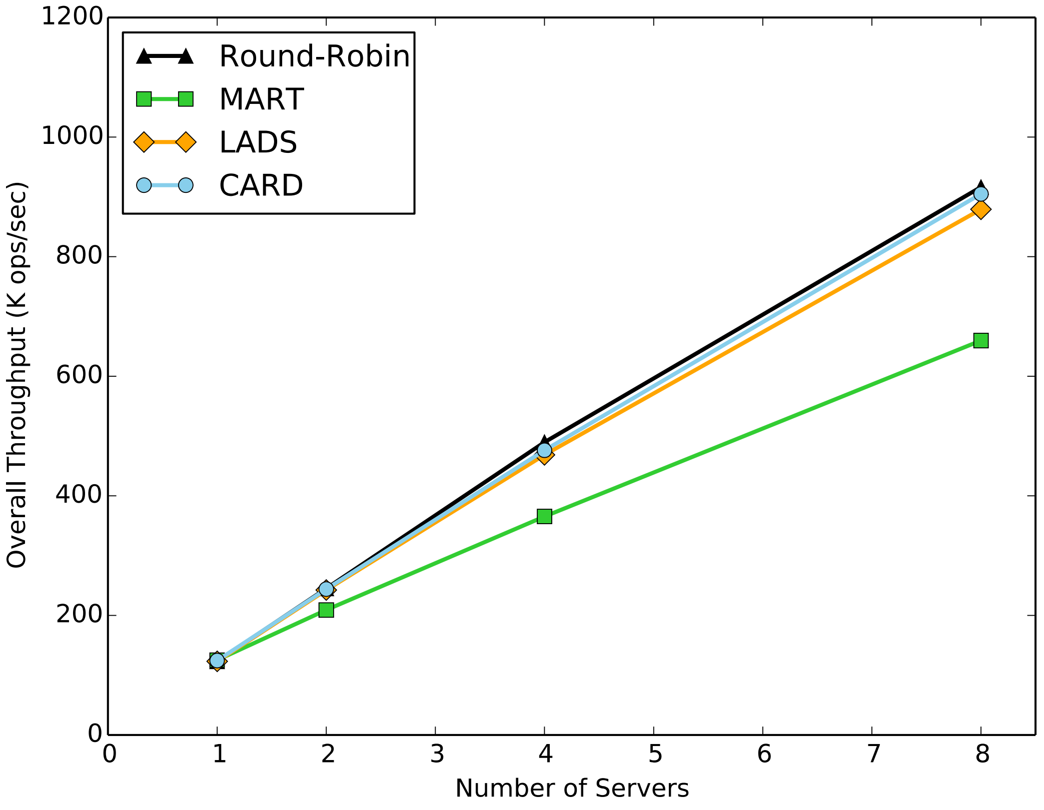
1) A basic threshold throttling strategy is not sufficient enough

2) Arriving rates are stabilized around servers' capacity when using CARD

Overall throughput in the homogeneous cluster

- Heuristic selection is a bad choice under heavy workloads

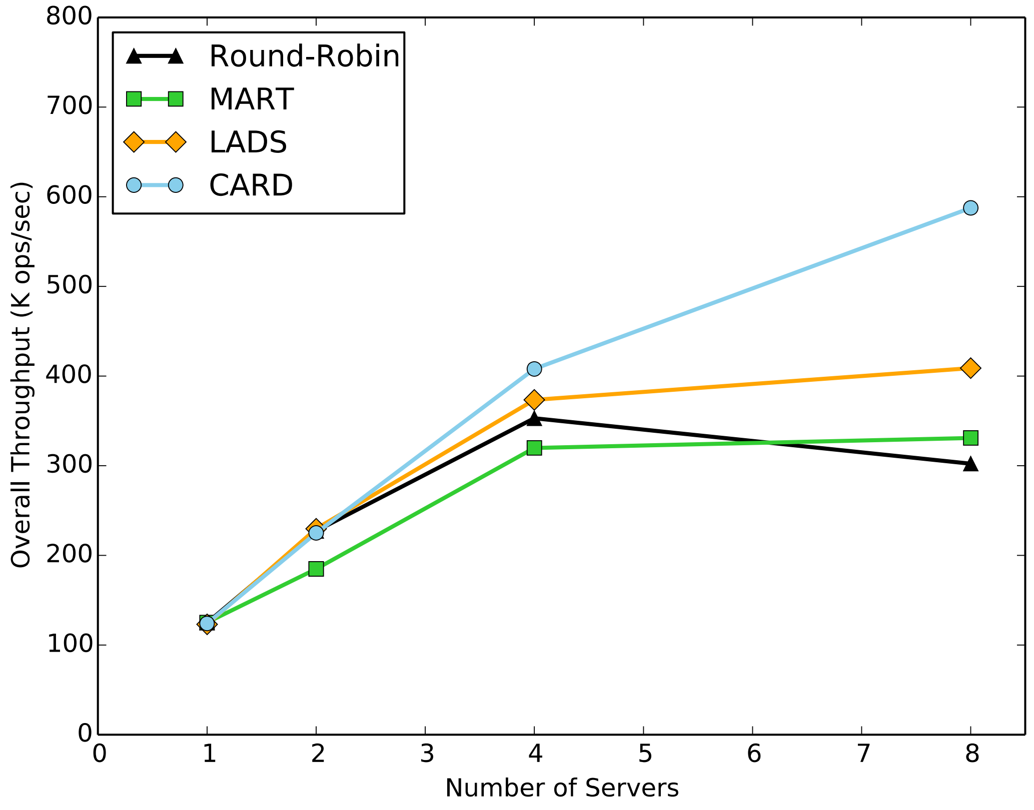
- In ideal homogenous environments, Round-Robin and CARD achieve great scalability



Overall throughput in the heterogeneous cluster

- Round-Robin is ineligible when facing heterogenous setups

- CARD outperforms other strategies and achieves excellent scalability



Summary: CARD

- Adaptive client-side throttling method: easy and efficient
- Redirect requests from the overloaded server to the underloaded server adaptively under heavy workloads
- Degrade into pure Round-Robin when facing light-weight workloads
- Boosts throughput significantly over competing strategies in heterogeneous environments