Fine-Grained TCP Tuning

Amery Hung, Zijian Zhang, Xiaochun Lu

System Technologies & Engineering, ByteDance

ByteDance 字节跳动

Agenda

Background

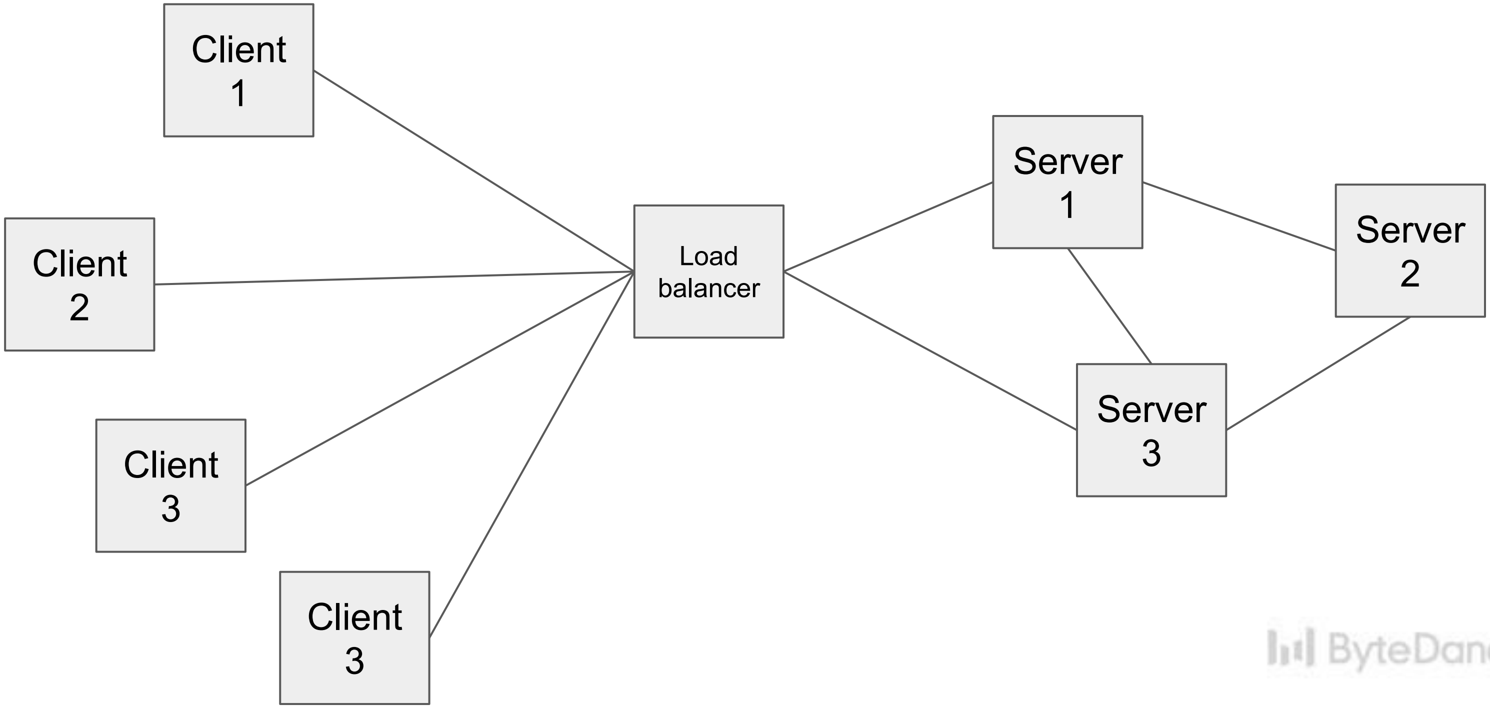
Design

Use case: Adaptive IW tuning

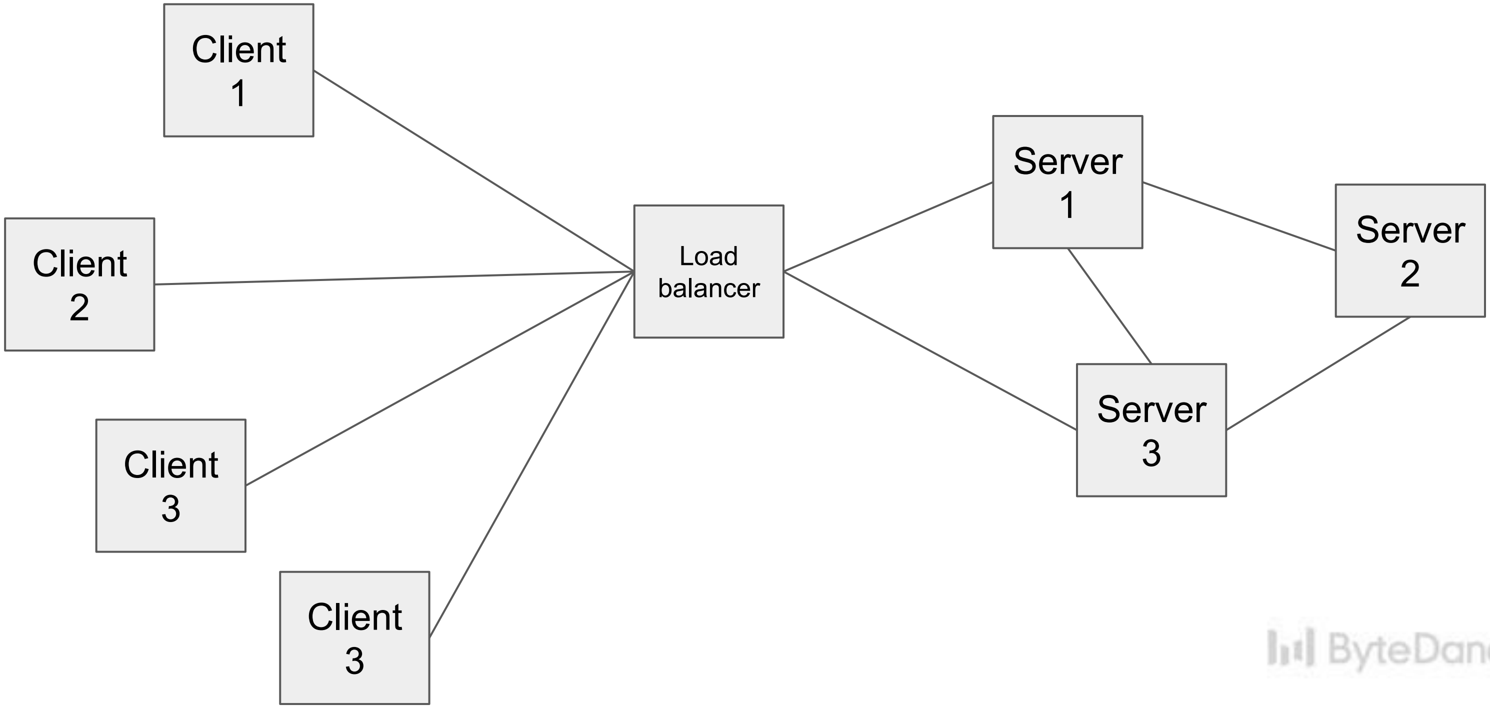
Evaluation

Future work

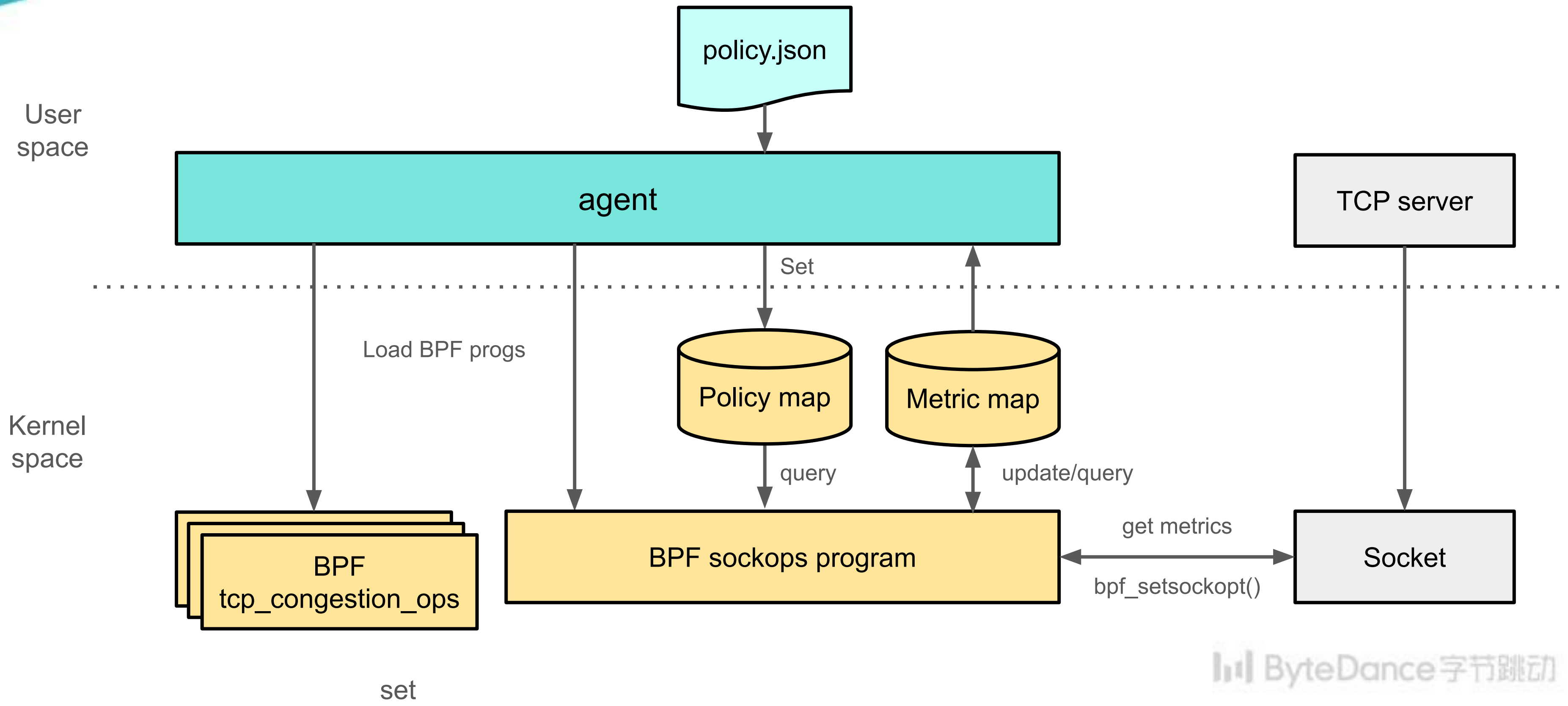
Diverse network environment



One size fit all solution → missed opportunities in performance optimization



Design


policy.json
User space
agent
TCP server
Set
Load BPF progs
Policy map
Metric map
Kernel space
query
update/query
get metrics
BPF tcp_congestion_ops
BPF sockops program
Socket
bpf_setsockopt()
set

Policy

- Features
  - Adaptive IW, Adaptive RTO, Adaptive RWND, Quick ACK, ACK priority, Rate limit detection
- Rate limit
- Congestion control algorithm
  - Reno, Cubic, BBR, DCTCP, PRAGUE, customized CCA…
- Network type
  - Data center, WiFi, Cellular, Long and fat network
- Tuning objective
  - Real time, Latency, Throughput

Metric

Aggregated metric of flows in the same path

- RTT_min minimum RTT detected in the path
- SRTT smooth RTT detected now for the path
- ssthresh slow start threshold
- loss marked lost packet
- BW_max maximum bandwidth discovered in the path
- flow_count # active flows in the path
- tstamp the time the metric last updated

Adaptive initial window (IW)

- It takes time for a flow to ramp up CWND to fully utilize the bandwidth.
- Accelerating slow start by setting initial congestion window close to last detected slow start threshold to fast ramp up throughput.
- Adjust IW based on network condition of path to avoid overshooting.
- Rate pacing from Initial Window to reduce traffic burst

Network metrics for adaptive IW

- RTT_MIN: minimum round trip time detected. it is mainly two way propagation delay zero buffer queue in the network
- SRTT: It is a critical measure of the latency inherent in a network connection. When congestion happens, SRTT will grow significantly
- Use SRTT / RTT_MIN ratio to measure network congestion level
- flow_count: simultaneous high incast may cause packet drop

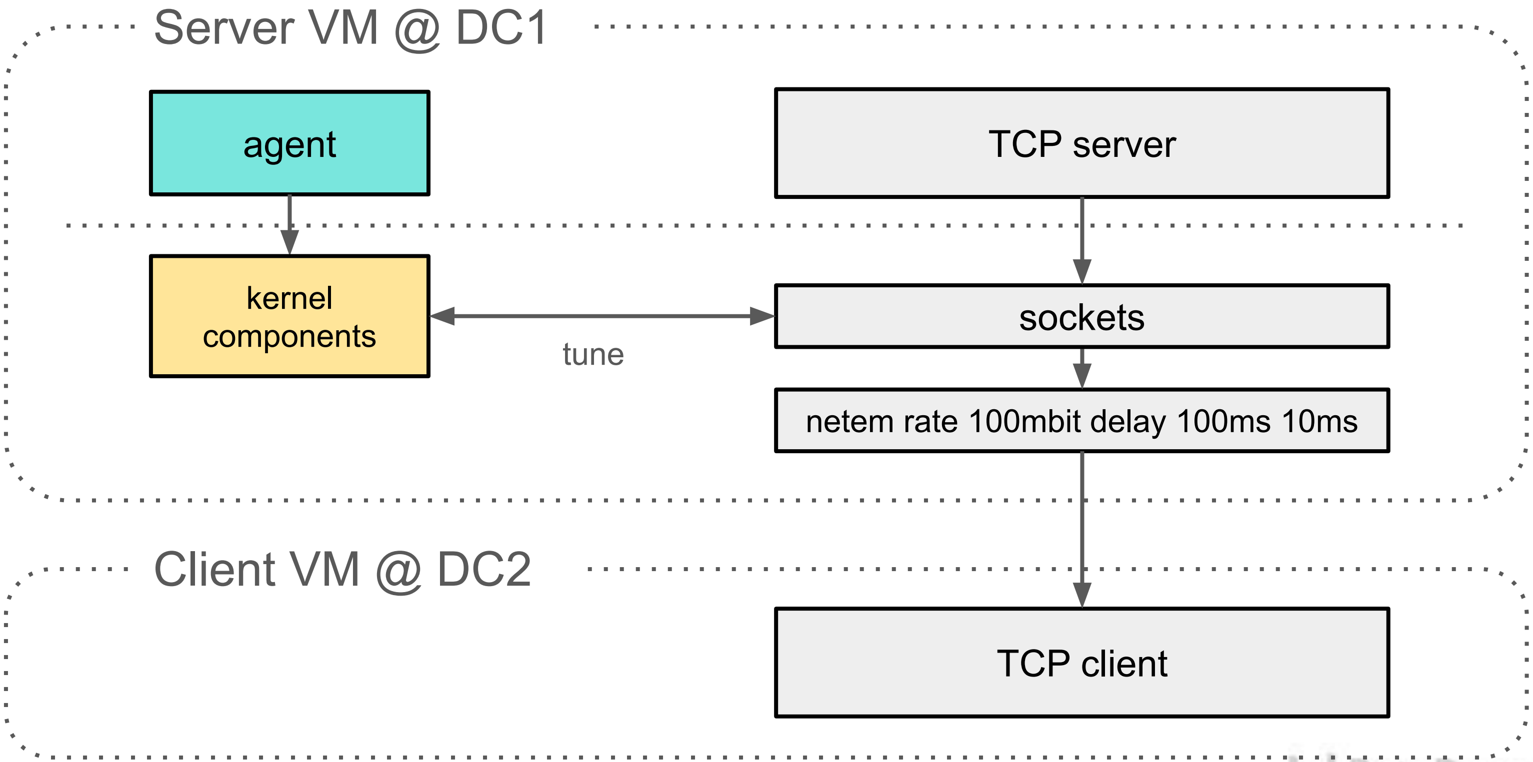
Adaptive initial window (IW)

```
snd_ssthresh = metric->snd_ssthresh / metric->flow_count;

if (metric->rtt_us < metric->rtt_min * 8)
    iw = max(snd_ssthresh / iw_low_load_divisor, iw_init);
else if (metric->rtt_us < metric->rtt_min * 16)
    iw = max(snd_ssthresh / iw_mid_load_divisor, iw_init);
else
    iw = max(snd_ssthresh / iw_high_load_divisor, 1);

bpf_setsockopt(ctx, SOL_TCP, TCP_BPF_IW, &iw, sizeof(iw));
```

Experiment setup

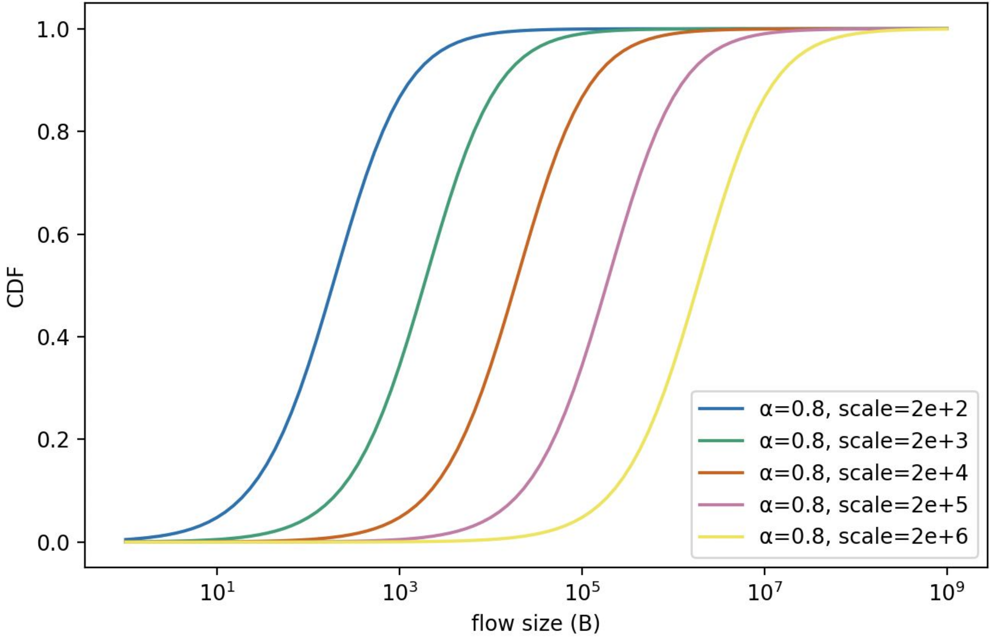


Synthetic workload

- Server sends 1,000 random flows to client
- Flow size follows Pareto distribution
- Varied concurrent flows (5-80)
- 5ms between each flow

Flow	Average flow size (B)
α = 0.8, scale = 2e+2	1 K
α = 0.8, scale = 2e+3	10 K
α = 0.8, scale = 2e+4	100 K
α = 0.8, scale = 2e+5	1 M
α = 0.8, scale = 2e+6	10 M

Average flow size (B)

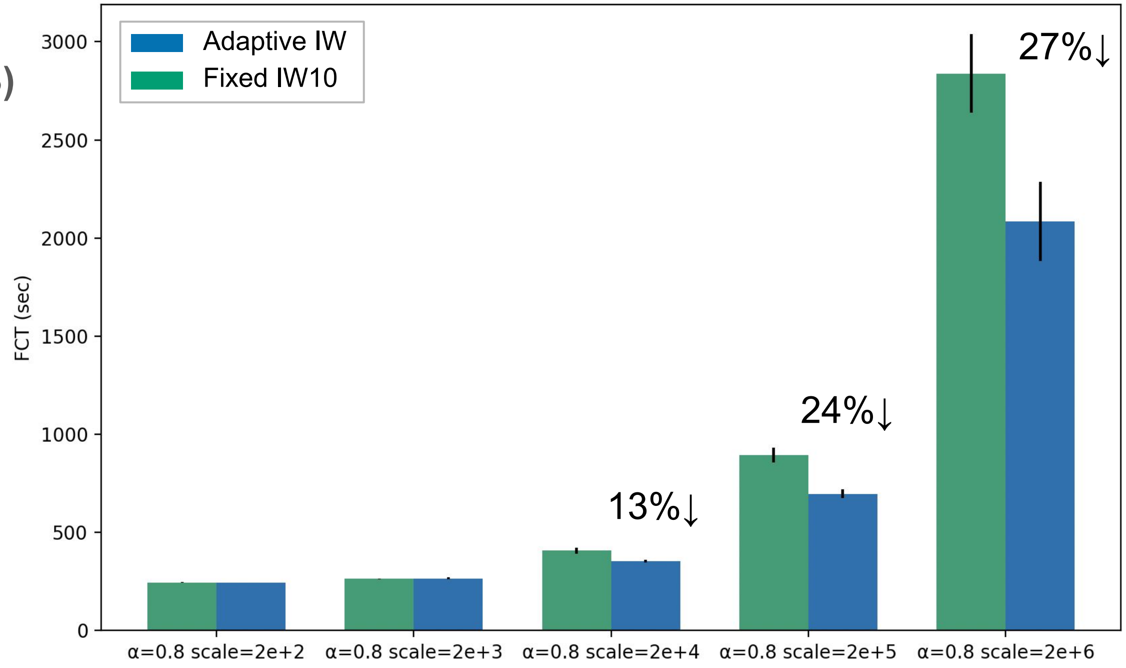


Assumption and limitation

- Assuming the network condition does not change in a short period of time
- Assuming no packet loss
- Potential change in the network condition across time

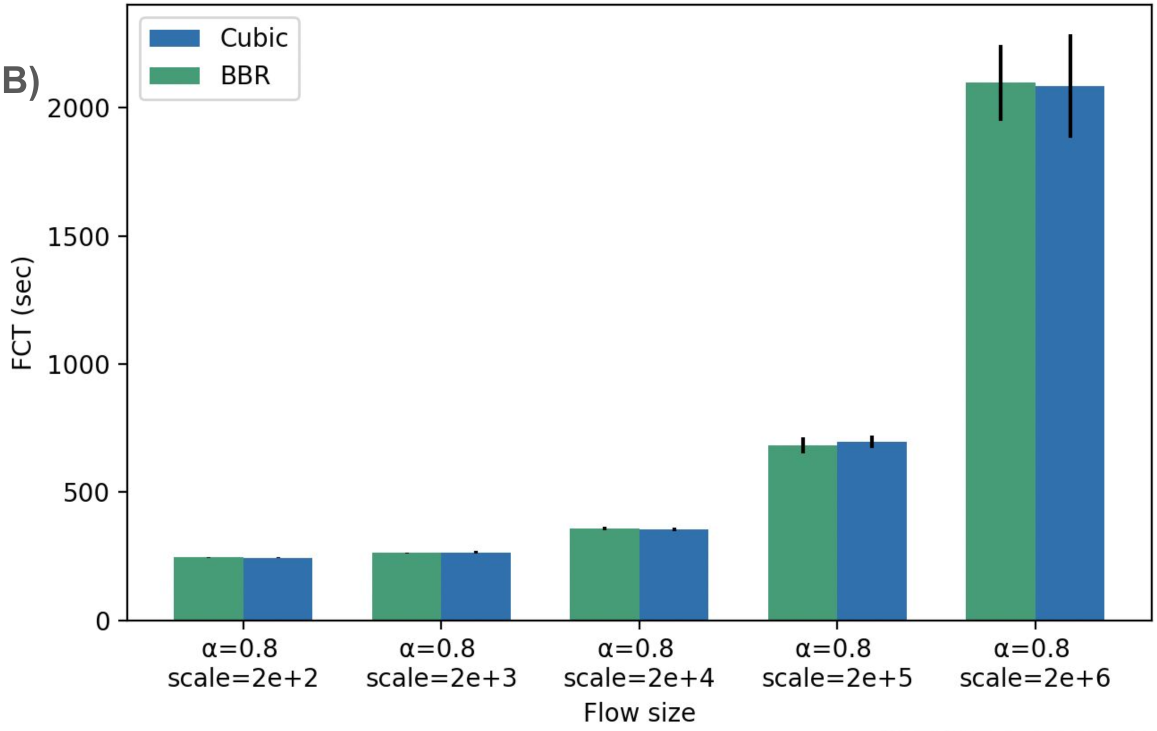
Overall throughput is improved for larger flow size

- **Varied flow size:**
  (Avg flow size = 1KB-10MB)
- Total flow: 1000
- Incast: 5
- Each repeated for 5 times
- Compare overall FCT between adaptive IW with fixed IW10



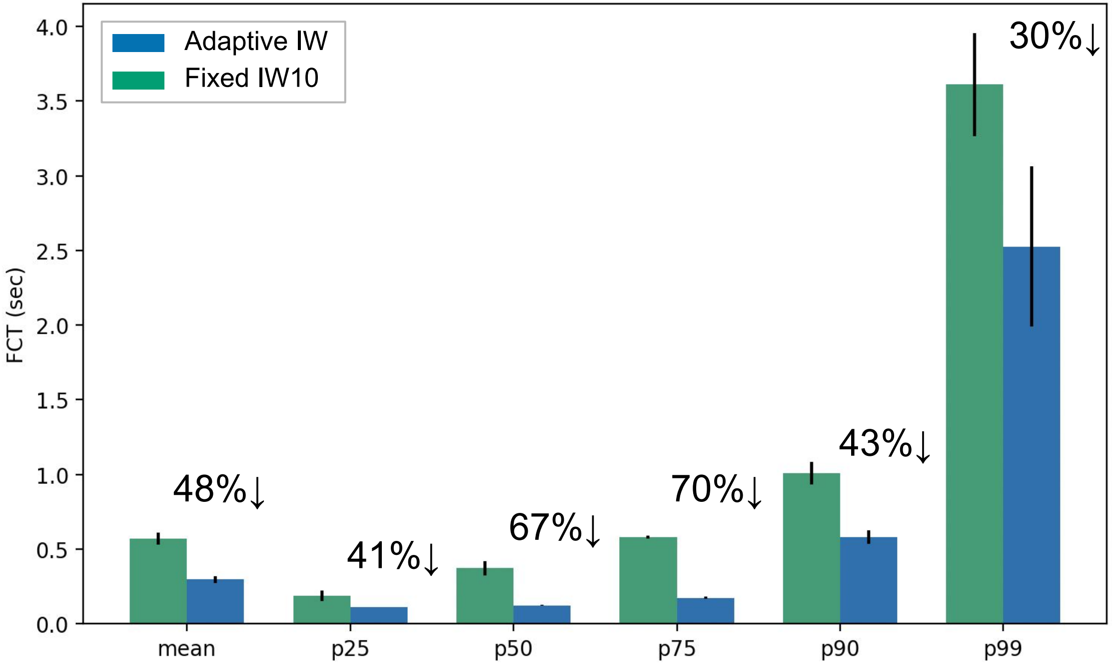
Adaptive IW works with different CCA

- **Varied flow size:**
  (Avg flow size = 1KB-10MB)
- Total flow: 1000
- Incast: 5
- Each repeated for 5 times
- Compare overall FCT between BBR and Cubic



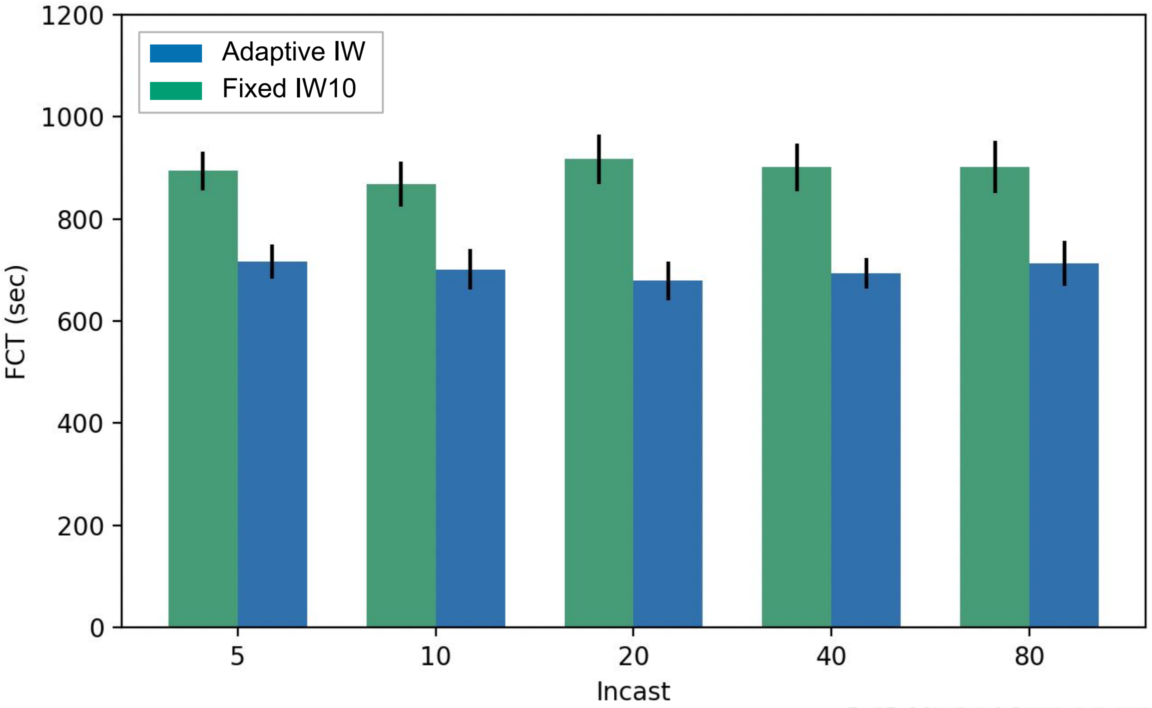
Adaptive IW improves flow completion time (FCT)

- Flow size:
  α = 0.8, scale = 2e+5
  (Avg flow size = 1MB)
- Total flow: 1000
- Incast: 5
- Each repeated for 5 times
- Compare tail FCT between adaptive IW with fixed IW10



Adaptive IW scales with incast

- Flow size:
  α = 0.8, scale = 2e+5
  (Avg flow size = 1MB)
- Total flow: 1000
- **Varied incast: 5-80**
- Each repeated for 5 times
- Compare tail FCT between adaptive IW with fixed IW10



Future work

- Test in production environment
- Customized CC in ebpf to allow more tuning
- Experiment with more tunings

Adaptive Receiving Window (RWND)

Optimize TCP receive window size based on Bandwidth-Delay Product (BDP) and total flow count

BDP: Calculate the BDP to understand the potential data in transit.

Normalization: Adjust the BDP by the total flow counter to ensure fair resource allocation among all active flows.

BDP = RTT * LINK_MBPS

G = F(flow_counter)

RWND_INIT= MAX(Beta * BDP / (MSS * G), 4)

Slow Start Threshold Detection

Utilize CC algorithms previously detected ssthresh for future reference.

Challenges & Solutions:

1. **Underestimation in Short Flows**
   - **Problem:** Short flows can lead to underestimated `ssthresh`.
   - **Solutions:**
     1. Implement a high pass filter with a minimum value.
     2. Use the maximum `ssthresh` value probed in the last 30 seconds.
2. **Overestimation Leads to High Retransmissions**
   - **Problem:** Overestimation can result in increased retransmissions.
   - **Solution:** Adjust `ssthresh` based on current network conditions using a loader divider.