# CS 795: Distributed Systems & Cloud Computing Fall 2018

Lec 7: Container registry analysis

Yue Cheng

# Final presentation poll

Plan to move discussion of Paper16 (RAMCloud) from Nov 28 to Dec 5

- So we will have two paper discussions on Dec 5 (the other will be led by me: Alibaba workload analysis)

Thinking about moving the final presentation to Dec 14 (2nd Friday of December)

- Roughly 3 hours for 7 teams
- Format: Pizza + talks

# Docker usage patterns remain a mystery

- How are Docker containers used and managed at scale?
- How can we streamline Docker workflows?
- How do we facilitate Docker performance analysis

# So what we did…

- Conduct a large-scale analysis of a real-world production Docker workload from geo-distributed IBM container service

- Provide insights and develop heuristics to improve Docker workflow performance

- Develop an open source Docker workflow analysis tool

FAST '18

*** Anwar et al.: Improving Docker Registry Design based on Production Workload Analysis**

# I will briefly talk about…

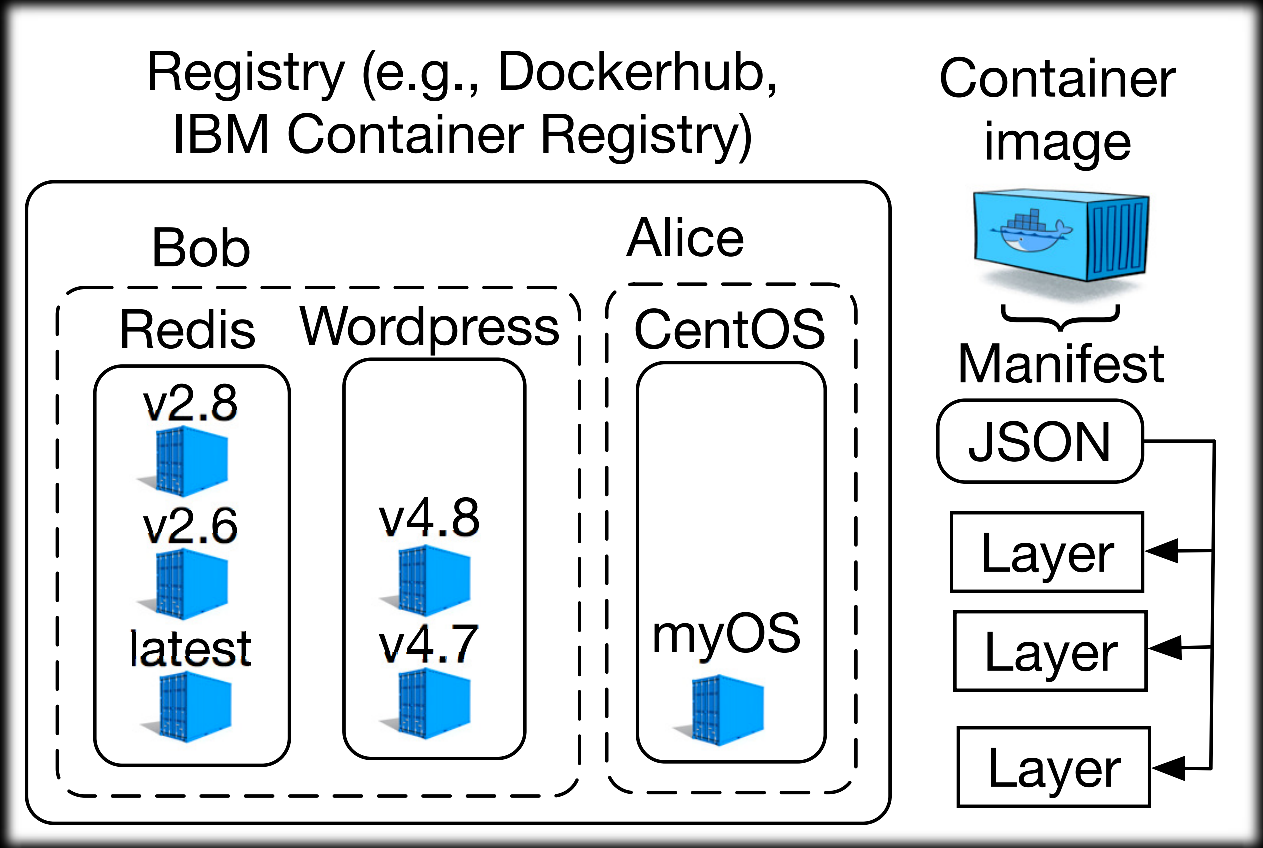
- **Conduct a large-scale analysis of a real-world production Docker workload from geo-distributed IBM container service**
- Provide insights and develop heuristics to improve Docker workflow performance
- Develop an open source Docker workflow analysis tool

FAST'18

*** Anwar et al.: Improving Docker Registry Design based on Production Workload Analysis**

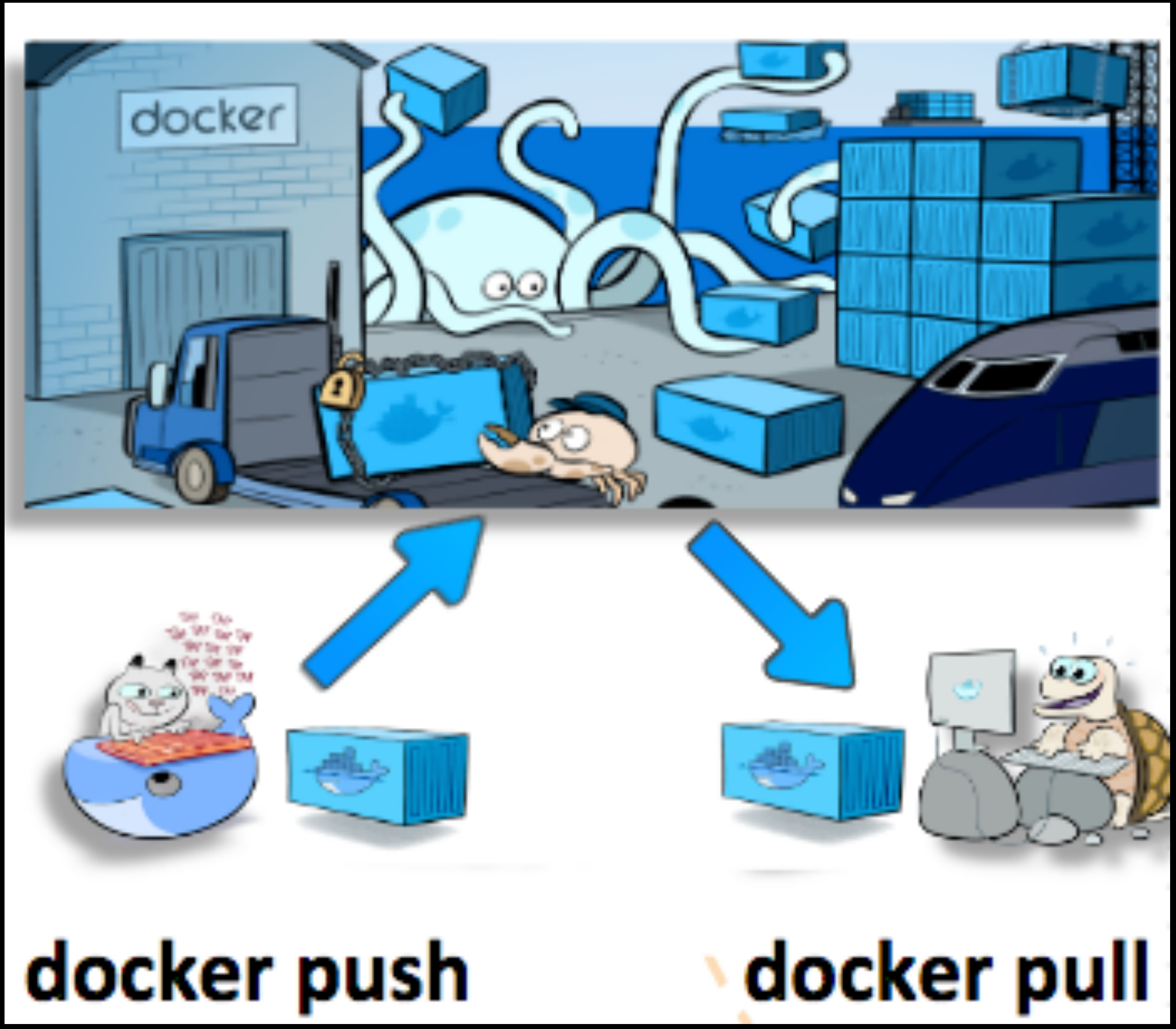
# Background: Docker container image

- Container images are divided into layers
- The metadata file is called manifest (JSON-formatted)
- Users create repositories to store images
- Images in a repository can have different tags (versions)



# Background: Docker container registry

- Docker container images are stored online in Docker Registry
- Push image:
  - HEAD layers
  - POST/PUT/PATCH layer
  - PUT manifest
- Pull image:
  - GET manifest
  - GET layers



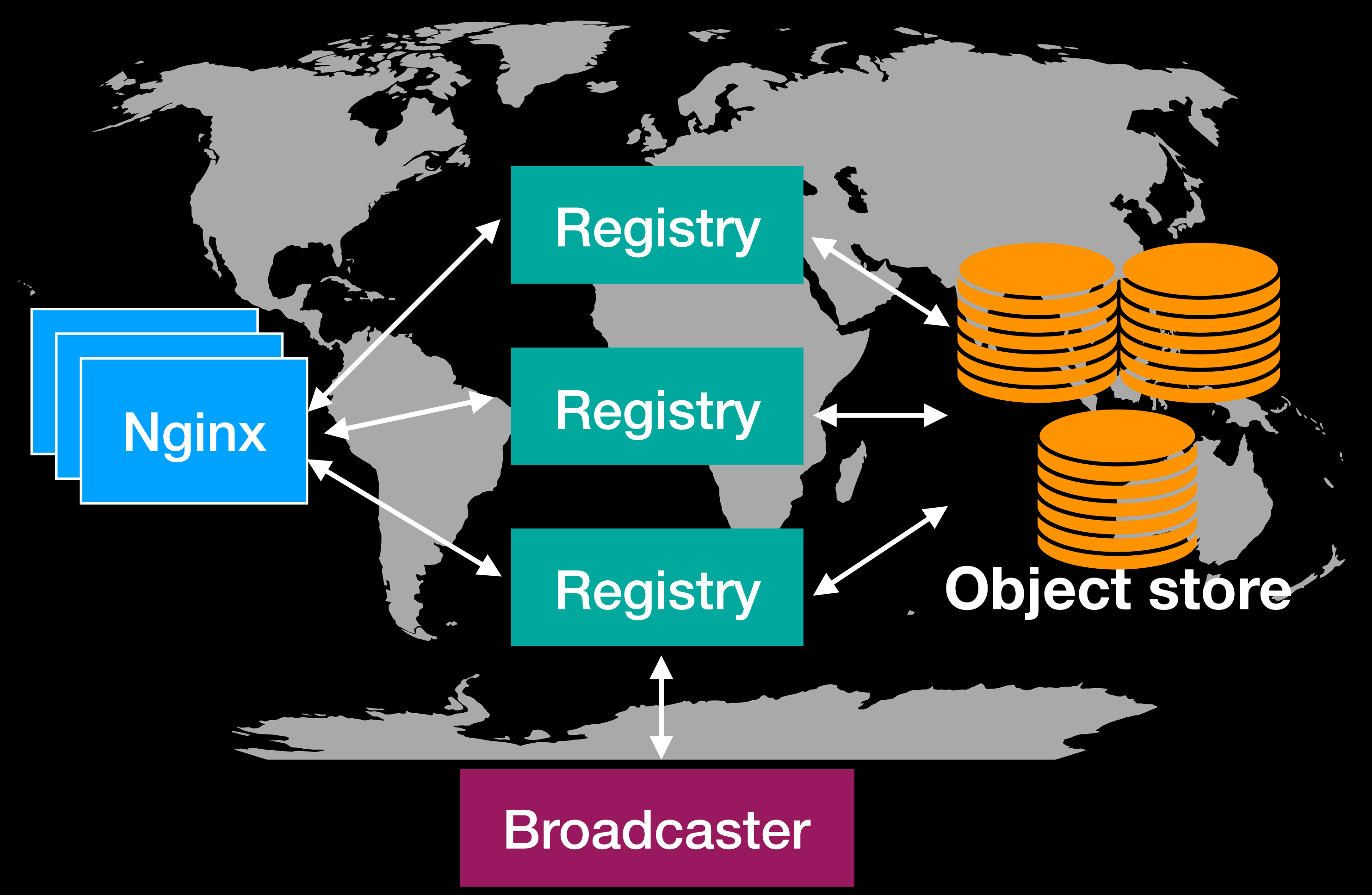
# The IBM Cloud Docker registry traces

- Capture a diverse set of customers: individuals, small & medium businesses, government institutions
- Cover five geographical locations and seven availability zones
- Span 75 days and 38M requests that account for more than ~181TB of data transferred

# IBM Docker registry service

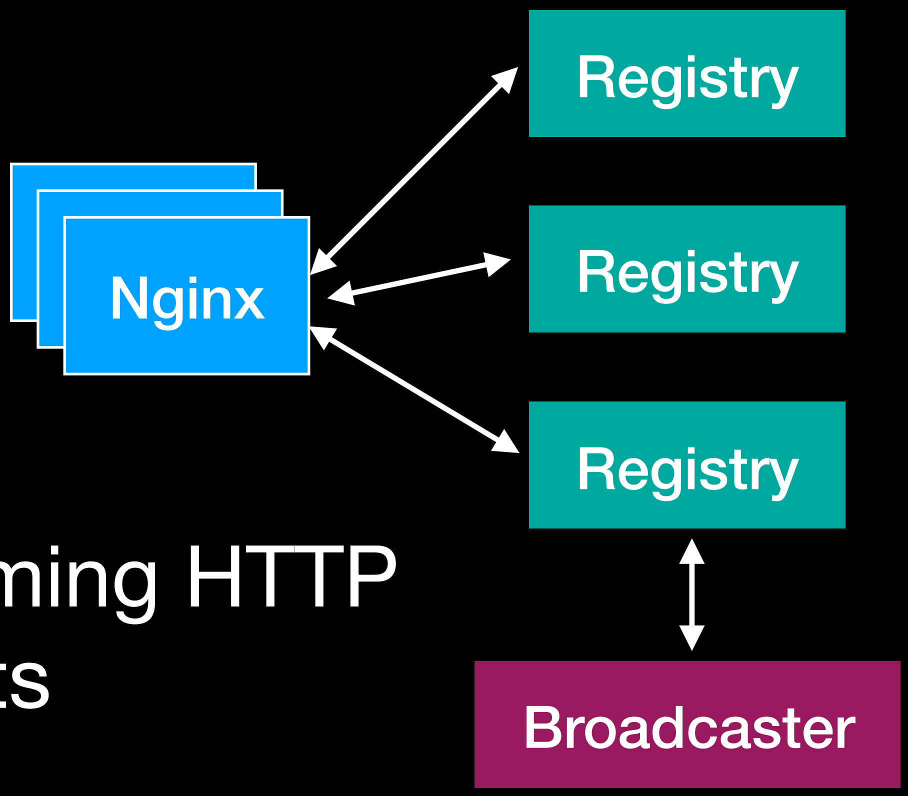
**Five geographical locations constitute seven Availability Zones (AZ)**

- Production
    1. Dallas (dal)
    2. London (lon)
    3. Frankfurt (fra)
    4. Sydney (syd)
- IBM internal
    5. Staging (stg)
- Testing
    6. Pre-staging (prs)
    7. Development (dev)



# Tracing methodology

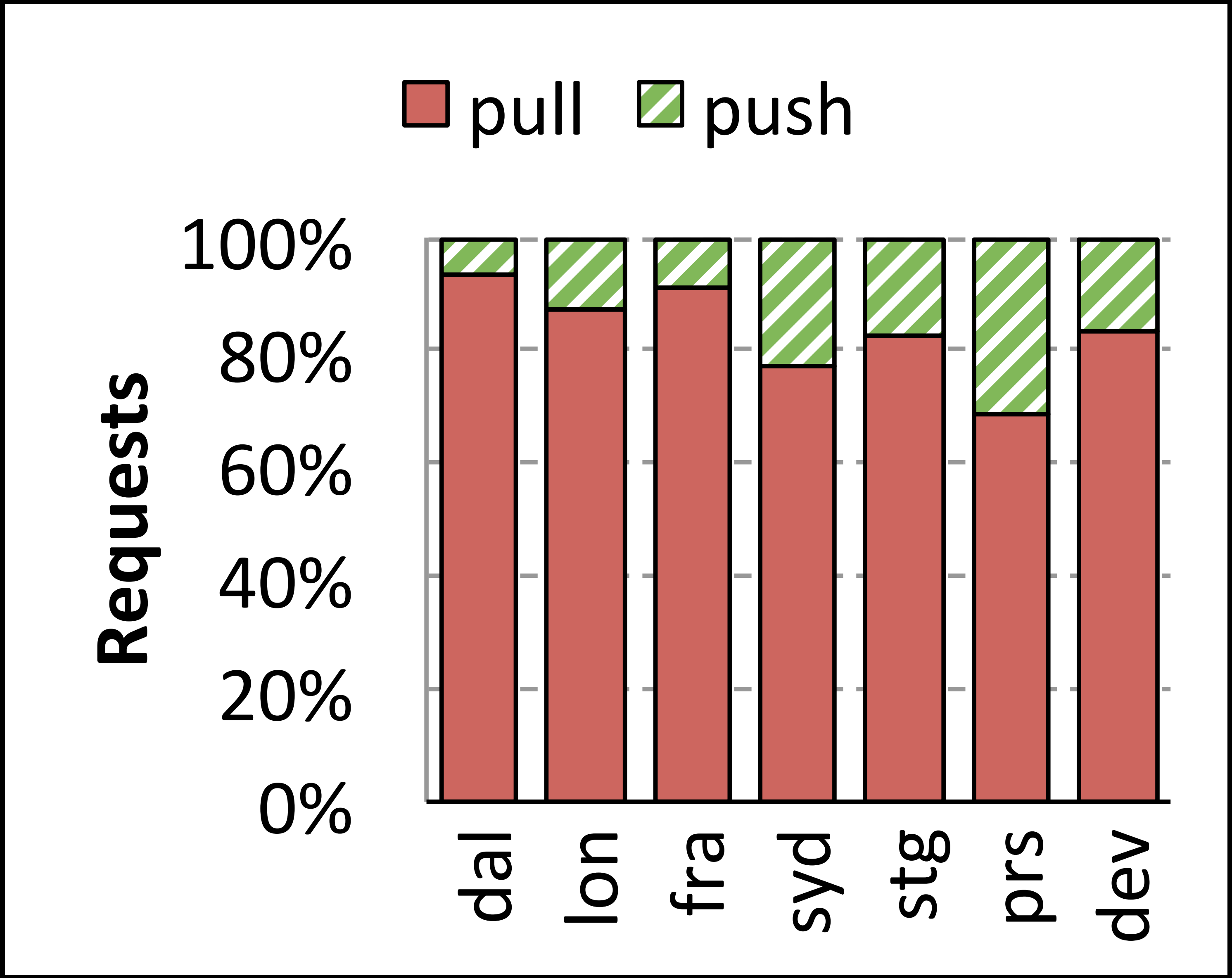
- Collected workload statistics data from **Registry, Ngnix, and Broadcaster**
- Studied requests:
  - GET, PUT, HEAD, PATCH, POST
- Combined traces by matching the incoming HTTP request identifier across the components
- Removed redundant fields and anonymized the whole traces



# To answer the following questions…

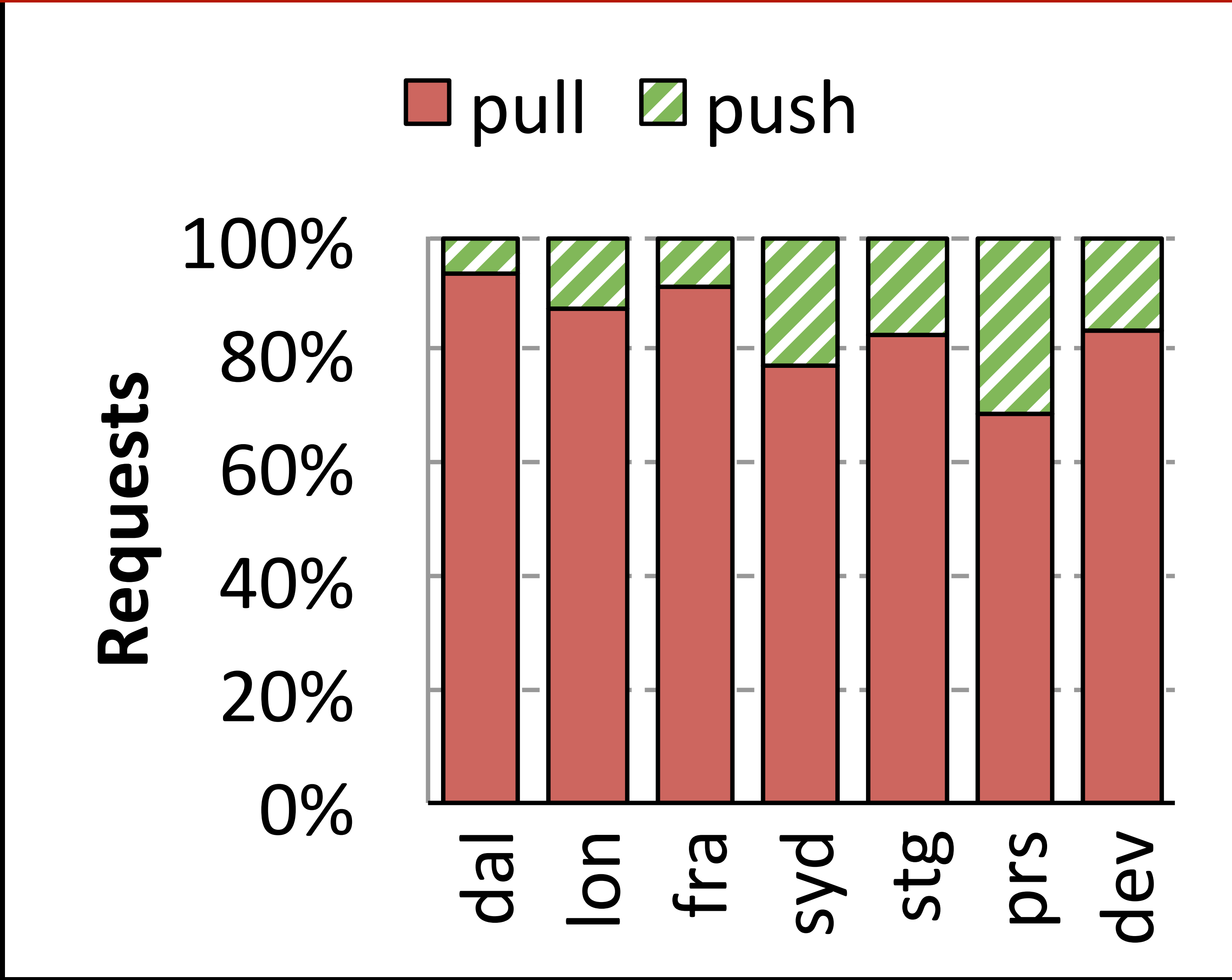
- **Q1: What is the distribution of request types?**
- **Q2: What is the manifest size distribution?**
- **Q3: What is the layer size distribution?**
- **Q4: Is there access locality?**
- … (rest details in paper)

# Q1: What is the distribution of request types

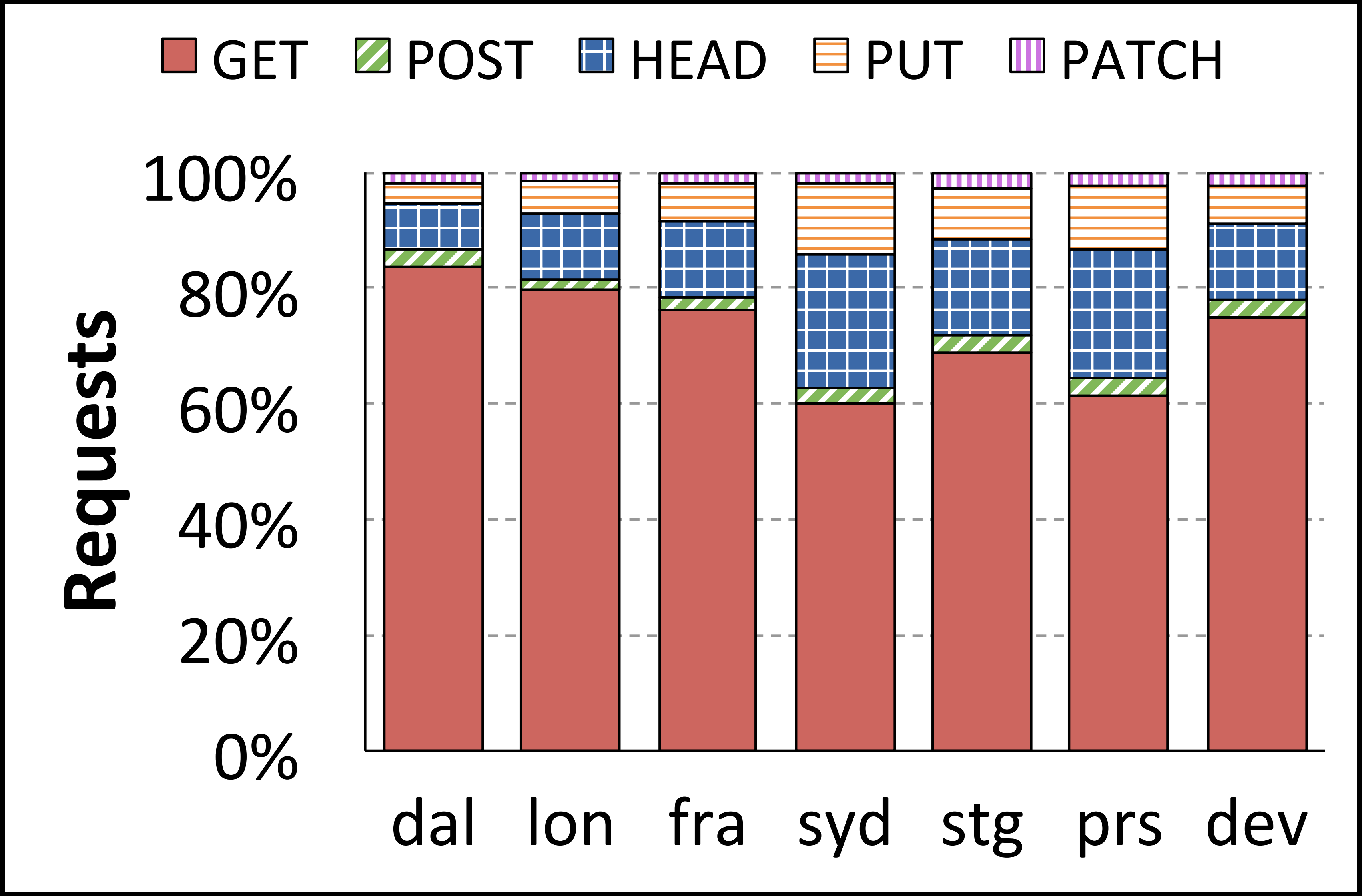


# Q1: What is the distribution of request types

**80-95% of requests are reads (pulls)**

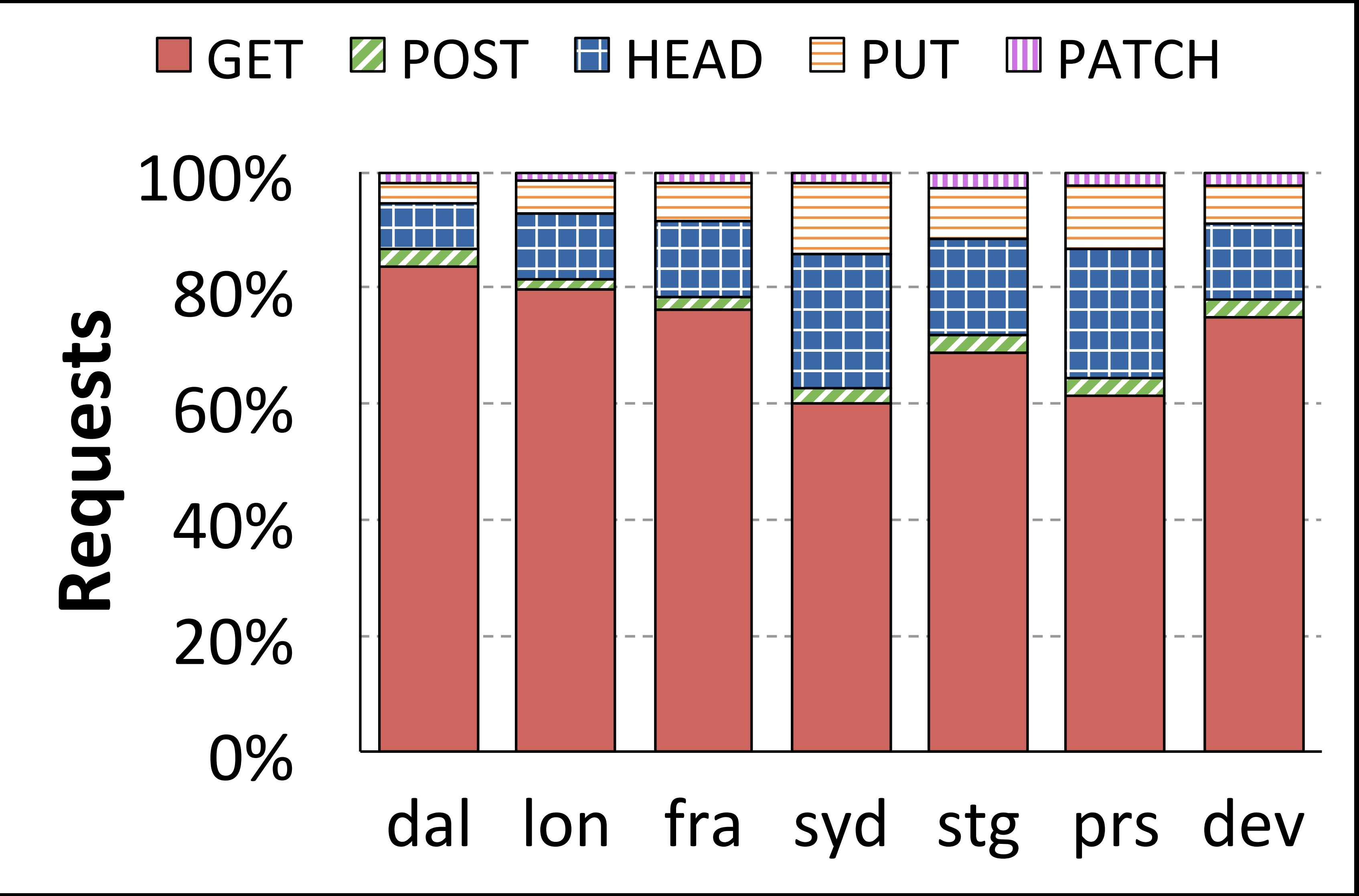


# Q1: What is the distribution of request types

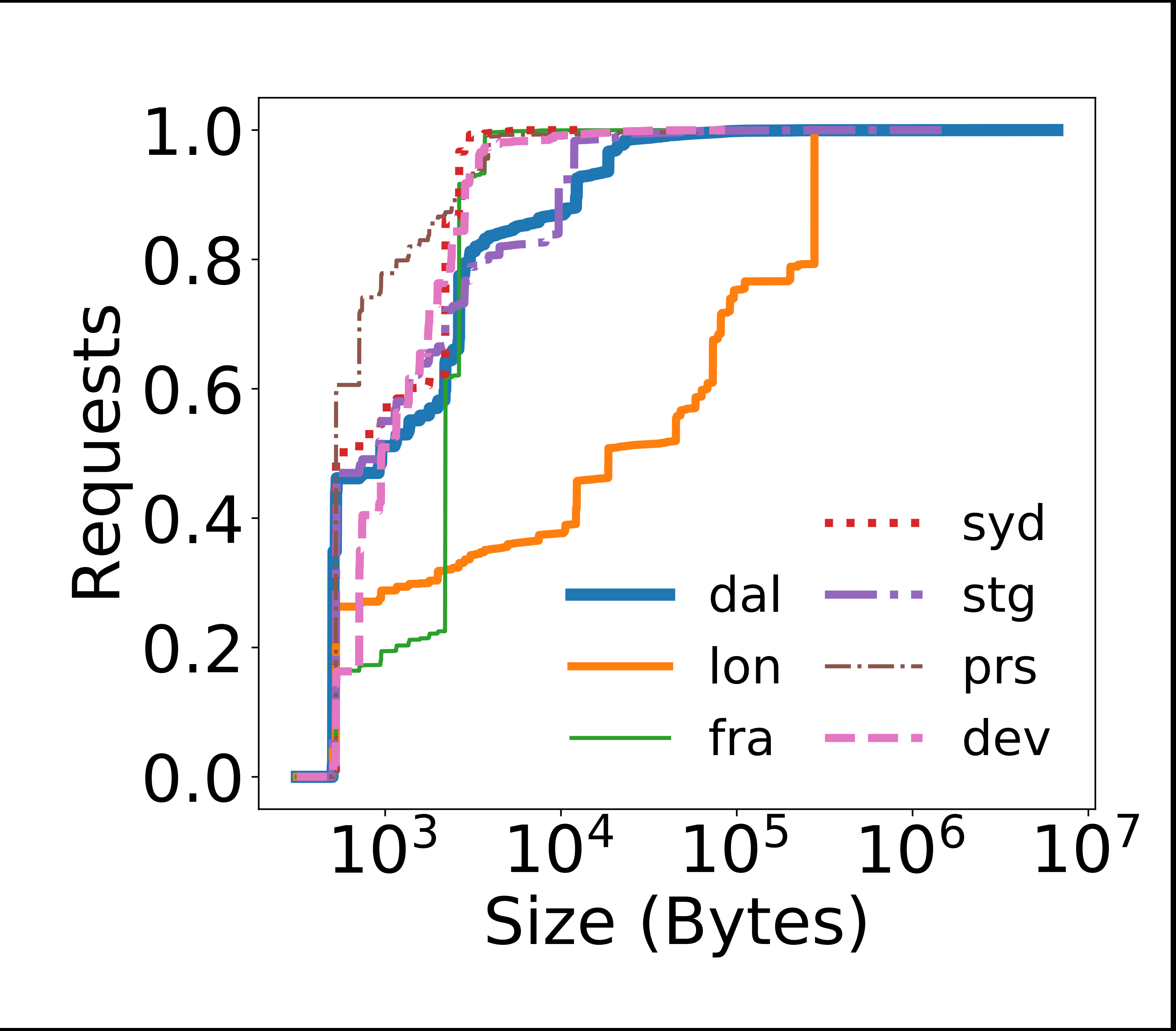


# Q1: What is the distribution of request types

**60% of requests are GET and 10-22% are HEAD requests**

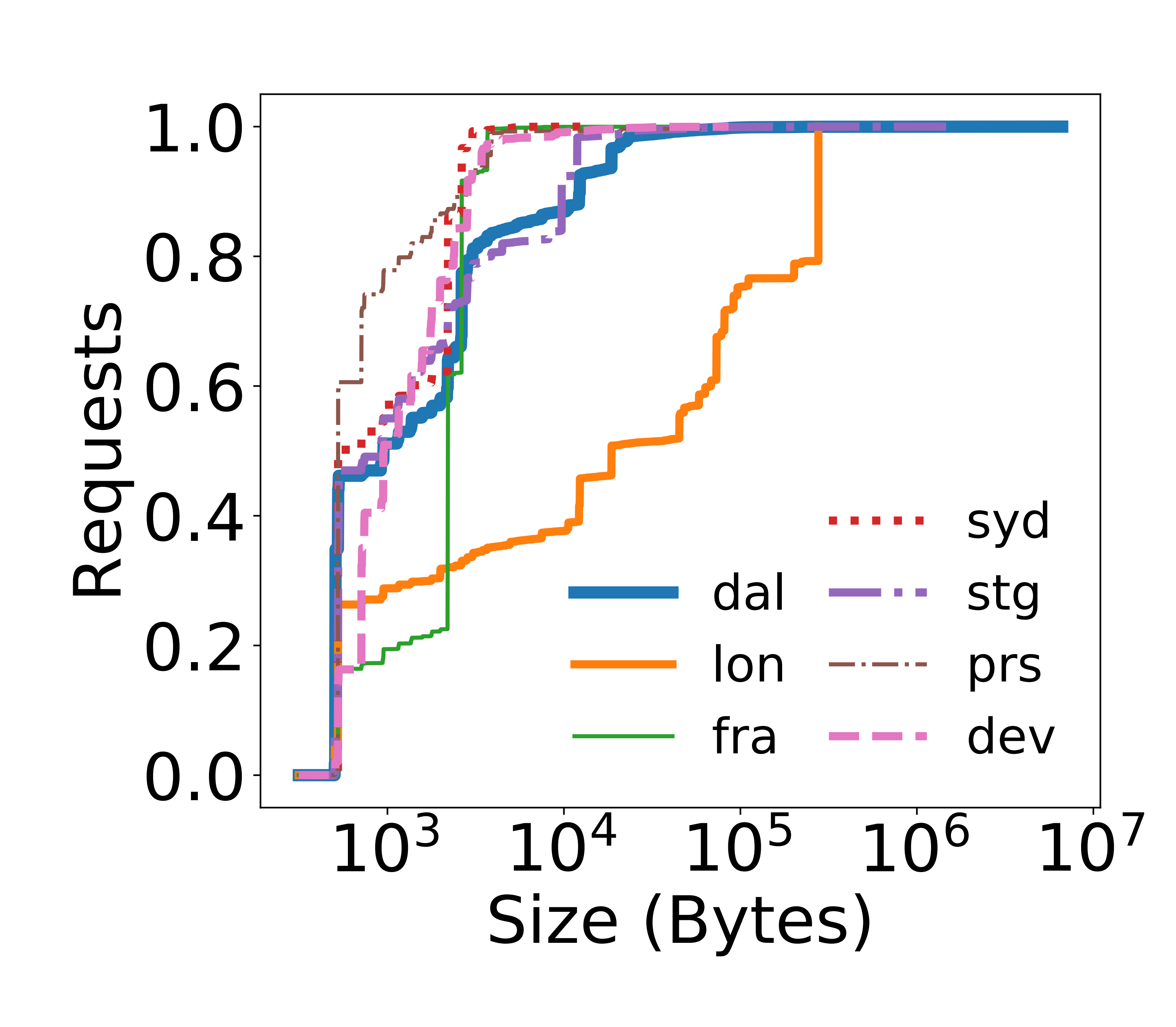


# Q2: What is the manifest size distribution

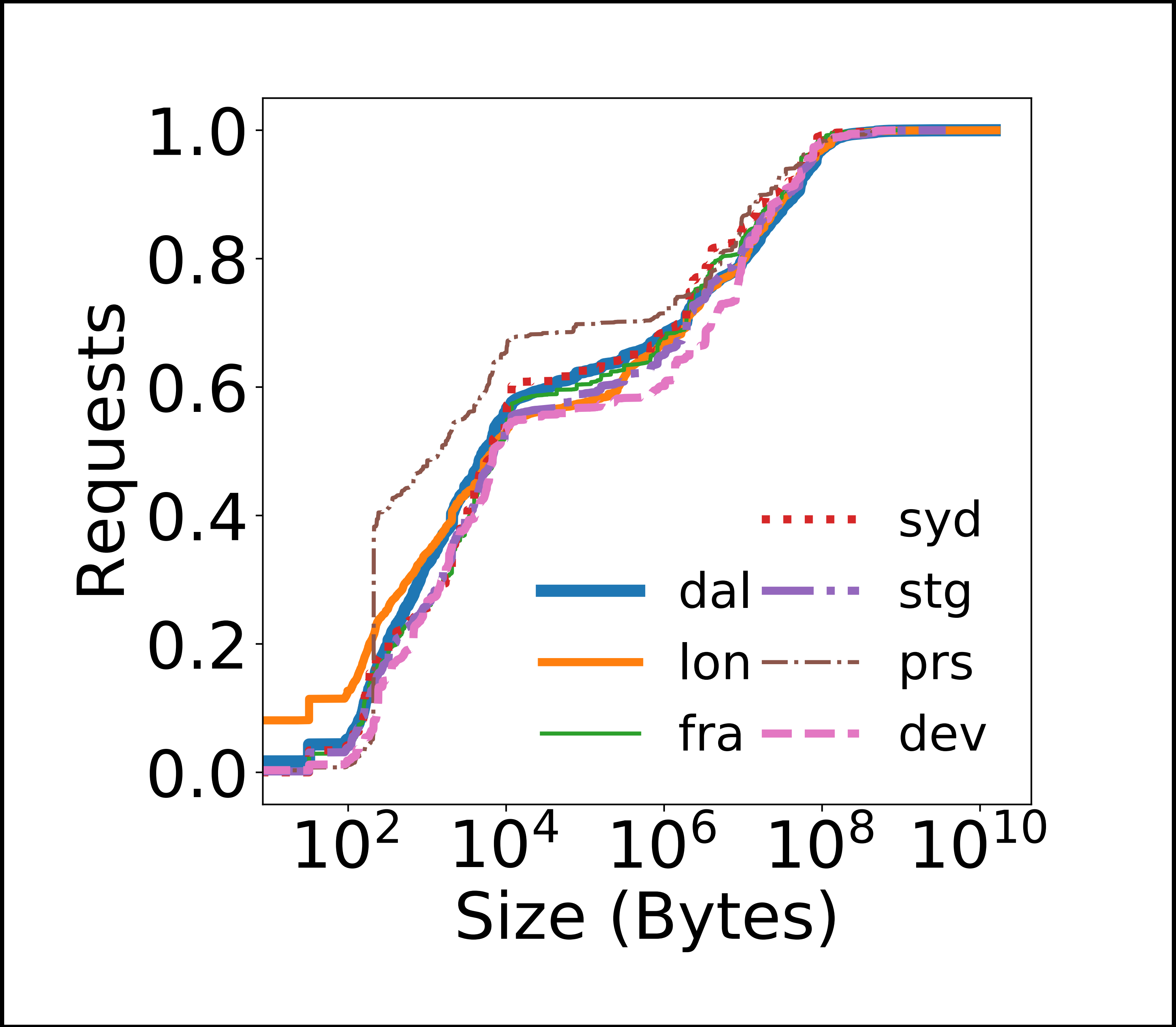


# Q2: What is the manifest size distribution

Typical manifest size is around 1KB

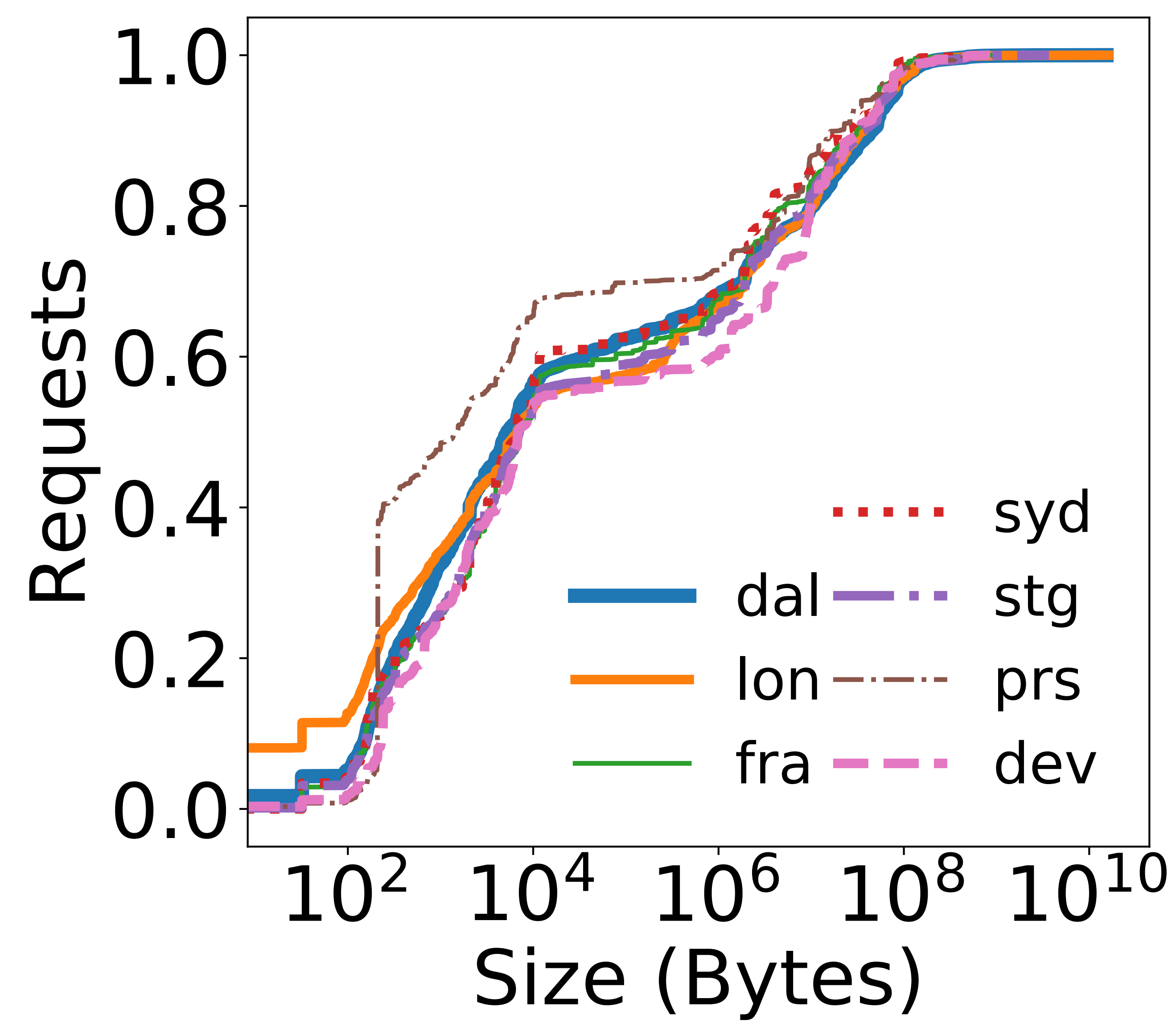


# Q3: What is the layer size distribution

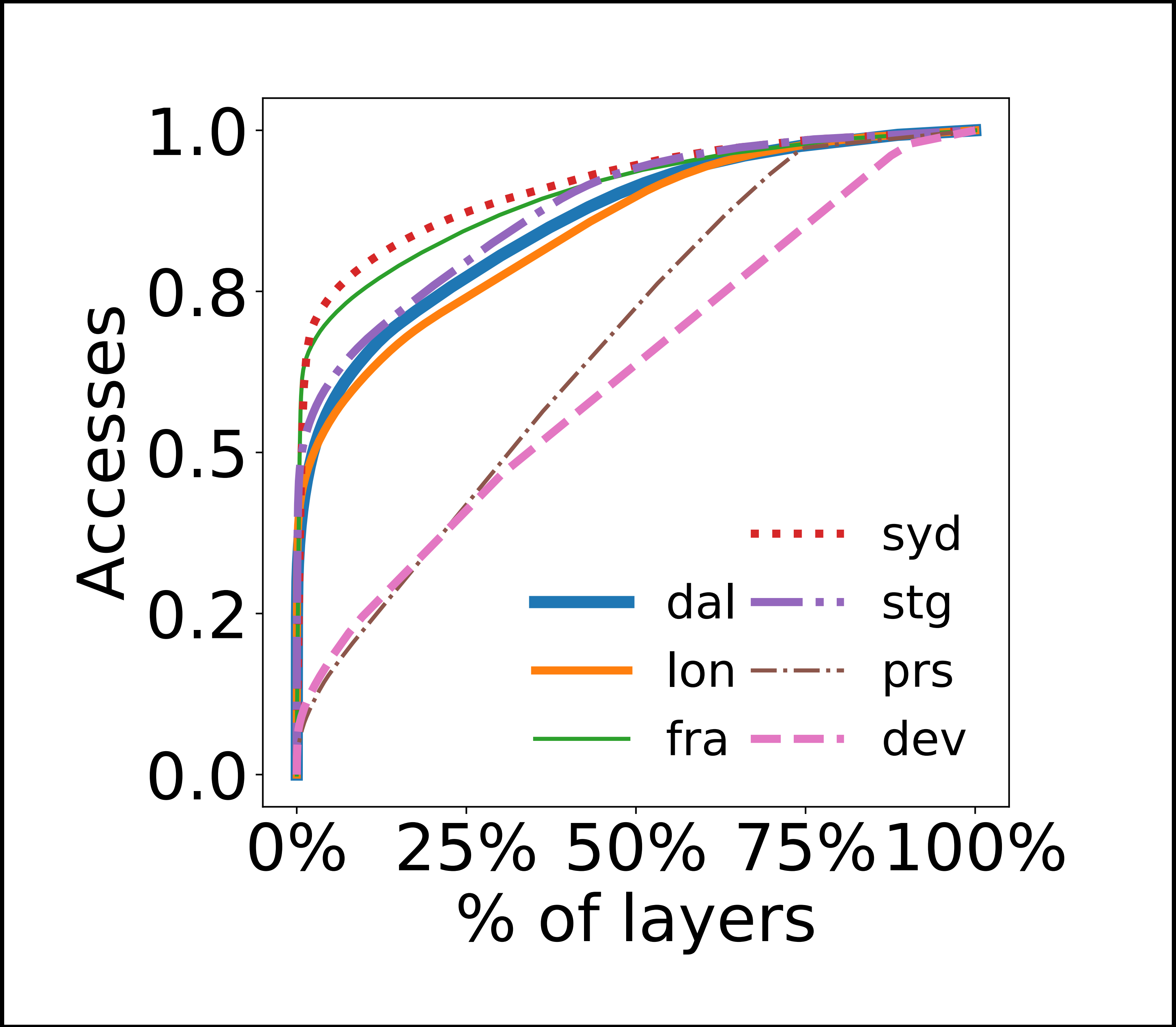


# Q3: What is the layer size distribution

65% of the layers are smaller than 1MB and around 80% are smaller than 10MB

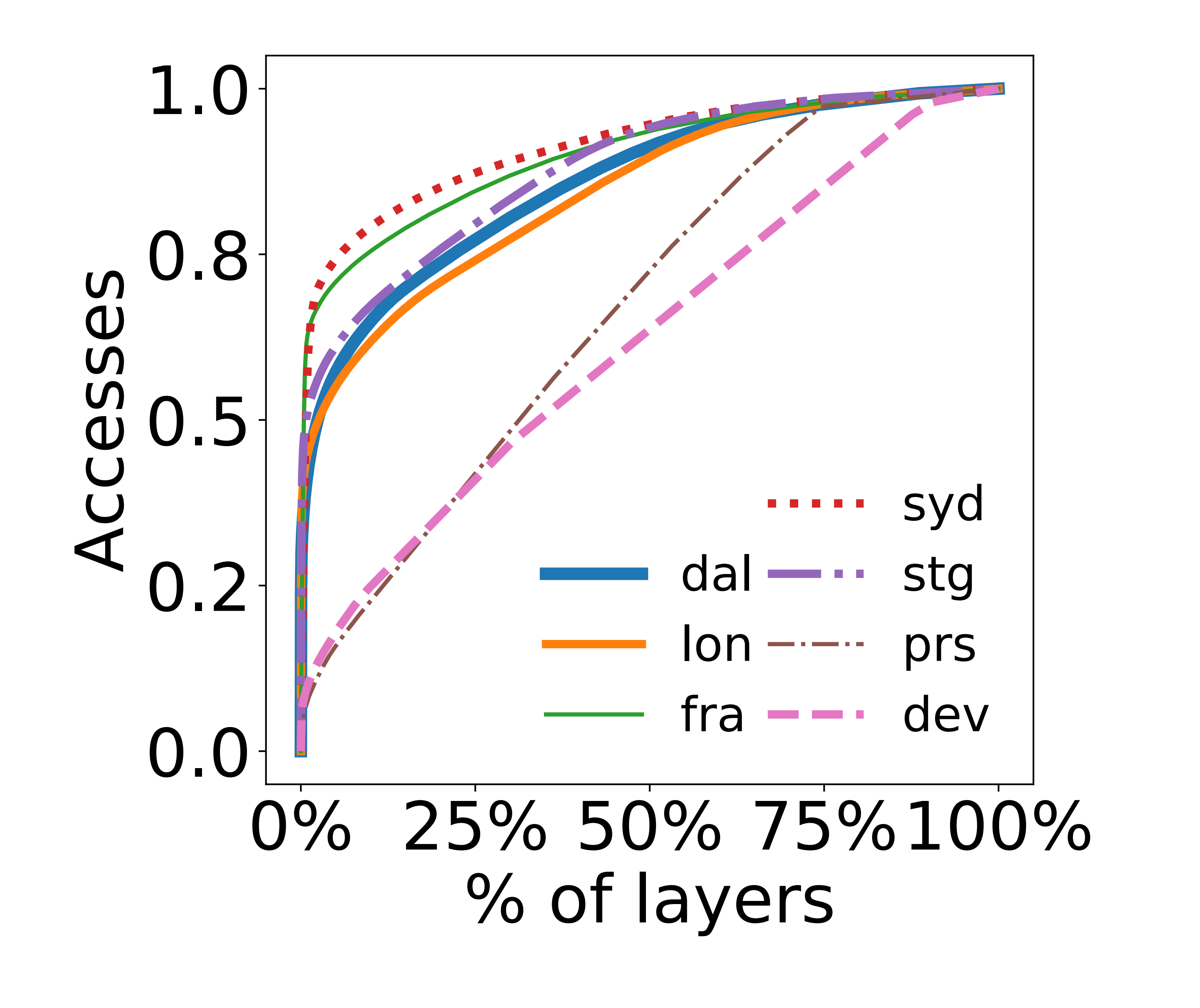


# Q4: Is there access locality?

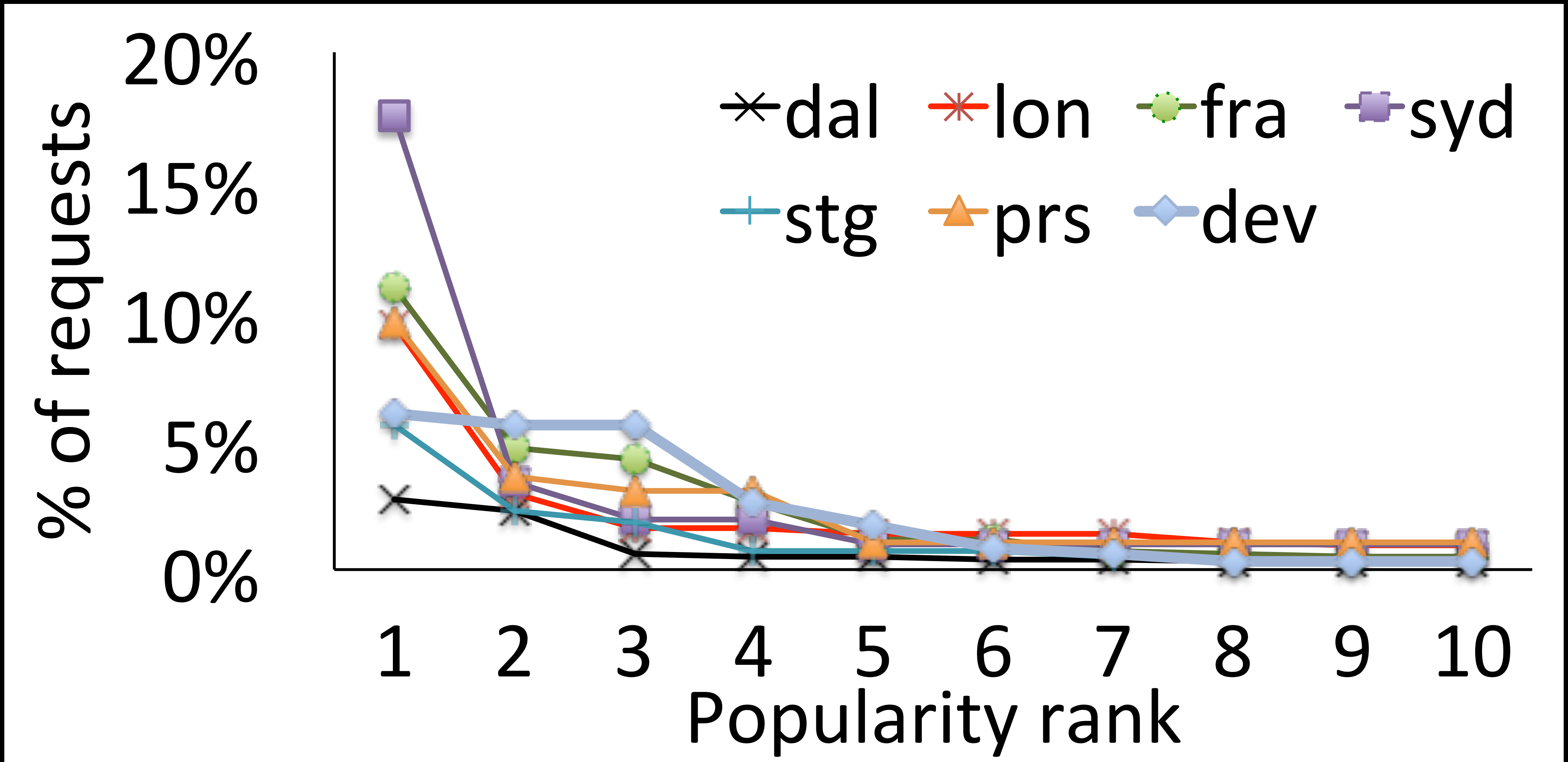


# Q4: Is there access locality?

1% of the most accessed layers account for 42% and 59% of all requests in dal and syd, respectively

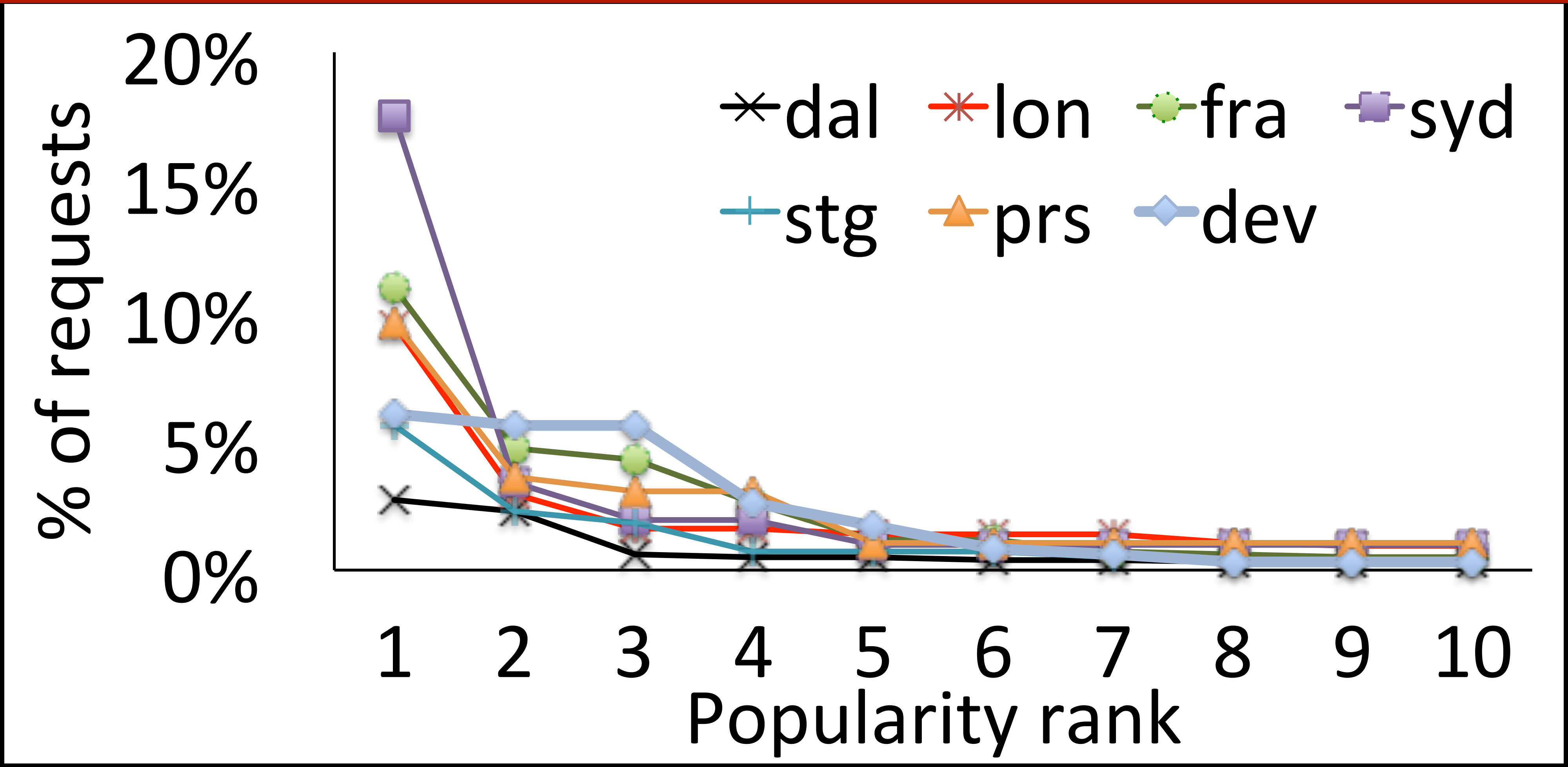


# Q4: Is there access locality?



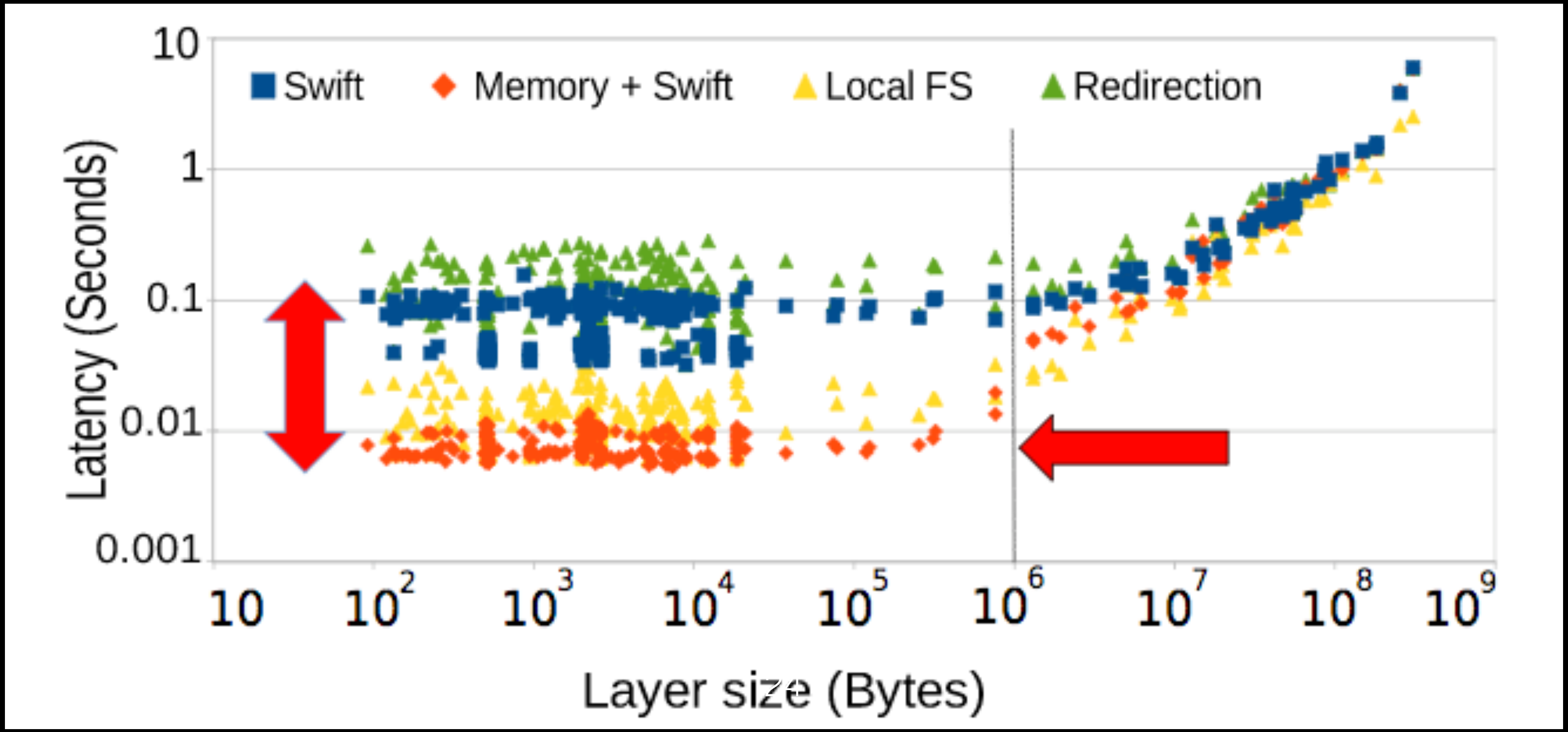
# Q4: Is there access locality?

The popular rate drops rapidly as we move from the top most popular to the 10th most popular layer



# Enabling further analysis: analysis of caching effect

- Experimental setup
  - Registry on 32-core machine with 64GB DRAM and 512GB SSD
  - OpenStack Swift object store on 10 similar nodes
  - Trace replayer on 6 separate nodes



# Summary

- Performed a quantitative characterization study of a production Docker registry deployment
  - Registry workload is read-intensive
  - Layer sizes are mostly small (**perfect for caching**)
  - Strong correlation exists between layer requests (**good for prefetching**)
- **Caching** is such an optimization technique that can be **universally** applied anywhere in the Computer Science world
  - Docker registry workloads checked √