Cachew

ML Input Data Processing as a Service

Dan Graur[1], Damien Aymon[1], Dan Kluser[1], Tanguy Albrici[1], Chandu Thekkath[2] and Ana Klimovic[1]

[1] ETH zürich

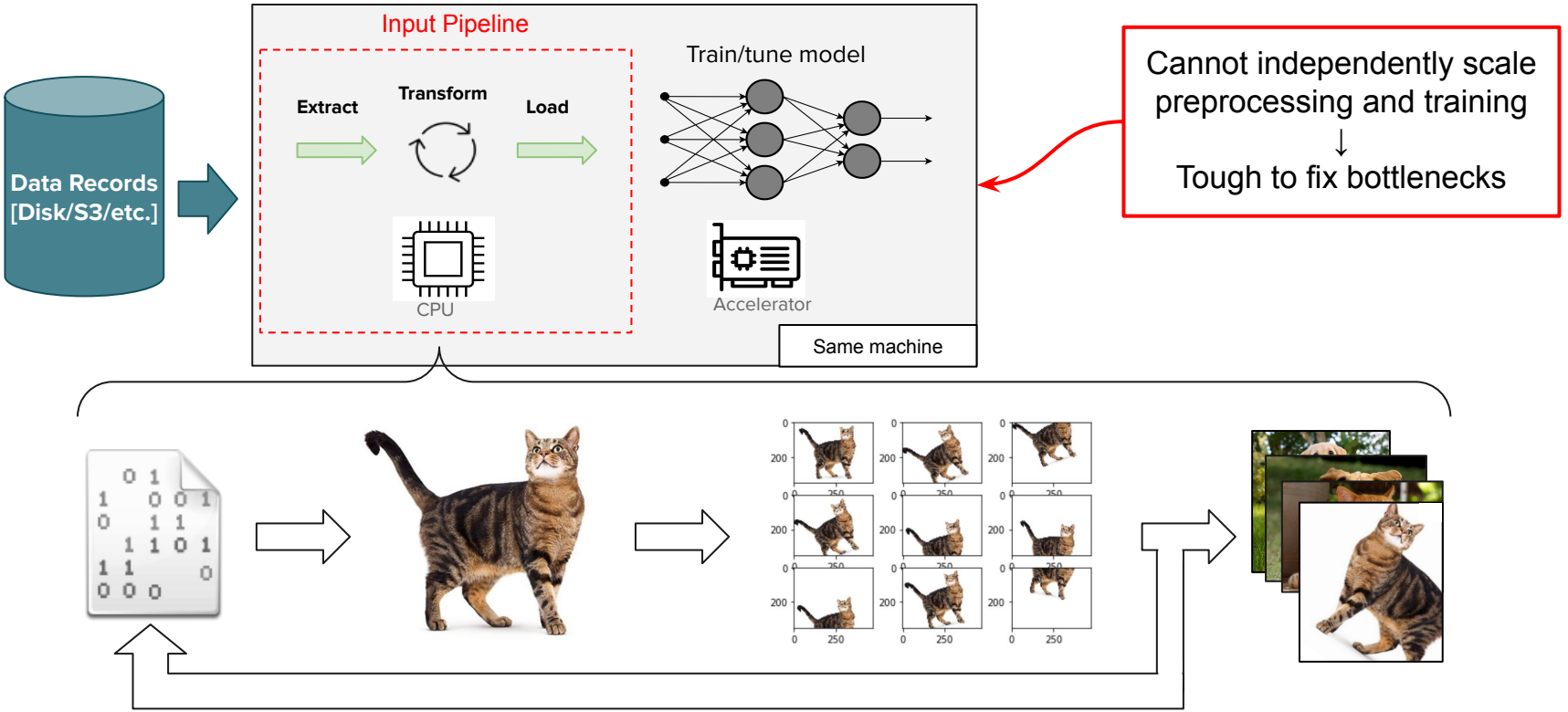
[2] Google

https://github.com/eth-easl/cachew

Efficient simulations on GPU hardware, October 24, 2022

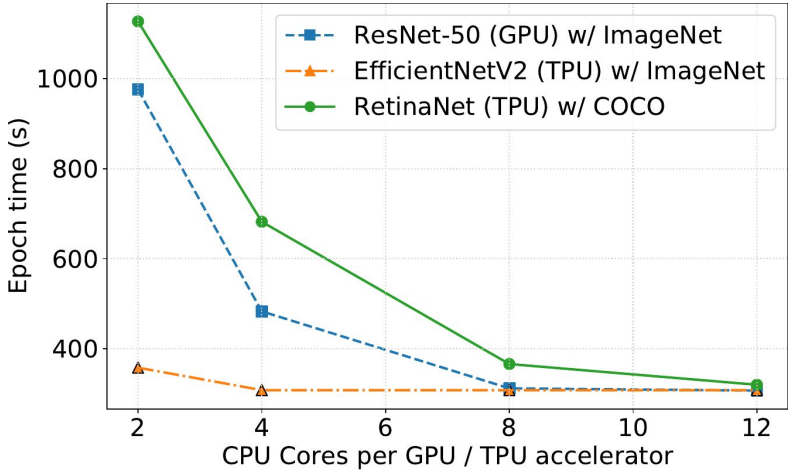
Systems@ETH zürich

Data Processing in ML Workloads



Input Preprocessing Challenges

- Waiting for batches costs time and money [1]



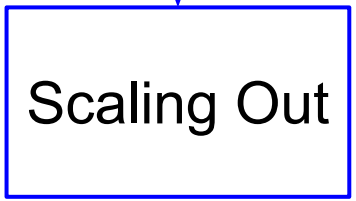
- Small set of pipelines account for most computation [1]
- Preprocessing can consume more power than training [2]

[1] Murray et al. *tf. data: A machine learning data processing framework.* VLDB'21.

[2] Zhao et. al. *Understanding and co-designing the data ingestion pipeline for industry-scale recsys training*, ISCA'22.

Opportunities

- Waiting for batches costs time and money [1]



- Small set of pipelines account for most computation [1]
- Preprocessing can consume more power than training [2]

[1] Murray et al. *tf. data: A machine learning data processing framework.* VLDB'21.

[2] Zhao et. al. *Understanding and co-designing the data ingestion pipeline for industry-scale recsys training*, ISCA'22.

Current Landscape in ML Preprocessing

- Solutions for disaggregating input pipeline and model exist:
  - tf.data service from Google

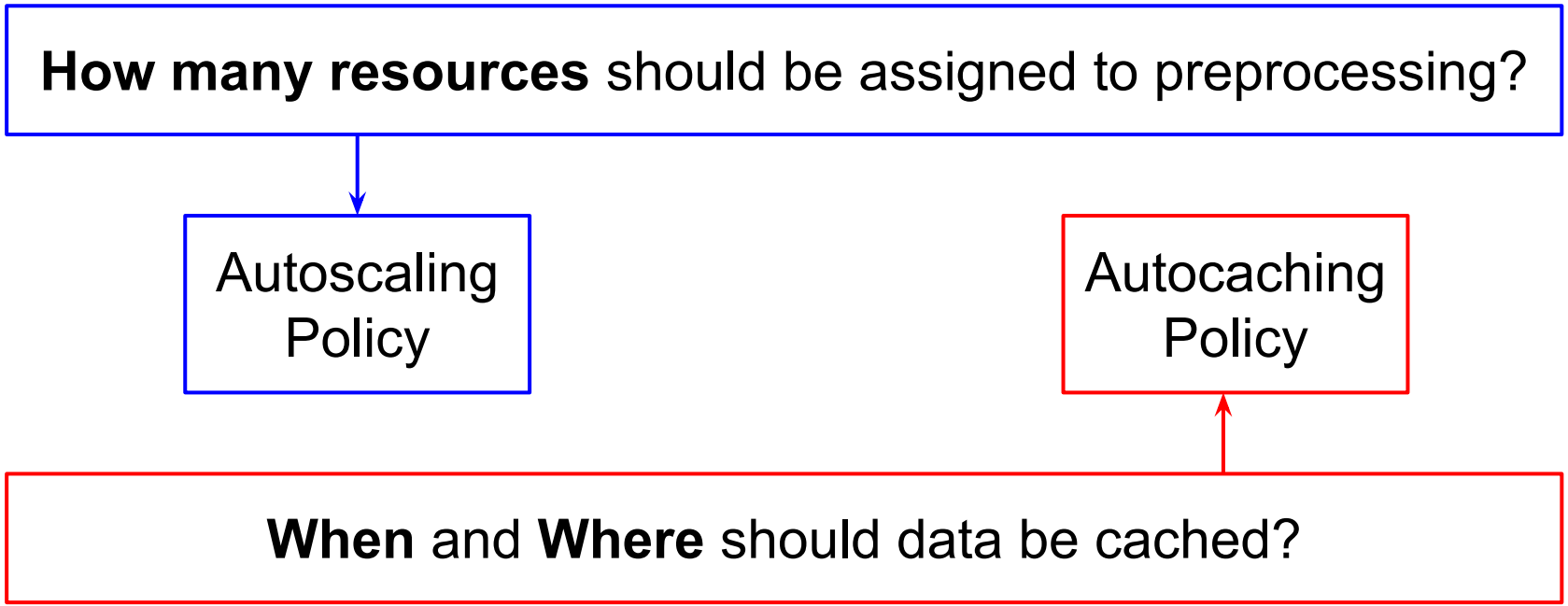
  → ***However, resource allocation for data processing is complex***

- Caching functionality already exist in many frameworks:

  → ***However, caching decisions are complex***

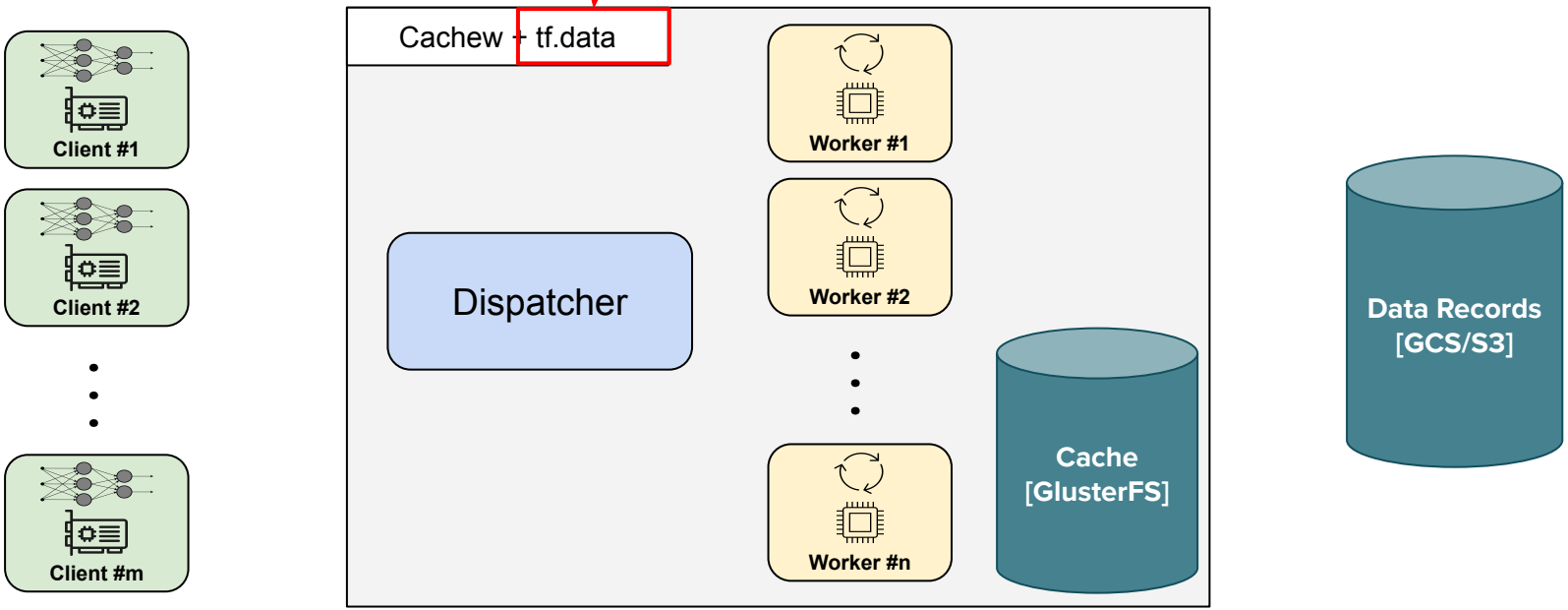
- **Automating these decisions is essential**

Main Contributions

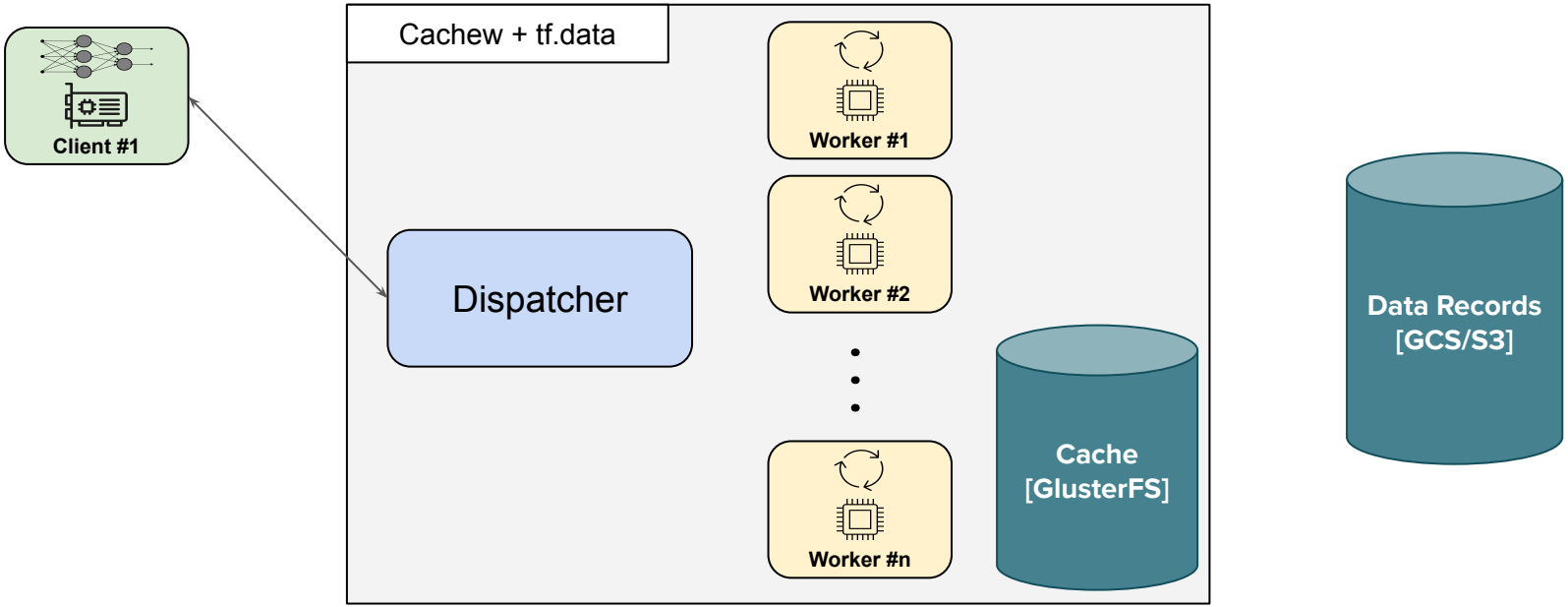


System Architecture

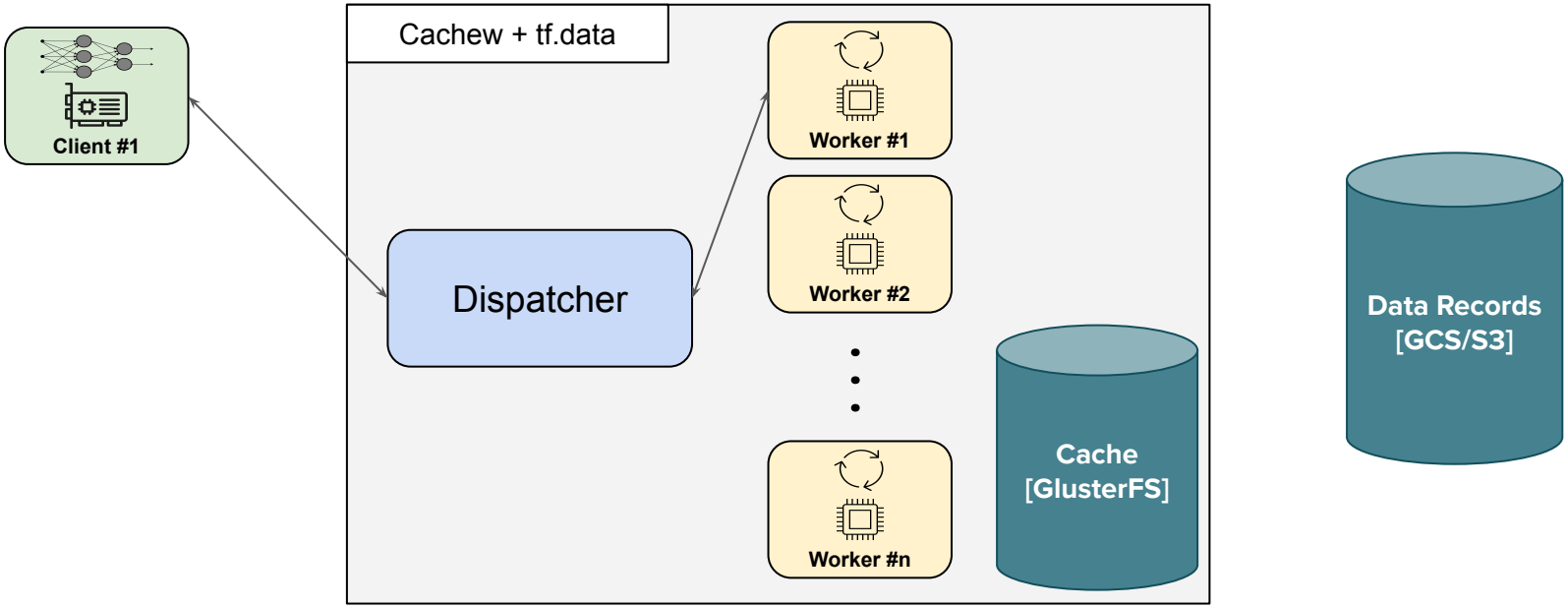
- Disaggregation available
- Open-source
- Large-scale
- Impactful



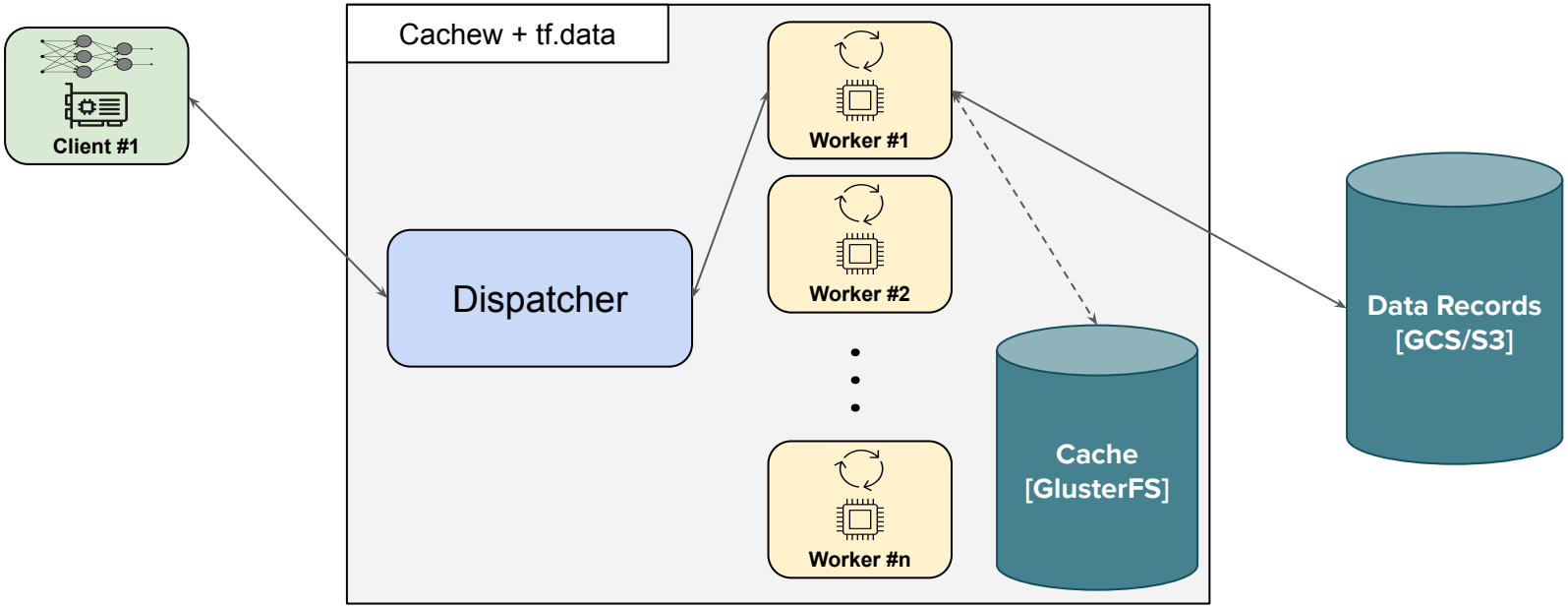
Client-Service Interaction in Cachew



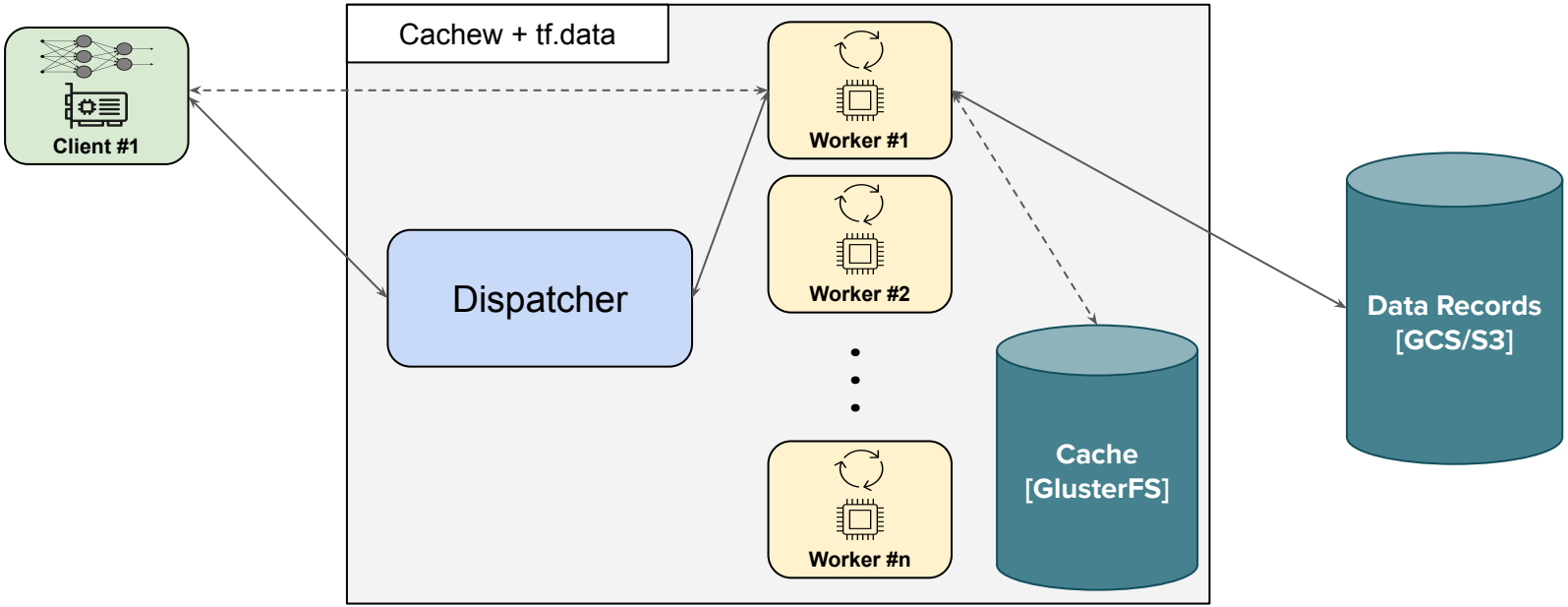
Client-Service Interaction in Cachew



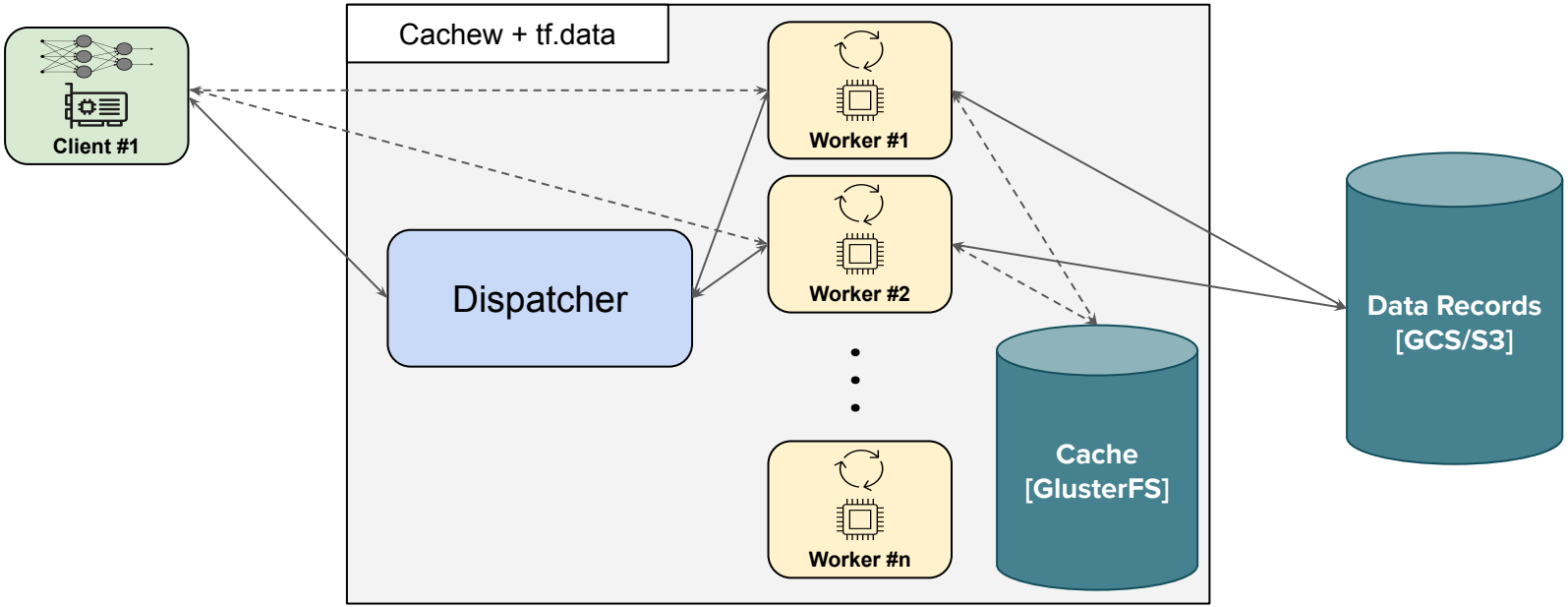
Client-Service Interaction in Cachew



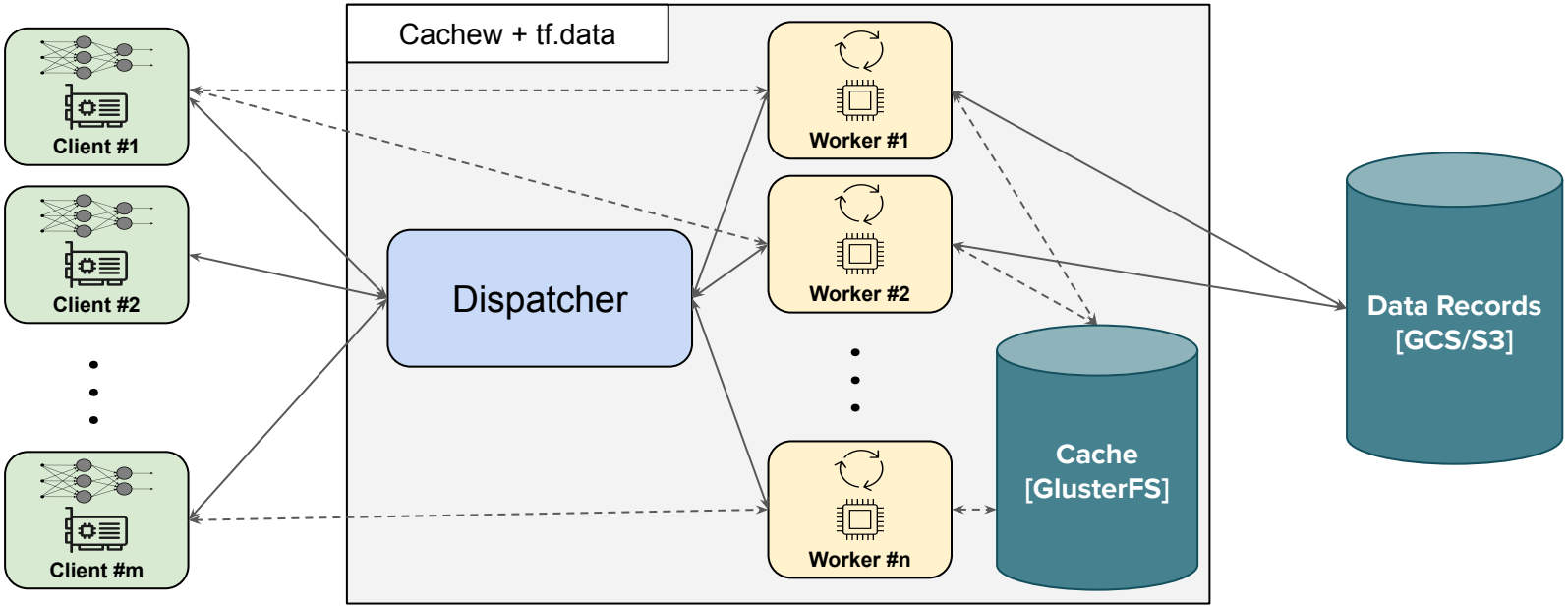
Client-Service Interaction in Cachew



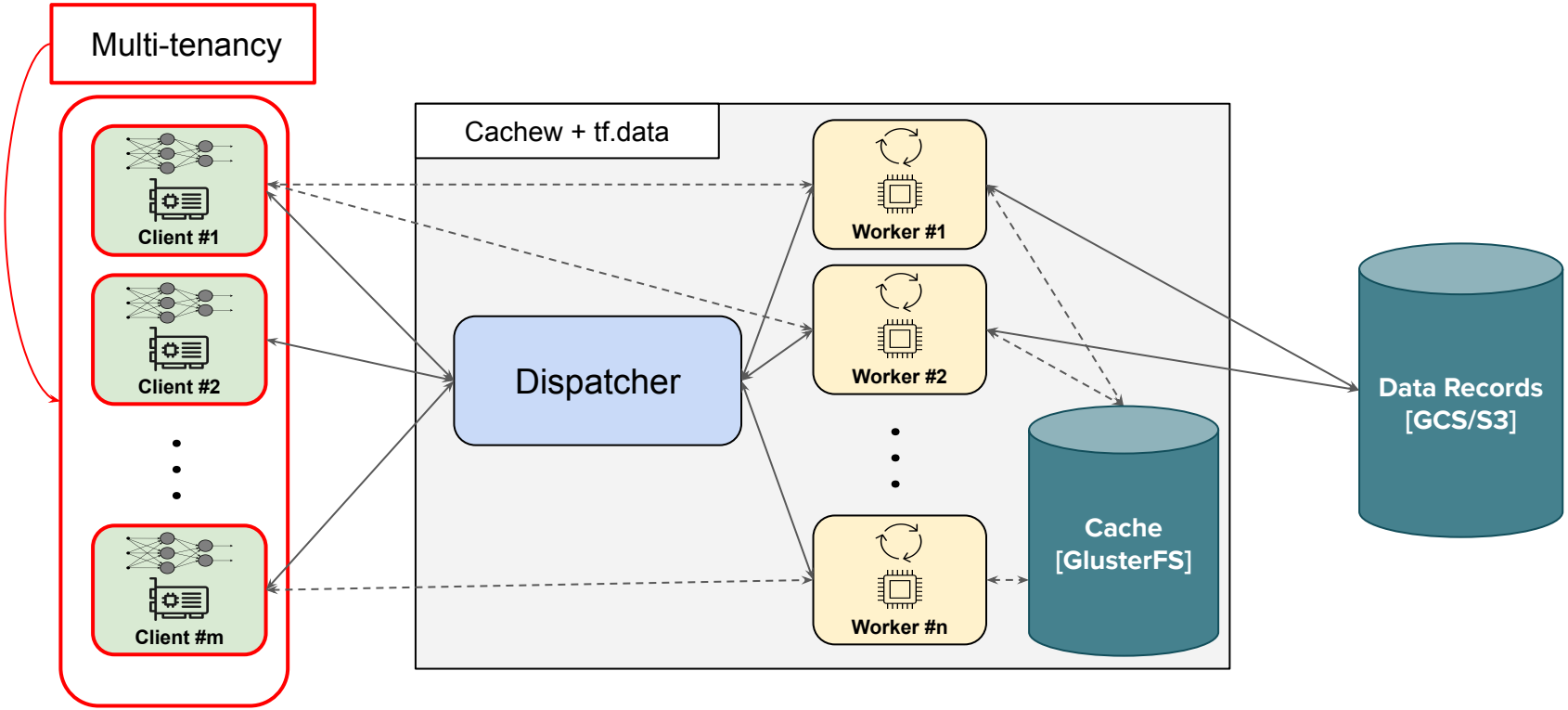
Client-Service Interaction in Cachew



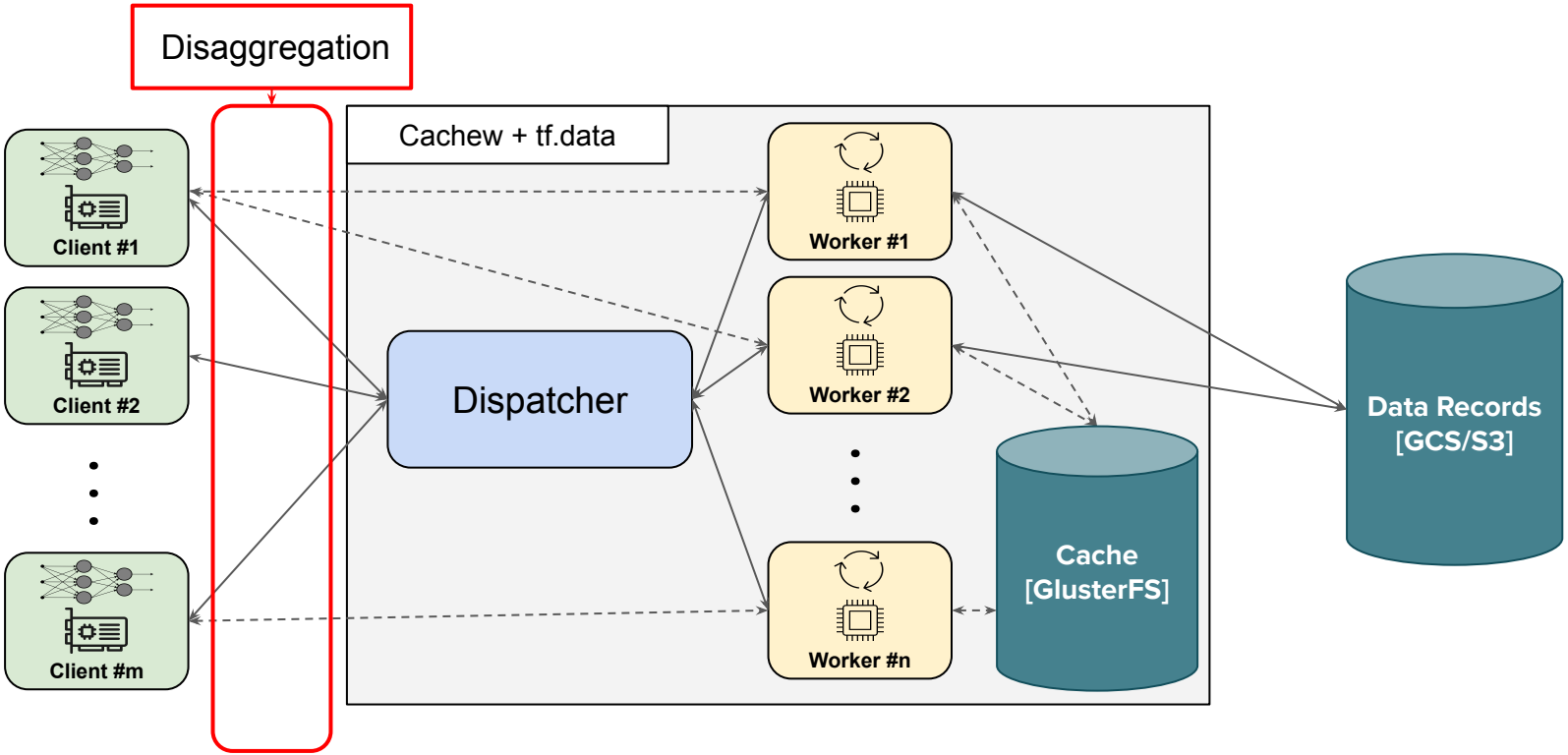
Client-Service Interaction in Cachew



Main Features of Cachew

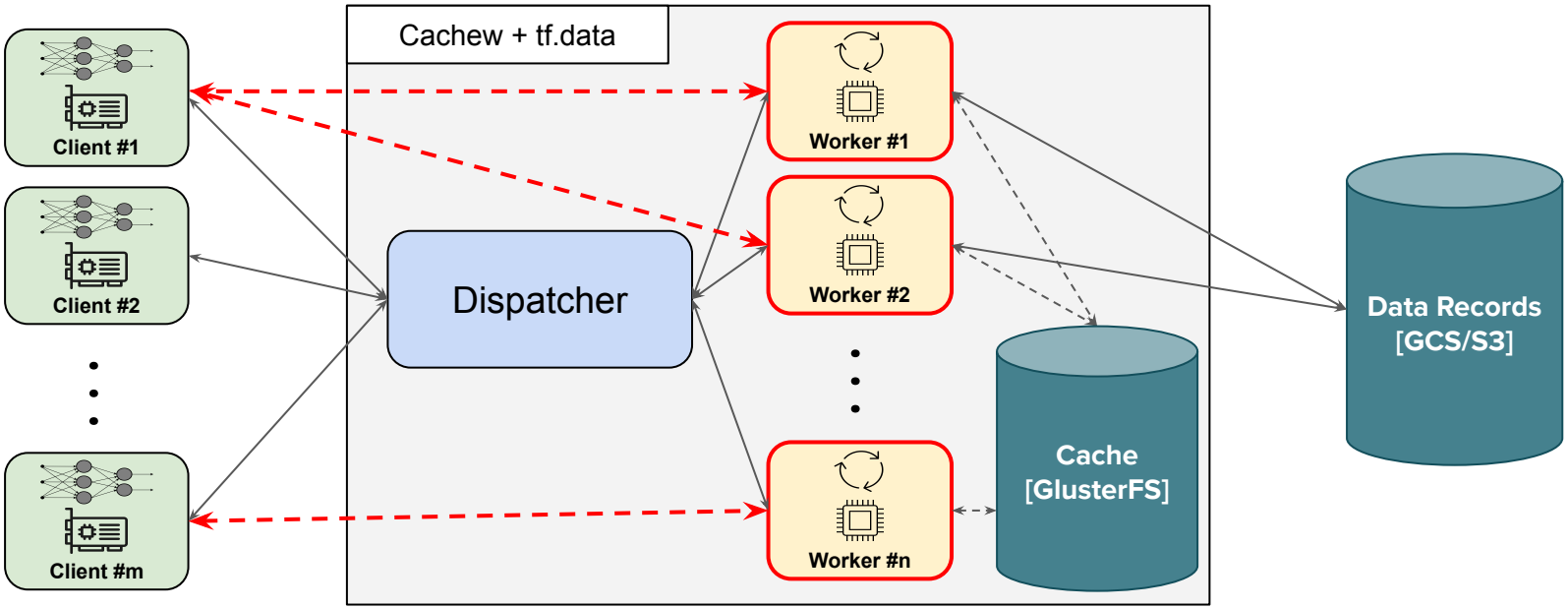


Main Features of Cachew



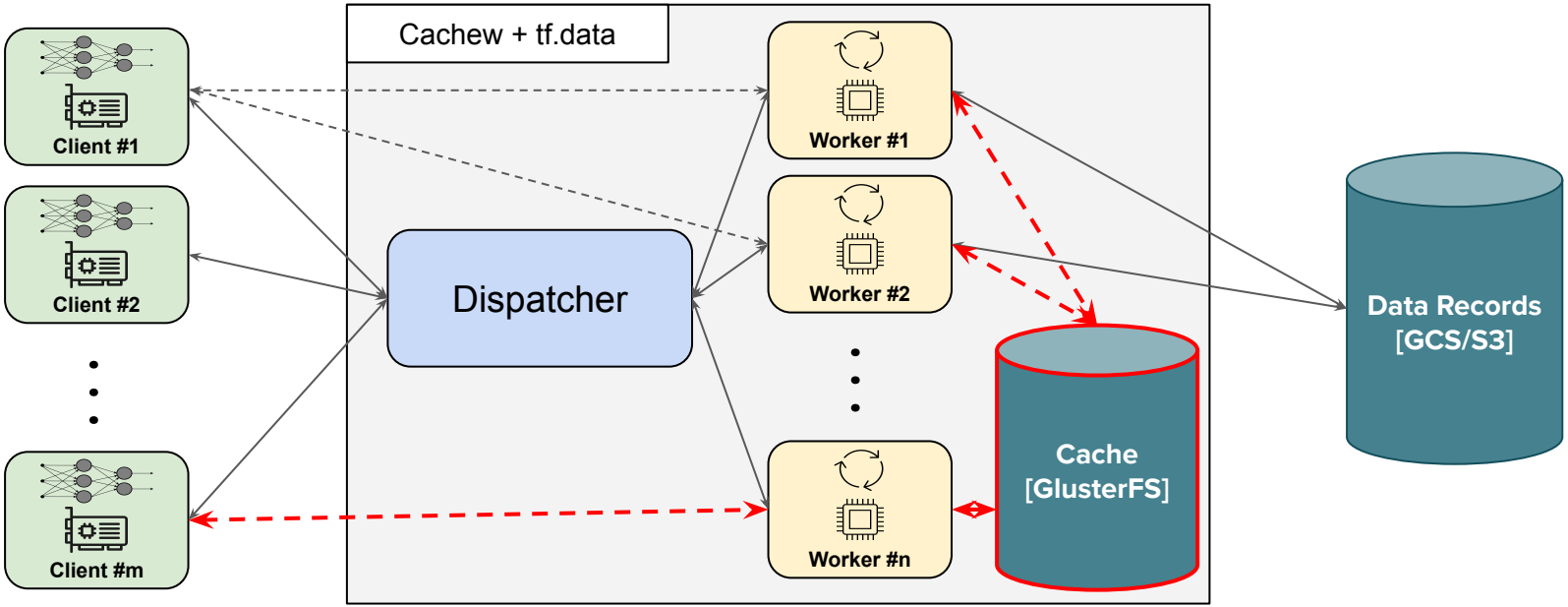
Main Features of Cachew

Autoscaling



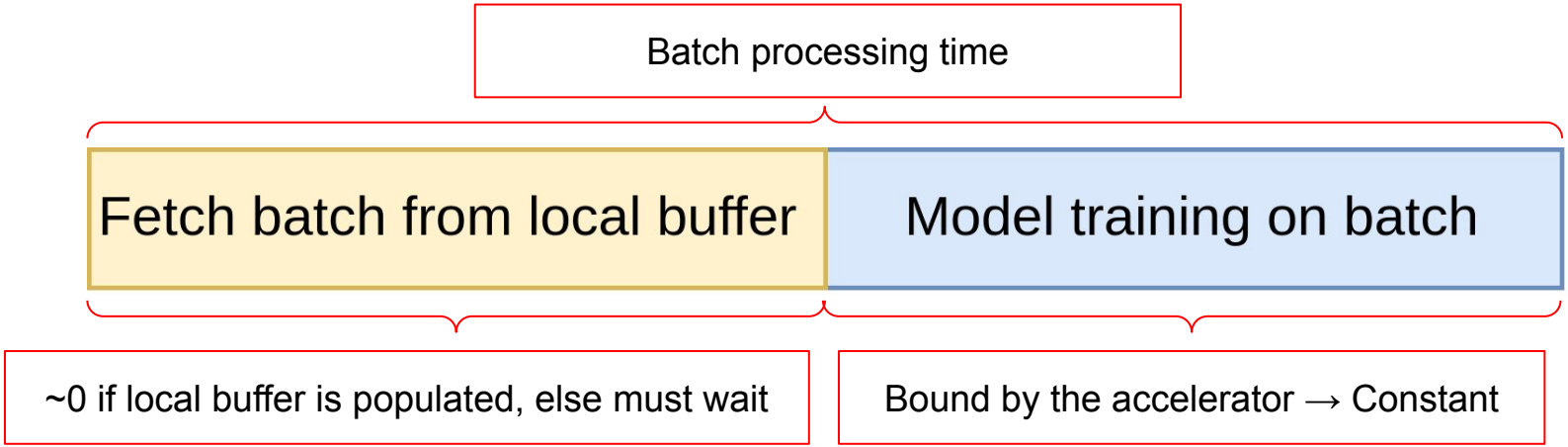
Main Features of Cachew

Autocaching



Autoscaling Policy

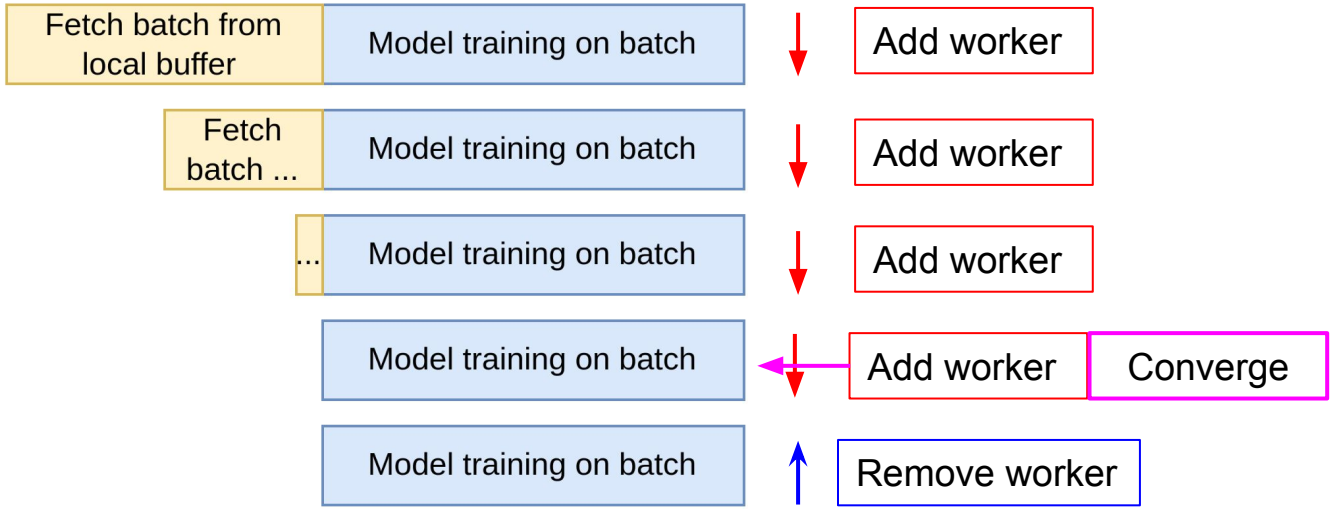
- Two steps are executed when processing a batch:



- Intuition: add workers to preprocessing until Batch Processing Time converges

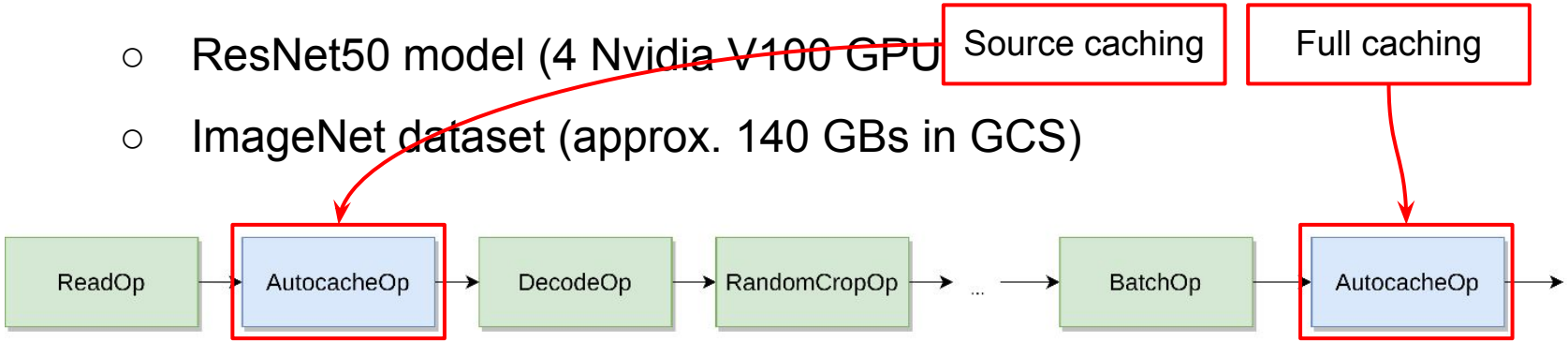
Autoscaling Policy Example

- Add workers until convergence



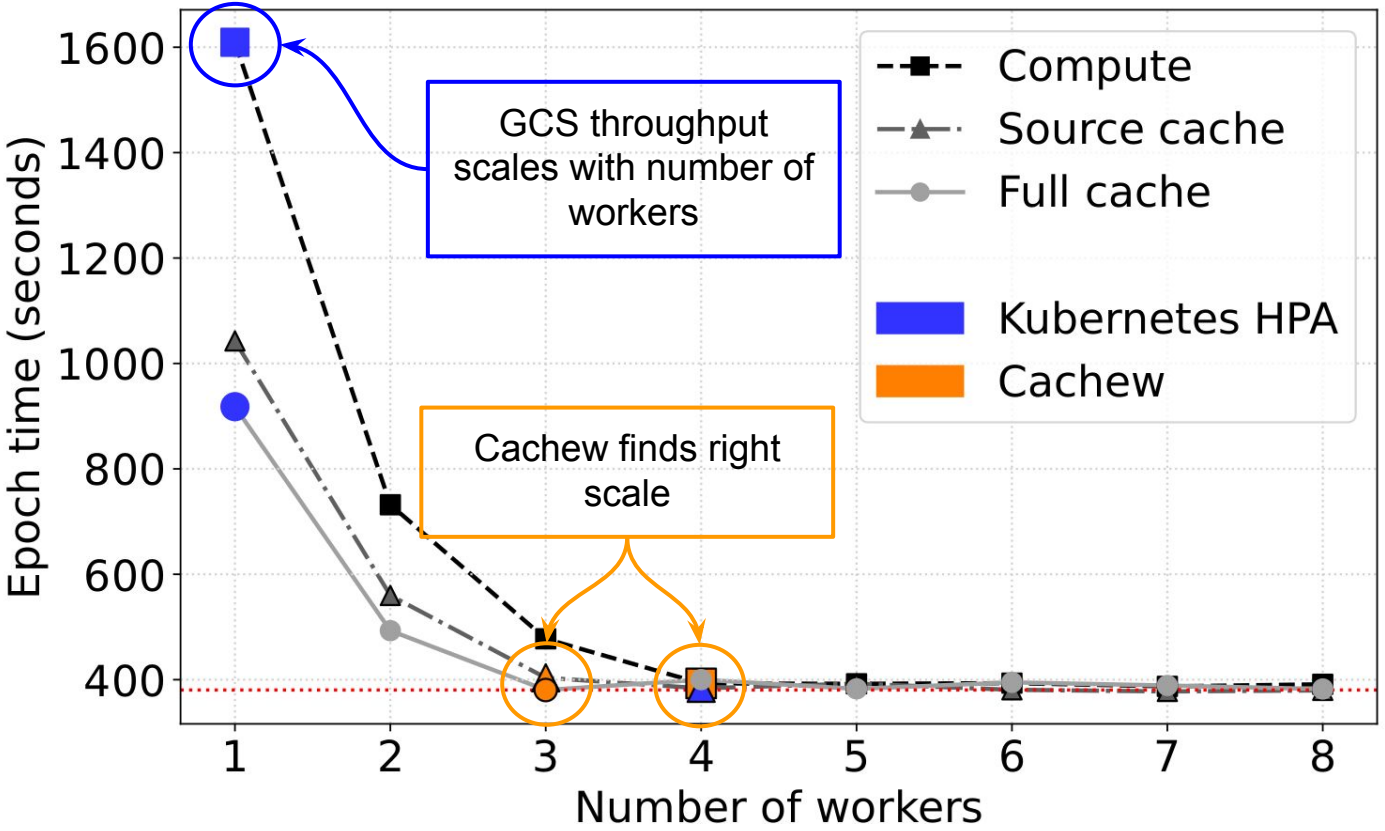
Evaluation: Autoscaling Policy

- Deep Learning Image Classification workload:
  - ResNet input pipeline (GCE n2-standard-8 instance)
  - ResNet50 model (4 Nvidia V100 GPU)
  - ImageNet dataset (approx. 140 GBs in GCS)



- Compare Autoscale Policy decision and Kubernetes HPA decision

Evaluation: Autoscaling Policy



Conclusion

- Data preprocessing is essential in ML workloads
- Often bottleneck causing expensive accelerator stalls
- We propose Cachew, an Input-Pipeline-as-a-Service system:
  - Autocaching and Autoscaling Policies with Multi-tenancy
- Open source: https://github.com/eth-easl/cachew
- Rich platform for future research