ESCHER – Expressive Scheduling with Ephemeral Resources
Romil Bhardwaj, Alexey Tumanov, Richard Liaw, Stephanie Wang, Philipp Moritz, Robert Nishihara, Ion Stoica
romilb@cs.berkeley.edu

Cluster schedulers need to be evolvable

- Scheduling requirements of modern distributed applications are getting increasingly complex
- E.g., Distributed training requires affinity, anti-affinity and gang scheduling – all in the same job
- Cluster frameworks must provide flexible scheduling control without the complexities of implementing a scheduler. Existing Schedulers are insufficient:

ESCHER Abstractions

Abstraction 1: Resource matching scheduler

Scheduler matches tasks resource requirements to node resource availabilities

Abstraction 2: Create resources at runtime

Frameworks provide an API for applications to create resources on nodes at runtime

Combining the Abstractions

A simple resource matching scheduler can be induced to make targeted placement decisions with short-lived ephemeral resources

- These abstractions are sufficient to allow applications to express any arbitrary scheduling policy
- Applications can use resource management to declaratively specify and execute scheduling constraints

Example - Load Balancing and Co-location

ESCHER has implementations of data locality, bin-packing, anti-affinity, soft constraints, gang scheduling, WFQ and compositions

Benchmarks

AlphaZero on ESCHER

ESCHER is 2x faster in exploring Go board states than an unaware scheduler

Performs comparably with a hard-coded policy, while requiring only 3 lines of changes

Distributed Training on ESCHER

ESCHER can live without GPUs and still perform comparably

Kubernetes MapReduce on ESCHER

ESCHER scales comparably with other workloads