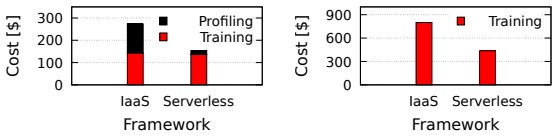
A. Bayesian Optimization based resource allocation Serverless vs IaaS

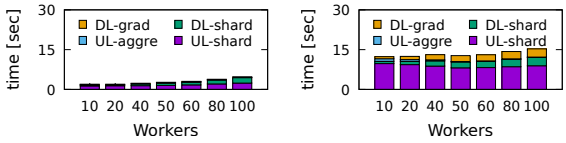


(a) Profiling and training cost using dynamic batching (b) Training cost for end-to-end training for 24 hours

Figure 1: Cost comparison for profiling and training via our Bayesian Optimizer with dynamic batching using Resnet-50 with Pytorch. Experiments using Resnet-18 with Tensorflow produced similar results.

To show how our Bayesian Optimizer can exploit serverless computing's "pay-as-you-go" pricing model, we conduct experiments for two scenarios: *dynamic batching* and *online learning*. In the case of dynamic batching, the memory size and number of workers can be scaled without redeployment and thus the profiling overhead in serverless is significantly lower than VM-based infrastructure (Figure 1a). Similarly, for online learning, due to the non-deterministic training times and continuous resource provisioning, the cost for IaaS is higher than serverless (Figure 1b)

B. Network I/O vs Storage I/O

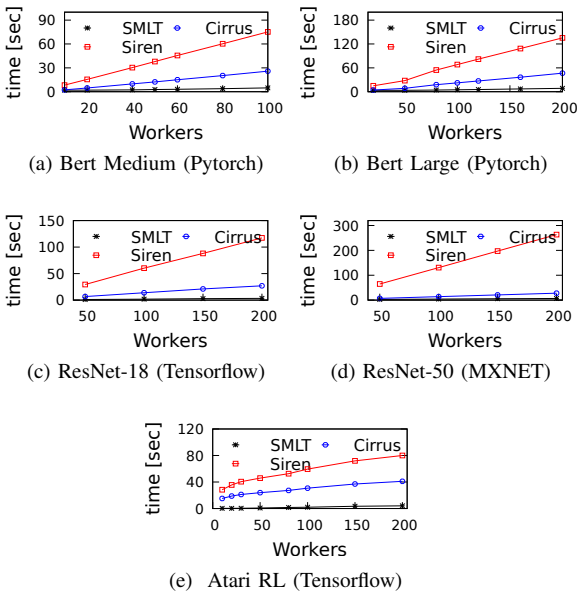


(a) Bert Medium with Redis (b) Bert Medium with S3

Figure 2: Performance Comparison of Bert Medium Redis vs S3 as an external Storage.

To demonstrate the rational behind using the in-memory key value store Redis, for our parameter store we compare the performance of network I/O vs Storage I/O. We mount S3 to our lambda function that act as a local storage which demonstrate the overhead of Storage I/O. Similarly Redis is hosted on an ECS container over a network which represents the network I/O. To reduce the randomness of the network delay, we place the ECS containers and our worker function (lambda) instances in the same virtual private cloud. Figure 2 shows the breakdown of communication steps for both Redis and S3. we denote *UL-Shard* as the time taken by the Shard Generator to shard the gradients and upload them to the storage; *DL-Shard* as the time for the Shard Aggregator to download and aggregate the shards to form the aggregated shards; *UL-aggre* as the time taken for uploading the aggregated shards, and *DL-grad* as the time for the Global Aggregator to download the aggregated shards. We can see that using Redis significantly reduces the communication time compared to using S3. Hence, from Figure 2 we can conclude that the storage I/O is significantly larger compare to network I/O.

C. Effectiveness of Hierarchical Model Synchronization



(a) Bert Medium (Pytorch) (b) Bert Large (Pytorch)

(c) ResNet-18 (Tensorflow) (d) ResNet-50 (MXNET)

(e) Atari RL (Tensorflow)

Figure 3: Per iteration communication time comparison between SMLT, *Siren* and *Cirrus*.

We evaluate the effectiveness of SMLT's proposed Hierarchical Model Synchronization by comparing the communication time between SMLT and the baselines *Siren* and *Cirrus*. Figure 3c shows the communication time as a function of the number of workers (scale-out) per iteration for all 5 benchmarks, respectively. We observe that for all three frameworks the communication time increases linearly as number of training workers increases. Such an increase in communication time is because with more workers more total gradients need to be transferred and thus have more communication demands between the workers and external storage based aggregator. However, the communication increase of SMLT is substantially lower than *Siren* and *Cirrus*, thanks to the hierarchical aggregation mechanism that significantly reduce the communication overhead in a serverless environment.

We further breakdown the communication time into individual communication steps during each training iteration in Figure 4. For *Siren* and *Cirrus*, *UL-grad* refers to the time taken by each worker to upload the gradient to the cloud storage and *DL-grad* is the time taken by the workers to download the parameters of all other workers for updating the model after every iteration. From the Figure, we can see that for both *Siren* and *Cirrus*, the main bottleneck is the *DL-grad*. Even though SMLT introduces extra steps due to the sharded approach, the overhead of

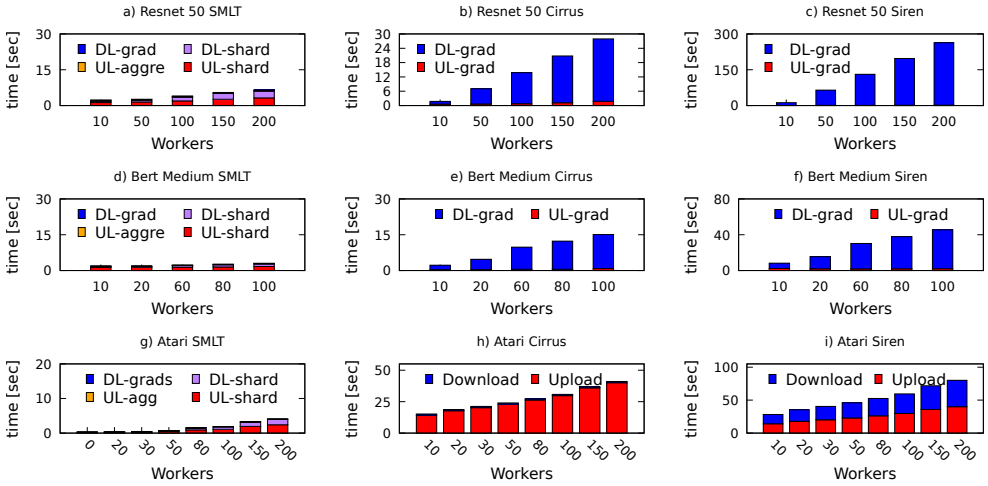


Figure 4: Communication time breakdown comparison of SMLT, Cirrus and Siren.

the extra steps is significantly lower than the overall reduction in *DL-grad*. One thing to note here is that in the case of RL based Atari model the size of upload data is significantly larger compare to the download data. We can observe the impact of larger upload data in Figure 4[g-h]. Particularly, Cirrus takes more time for uploads compare to download in the case of RL based Atari model. The longer upload time is due to the large simulation data shared by each worker after every iteration, and also reflects the limitation of a single parameter server with the increased data communication.