Leaf Technical Report

Ryan Marr

July 20, 2025

1 Concept

Leaf enables easy distributed training of machine learning models. It's designed to work on any number of machines running any operating system. Only small changes to the training code need to be made.

2 Interface

```
# Add resources and create trainer

from leaf import LeafConfig, LeafTrainer
config = LeafConfig()
config.add_server(
    server_name="",
    username="",
    hostname="",
    port=,
)
config.print_all_resources()
leaf_trainer = LeafTrainer(config)

# Register models, criterions and optimizer with leaf

model_registered = leaf_trainer.register_model(model)
criterion_registered = leaf_trainer.register_criterion(model_registered, criterion)
optimizer_registered = leaf_trainer.register_optimizer(model_registered, optimizer)
```

3 Example

```
+from leaf import LeafConfig, LeafTrainer

+config = LeafConfig()
+config.add_server(
+    server_name="gpu-server-1",
+    username="root",
+    hostname="76.71.171.219",
+    port=11111,
+)
+config.add_server(
+    server_name="gpu-server-2",
+    username="root",
+    hostname="76.71.171.219",
+    port=11111,
+)
+config.print_all_resources()
```

```
17
18 # Set device
19 device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
20
21 # Hyperparameters
22 num_epochs = 50
23 batch_size = 128
24 learning_rate = 0.001
25
26 # Data preprocessing
27 transform_train = transforms.Compose([
28     transforms.RandomCrop(32, padding=4),
29     transforms.RandomHorizontalFlip(),
30     transforms.ToTensor(),
31     transforms.Normalize((0.4914, 0.4822, 0.4465), (0.2023, 0.1994, 0.2010)),
32 ])
33
34 transform_test = transforms.Compose([
35     transforms.ToTensor(),
36     transforms.Normalize((0.4914, 0.4822, 0.4465), (0.2023, 0.1994, 0.2010)),
37 ])
38
39 # Load CIFAR-10 dataset
40 trainset = torchvision.datasets.CIFAR10(root='./data', train=True,
41                                         download=True, transform=
                                               transform_train)
42 trainloader = DataLoader(trainset, batch_size=batch_size,
43                          shuffle=True, num_workers=2)
44
45 testset = torchvision.datasets.CIFAR10(root='./data', train=False,
46                                        download=True, transform=transform_test
                                              )
47 testloader = DataLoader(testset, batch_size=batch_size,
48                         shuffle=False, num_workers=2)
49
50 # Load pretrained ResNet-50 and modify for CIFAR-10
51 model = resnet50(pretrained=True)
52 model.conv1 = nn.Conv2d(3, 64, kernel_size=3, stride=1, padding=1, bias=False)
53 model.maxpool = nn.Identity()  # Remove maxpool as CIFAR-10 images are small
54 model.fc = nn.Linear(model.fc.in_features, 10)  # Change output to 10 classes
55 model = model.to(device)
56
57 # Loss function and optimizer
58 criterion = nn.CrossEntropyLoss()
59 optimizer = optim.Adam(model.parameters(), lr=learning_rate)
60 scheduler = optim.lr_scheduler.ReduceLROnPlateau(optimizer, 'min', patience=3)
61
62 # Register models and criteria
63 +model = leaf_trainer.register_model(model)
64 +criterion = leaf_trainer.register_criterion(model, criterion)
65 +optimizer = leaf_trainer.register_optimizer(model, optimizer)
66
67 # Training loop
68 def train():
69     model.train()
70     running_loss = 0.0
71     correct = 0
72     total = 0
73
74     for batch_idx, (inputs, targets) in enumerate(trainloader):
75         inputs, targets = inputs.to(device), targets.to(device)
76
```

```
        optimizer.zero_grad()
        outputs = model(inputs)
        loss = criterion(outputs, targets)
        loss.backward()
        optimizer.step()

        running_loss += loss.item()
        _, predicted = outputs.max(1)
        total += targets.size(0)
        correct += predicted.eq(targets).sum().item()

        if (batch_idx + 1) % 100 == 0:
            print(f'Batch: {batch_idx + 1} | Loss: {running_loss/(batch_idx +
                1):.3f} | '
                  f'Acc: {100.*correct/total:.2f}%')

    return running_loss/len(trainloader), 100.*correct/total
```

4 Server Communication

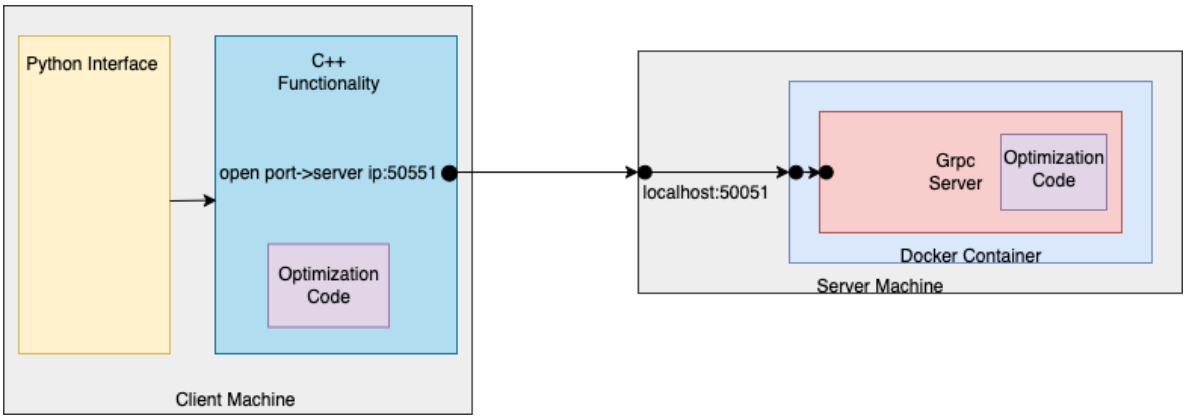

Figure 1: Server Communication Architecture

When adding a new server, leaf will add a docker container on the server and start a grpc server. When the distributed model does a forward pass, it divides the input across all servers then averages the gradient from all servers.