Systems Support for Federated Computation

Mosharaf Chowdhury
November 2021
Learning

What to Run? Where?

When to Run? How?

Analytics

Data

A Systems View of Learning and Analytics
A Systems View of Learning and Analytics

![Diagram showing the flow from Data to Application, Planning, and Execution]

Data

Application

Planning

Execution

Data
Cloud L&A

Cross-Silo FL&A

Cross-Device FL&A
Network is King!

1. Low bandwidth
2. High latency
3. Asymmetric topology
4. Dynamic variations
Application

Planning

Execution

Data

CellScope @ MobiCom’18
Fed-ensemble @ arXiv’21
Auxo
QOOP @ OSDI’18
Oort @ OSDI’21
NOCS @ SPAA’19
Terra @ arXiv’19
Sol @ NSDI’20
Flamingo
FedScale @ arXiv’21
Pando @ NSDI’20

https://github.com/SymbioticLab
Application

Planning

Execution

Data

**Oort:** Cross-Device FL&A

**Sol:** Cross-Silo FL&A

FedScale.ai

https://github.com/SymbioticLab
Sol
Fast Distributed Computation Over Slow Networks

w/ Fan Lai, Jie You, and others
NSDI'20
Latency Impact on Short Computations

- Central Coordinator
- Worker

- Tasks
- Tasks

- High latency

- Launch
- Complete

5X worse completion times for interactive analytics when running on 1ms vs 100ms networks
Bandwidth Impact on Long Computations

3X worse completion times for machine learning when running on 10Gbps vs 1 Gbps networks
Low Bandwidth
High Latency → Compute Idling
Worker

Sol in One Slide

Central Coordinator

Silo Manager

Serverless Workers

Launch (■)

Complete (■)

Launch (■)

Launch (■)

Busy

Time

Tasks

Tasks

High latency
Low bandwidth

Low latency
High bandwidth
Challenges

1. How many tasks to push?
2. When to push?
3. How to handle dependencies?
4. How to handle failures?
5. …
Large Performance Improvements

- **4X-16X** improvement in cross-silo federated learning and analytics
- **1.8X** improvement in compute utilization

Deployed across 10 silos
Baseline: Apache Spark
Workloads: TPC-DS/H and HiBench
1. Heterogeneous data
2. Heterogeneous devices
3. Enormous scale
4. Pervasive uncertainty
Oort

Efficient Federated Learning via Guided Participant Selection

w/ Fan Lai and others

OSDI’21 Distinguished Artifact
Random Client Selection Can be Suboptimal

Inefficient training when overlooking heterogeneity

- Non-IID data leads to more rounds, lower accuracy
- Heterogenous devices lead to longer rounds

No guarantees on what the sampled population is being tested

- Developer may want representative distribution
Random Selection Can be Suboptimal

OpenImage dataset with 1.6M images
14k clients; 100 per round (randomly selected)
Random Selection Can be Suboptimal

OpenImage dataset with 1.6M images
14k clients; 100 per round (randomly selected)
Time-to-Accuracy in Training

# of Rounds Taken for Target Accuracy

Avg. Round Duration

MobileNet on OpenImage dataset

Centralized

FedYoGi + Random
Oort in One Slide

1. Exploit high stat. utility clients
2. Prioritize high system utility clients

FedYoGi + Random

Centralized

FedYoGi + Oort
Challenges

1. Which clients would improve statistical efficiency?
2. How to tradeoff statistical and system efficiency?
3. How to avoid stale information at scale?
4. How to be robust against noise?
5. …
## Large Performance Improvements

<table>
<thead>
<tr>
<th></th>
<th>Stats.</th>
<th>Sys.</th>
<th>Overall</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OpenImage/MobileNet</strong></td>
<td>2.3X</td>
<td>1.5X</td>
<td>3.3X</td>
<td>+9.8%</td>
</tr>
<tr>
<td><strong>Reddit/Albert</strong></td>
<td>1.5X</td>
<td>4.9X</td>
<td>7.3X</td>
<td>+4.4%</td>
</tr>
<tr>
<td><strong>Google Speech/ResNet-34</strong></td>
<td>1.2X</td>
<td>1.1X</td>
<td>1.3X</td>
<td>+2.2%</td>
</tr>
</tbody>
</table>
FedScale.ai
Benchmarking Model and System Performance of Federated Learning at Scale

w/ Fan Lai and others
ResilientFL’21 Best Paper
arXiv’21 (2105.11367)
Missing Pieces in Existing Benchmarks

- **Systems details**
  - Network latency-bandwidth characteristics
  - End device characteristics (compute resources, battery, connectivity etc.)
  - Cloud resource characteristics
- **Scale**
  - Heterogeneity of client data
  - Availability of clients
Millions of Client Systems Traces

- Heterogeneous computation & communication speed
- Dynamics of client availability in the wild
Large Datasets and Scalable Runtime

<table>
<thead>
<tr>
<th>Category</th>
<th>Name</th>
<th>Data Type</th>
<th>#Clients</th>
<th>#Instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>CV</td>
<td>iNature</td>
<td>Image</td>
<td>2,295</td>
<td>193K</td>
</tr>
<tr>
<td></td>
<td>FEMNIST</td>
<td>Image</td>
<td>3,400</td>
<td>640K</td>
</tr>
<tr>
<td></td>
<td>OpenImage</td>
<td>Image</td>
<td>13,771</td>
<td>1.3M</td>
</tr>
<tr>
<td></td>
<td>Google Landmark</td>
<td>Image</td>
<td>43,484</td>
<td>3.6M</td>
</tr>
<tr>
<td></td>
<td>Charades</td>
<td>Video</td>
<td>266</td>
<td>10K</td>
</tr>
<tr>
<td></td>
<td>VLOG</td>
<td>Video</td>
<td>4,900</td>
<td>9.6K</td>
</tr>
<tr>
<td></td>
<td>Waymo Motion</td>
<td>Video</td>
<td>496,358</td>
<td>32.5M</td>
</tr>
<tr>
<td>NLP</td>
<td>Europarl</td>
<td>Text</td>
<td>27,835</td>
<td>1.2M</td>
</tr>
<tr>
<td></td>
<td>Blog Corpus</td>
<td>Text</td>
<td>19,320</td>
<td>137M</td>
</tr>
<tr>
<td></td>
<td>Stackoverflow</td>
<td>Text</td>
<td>342,477</td>
<td>135M</td>
</tr>
<tr>
<td></td>
<td>Reddit</td>
<td>Text</td>
<td>1,660,820</td>
<td>351M</td>
</tr>
<tr>
<td></td>
<td>Amazon Review</td>
<td>Text</td>
<td>1,822,925</td>
<td>166M</td>
</tr>
<tr>
<td></td>
<td>CoQA</td>
<td>Text</td>
<td>7,189</td>
<td>114K</td>
</tr>
<tr>
<td></td>
<td>LibriTTS</td>
<td>Text</td>
<td>2,456</td>
<td>37K</td>
</tr>
<tr>
<td></td>
<td>Google Speech</td>
<td>Audio</td>
<td>2,618</td>
<td>105K</td>
</tr>
<tr>
<td></td>
<td>Common Voice</td>
<td>Audio</td>
<td>12,976</td>
<td>1.1M</td>
</tr>
<tr>
<td>Misc ML</td>
<td>Taobao</td>
<td>Text</td>
<td>182,806</td>
<td>20.9M</td>
</tr>
<tr>
<td></td>
<td>Fox Go</td>
<td>Text</td>
<td>150,333</td>
<td>4.9M</td>
</tr>
</tbody>
</table>

FedScale can support **orders-of-magnitude** more clients on the same underlying cluster.

![Evaluation Duration Round](image)

ShuffleNet on OpenImage dataset
10 GPUs
1. Data traces
2. System traces
3. Models
4. Scale factors
5. Scalable runtime
6. Diverse backends
7. Metrics
8. …
Application → Planning → Execution ↔ Data

Research: Rethink software stacks
- Network-Aware
- Heterogeneity-Aware
- Adaptive

Service: Create evaluation platforms
- Faithful representation
- Easy to use
- Fast and scalable

CellScope@MobiCom’18
Fed-ensemble@arXiv’21
Auxo
QOOP@OSDI’18
Oort@OSDI’21
NOCS@SPAA’19
Terra@arXiv’19
Sol@NSDI’20
Flamingo
FedScale@arXiv’21
Pando@NSDI’20

https://github.com/SymbioticLab
Current PhD Students

Jae-Won Chung  Insu Jang  Fan Lai  Jiachen Liu  Hasan Al Maruf  Sanjay Singaparam  Jie You  Peifeng Yu  Yiwen Zhang

Undergraduate & Master’s

Zhezheng Chen  Yinwei Dai  Shuoren Fu  Yash Gaitonde  Songyuan Guan

Chuhenh Hu  Jack Kosaian  Qiye Li  Yang Liu  Yuze Lou

Alexander Neben  Yuqing Qiu  Wenting Tan  Yue Tan  Kaiwei Tu

Yuchen Wang  Yujia Xie  Yilei Xu  Jiaxing Yang  Yiwei Zhang

Jianchun Zhu  Jingyuan Zhu  Xiangfeng Zhu

Collaborators

Aditya Akella  Ganesh Ananthanarayan  Wei Bai  Thomas M Braun  Vladimir Braverman  Shuchi Chawla  Kai Chen  Li Chen  Asaf Cidon  Sung Won Choi  Yanhui Geng  Ali Ghodsi


Youngmoon Lee  Li Erran Li  Hongqiang Liu  Zhenhua Liu  Harsha V. Madhyastha  Kshiteej Mahajan  Barzan Mozafari  Linh Nguyen  Aurojit Panda  Manish Purohit  Junjie Qian  Kannan Ramchandran

K.V. Rashmi  Naichen Shi  Kang G. Shin  Scott Shenker  Brent Stephens  Ion Stoica  Muneesh Tewari  Xiao Sun  Mohammed Uluyol  Shivaram Venkataraman  Carl Waldspurger  Hongyi Wang

Jingfeng Wu  Sheng Yang  Biren Yi  Dong Young Yoon  Zhuolong Yu  Hong Zhang  Junxue Zhang  Yuhong Zhong  Yibo Zhu

https://github.com/SymbioticLab
Core Ideas

Reduce compute idleness in silos by redesigning both control and data planes of federated systems

1. **Sol** Control Plane
   - **Proactively push work** to workers in remote sites before they ask for additional work
   - **Decouple computation and communication** roles of tasks using serverless compute and disaggregated storage

2. **Sol** Data Plane
Performance Breakdown

16.4X improvement in cross-silo federated analytics
How to Use Oort?

1. Select subset with \(<X \) deviation from the global distribution
   
   \[
   \text{participants} = \text{oort.select_by_deviation}(\text{dev_target}, \\
   \text{range_of_capacity}, \text{total_num_clients})
   \]

2. Select \([N_1, N_2, \ldots, N_K]\) samples of categories \([C_1, C_2, \ldots, C_K]\)
   
   \[
   \text{participants} = \text{oort.select_by_category}(\text{request_list}, \\
   \text{testing_config})
   \]
## Comparison

<table>
<thead>
<tr>
<th></th>
<th>LEAF</th>
<th>FedEval</th>
<th>FedML</th>
<th>Flower</th>
<th>FedScale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heter. Client Dataset</td>
<td>![Circle]</td>
<td>![Cross]</td>
<td>![Circle]</td>
<td>![Circle]</td>
<td>![Checkmark]</td>
</tr>
<tr>
<td>Heter. System Speed</td>
<td>![Cross]</td>
<td>![Cross]</td>
<td>![Cross]</td>
<td>![Cross]</td>
<td>![Checkmark]</td>
</tr>
<tr>
<td>Client Availability</td>
<td>![Cross]</td>
<td>![Cross]</td>
<td>![Cross]</td>
<td>![Cross]</td>
<td>![Checkmark]</td>
</tr>
<tr>
<td>Scalable Platform</td>
<td>![Cross]</td>
<td>![Circle]</td>
<td>![Circle]</td>
<td>![Checkmark]</td>
<td>![Checkmark]</td>
</tr>
<tr>
<td>Flexible APIs</td>
<td>![Cross]</td>
<td>![Cross]</td>
<td>![Checkmark]</td>
<td>![Checkmark]</td>
<td>![Checkmark]</td>
</tr>
</tbody>
</table>
FAR: FedScale Automated Runtime

Scalable eval platform
- GPUs/CPUs
- High resource util.

Practical runtime
- Convergence
- System duration

---

**Metrics**
- Accuracy/loss
- Round to Acc.
- Time to Acc.
- Comm. Cost
- Comp. Cost
- ...
FAR: Easily-Deployable Benchmarking

- Flexible APIs to automatically integrate new plugins
  - Little effort to customize/benchmark new designs

<table>
<thead>
<tr>
<th>Module</th>
<th>API Name</th>
<th>Example Use Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggregator</td>
<td>round_completion_handler(*args)</td>
<td>Adaptive/secure model aggregation</td>
</tr>
<tr>
<td>Simulator</td>
<td>client_completion_handler(client_id, msg)</td>
<td>Straggler mitigation</td>
</tr>
<tr>
<td></td>
<td>push_msg_to_client(client_id, msg)</td>
<td>Model compression</td>
</tr>
<tr>
<td>Client</td>
<td>select_clients(*args)</td>
<td>Client selection</td>
</tr>
<tr>
<td>Manager</td>
<td>select_model_for_client(client_id)</td>
<td>Adaptive model selection</td>
</tr>
<tr>
<td></td>
<td>train(client_data, model, config)</td>
<td>Local SGD/malicious attack</td>
</tr>
<tr>
<td>Client</td>
<td>push_msg_to_aggregator(msg)</td>
<td>Model compression</td>
</tr>
<tr>
<td>Simulator</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Some Example APIs
FAR: Easily-Deployable Benchmarking

• Flexible APIs to automatically integrate new plugins
  • Little effort to customize/benchmark new designs

```python
from fedscale.core.client import Client

class Customized_Client(Client):
    # Customize the training on each client
    def train(self, client_data, model, conf):
        # Get the training result from
        # the default training component
        training_result = super().train(client_data, model, conf)

        # Clip updates and add noise
        secure_result = secure_impl(training_result)
        return secure_result
```

A few lines are enough for benchmarking

FedScale can benchmark more realistic statistical/system performance