ATP: In-network Aggregation for Multi-tenant Learning

Chonlam Lao*, Yanfang Le*, Kshiteej Mahajan, Yixi Chen, Wenfei Wu, Aditya Akella, Michael Swift

Tsinghua University       University of Wisconsin-Madison

* = co-primary authors
Distributed Training (PS Architecture)

Parameter Servers (PS)

Worker1 Worker2 Worker3 Worker4

Workers

Gradients

a_1 b_1 a_2 b_2 a_3 b_3 a_4 b_4
Distributed Training (PS Architecture)

Parameter Servers (PS)

PS 1

Workers

a₁ b₁
Worker1

a₂ b₂
Worker2

a₃ b₃
Worker3

a₄ b₄
Worker4

Gradients

2
Distributed Training (PS Architecture)

Parameter Servers (PS)

\[ a' = a_1 + a_2 + a_3 + a_4 \]

Workers

- Worker1: \( a_1, b_1 \)
- Worker2: \( a_2, b_2 \)
- Worker3: \( a_3, b_3 \)
- Worker4: \( a_4, b_4 \)

Gradients
Distributed Training (PS Architecture)

Parameter Servers (PS)

\[ a' = a_1 + a_2 + a_3 + a_4 \]

Workers

\[ a' \]

\[ b_1 \]

Worker1

\[ a' \]

\[ b_2 \]

Worker2

\[ a' \]

\[ b_3 \]

Worker3

\[ a' \]

\[ b_4 \]

Worker4

Gradients
Distributed Training (PS Architecture)

Network can be bottleneck for Distributed Training

Parameter Servers (PS)  \[ a' = a_1 + a_2 + a_3 + a_4 \]

Workers

Worker1  \[ a' b_1 \]
Worker2  \[ a' b_2 \]
Worker3  \[ a' b_3 \]
Worker4  \[ a' b_4 \]
Trend of In-network Computation

• Programmable switch offers in-transit packet processing and in-network state
Trend of In-network Computation

• Programmable switch offers in-transit packet processing and in-network state
Trend of In-network Computation

• Programmable switch offers in-transit packet processing and in-network state

• Reduce training time by moving gradient aggregation into the network
State-of-the-art In-network Aggregation

• SwitchML (Sapio et al. NSDI’21)
  • Target single-rack settings
State-of-the-art In-network Aggregation

- SwitchML (Sapio et al. NSDI’21)
  - Target single-rack settings
  - Support multiple jobs by static partitioning of switch resources
State-of-the-art In-network Aggregation

- SwitchML (Sapio et al. NSDI’21)
  - Target single-rack settings
  - Support multiple jobs by static partitioning of switch resources

- Short comings
State-of-the-art In-network Aggregation

• SwitchML (Sapiro et al. NSDI’21)
  • Target single-rack settings
  • Support multiple jobs by static partitioning of switch resources

• Short comings
  • Inefficiently use the switch resources
State-of-the-art In-network Aggregation

- SwitchML (Sapio et al. NSDI’21)
  - Target single-rack settings
  - Support multiple jobs by static partitioning of switch resources

- Shortcomings
  - Inefficiently use the switch resources
State-of-the-art In-network Aggregation

- SwitchML (Sonio et al. NSDI’21)
  - Target single-rack settings
  - Support multiple jobs by static partitioning of switch resources

- Shortcomings
  - Inefficiently use the switch resources
  - Does not consider multi-rack setting
State-of-the-art In-network Aggregation

- SwitchML (Sapio et al. NSDI’21)
  - Target single-rack settings
  - Support multiple jobs by static partitioning of switch resources

- Shortcomings
  - Inefficiently use the switch resources
  - Does not consider multi-rack setting

<table>
<thead>
<tr>
<th>Time</th>
<th>System</th>
<th>Number of Nodes</th>
<th>Number of V100 GPUs</th>
</tr>
</thead>
<tbody>
<tr>
<td>47 min</td>
<td>DGX SuperPOD</td>
<td>92 x DGX-2H</td>
<td>1,472</td>
</tr>
<tr>
<td>67 min</td>
<td>DGX SuperPOD</td>
<td>64 x DGX-2H</td>
<td>1,024</td>
</tr>
</tbody>
</table>
Key Goal

Speed up multiple DT jobs in a cluster while maximizing the benefits from in-network multi-switch aggregation
Outline

• Multi-tenant
• Multi-rack
• Additional challenges
  • Reliability
  • Congestion control
  • Improve floating point computation
• Evaluation
Multi-tenant: dynamic allocation
Multi-tenant: dynamic allocation

- Objective: maximize switch resource utilization
- Key idea: dynamic allocation in per-packet level
Multi-tenant: dynamic allocation

• Objective: maximize switch resource utilization
• Key idea: dynamic allocation in per-packet level
Multi-tenant: dynamic allocation

• Objective: maximize switch resource utilization
• Key idea: dynamic allocation in per-packet level
  • Randomly hash gradient packets to whole memory
Multi-tenant: dynamic allocation

- Objective: maximize switch resource utilization
- Key idea: dynamic allocation in per-packet level
  - Randomly hash gradient packets to whole memory
Multi-tenant: dynamic allocation

- Objective: maximize switch resource utilization
- Key idea: dynamic allocation in per-packet level
  - Randomly hash gradient packets to whole memory
Multi-tenant: dynamic allocation

- Objective: maximize switch resource utilization
- Key idea: dynamic allocation in per-packet level
  - Randomly hash gradient packets to whole memory
Multi-tenant: dynamic allocation

• Objective: maximize switch resource utilization
• Key idea: dynamic allocation in per-packet level
  • Randomly hash gradient packets to whole memory
Multi-tenant: dynamic allocation

- Objective: maximize switch resource utilization
- Key idea: dynamic allocation in per-packet level
  - Randomly hash gradient packets to whole memory
Multi-tenant: dynamic allocation

- Objective: maximize switch resource utilization
- Key idea: dynamic allocation in per-packet level
  - Randomly hash gradient packets to whole memory
Multi-tenant: dynamic allocation

- Objective: maximize switch resource utilization
- Key idea: dynamic allocation in per-packet level
  - Randomly hash gradient packets to whole memory
Multi-tenant: dynamic allocation

- Objective: maximize switch resource utilization
- Key idea: dynamic allocation in per-packet level
  - Randomly hash gradient packets to whole memory
Multi-tenant: dynamic allocation

• Objective: maximize switch resource utilization
• Key idea: dynamic allocation in per-packet level
  • Randomly hash gradient packets to whole memory
Challenge 1: Heavy Contention
Challenge 1: Heavy Contention

Best-effort
Challenge 1: Heavy Contention

Best-effort
Challenge 1: Heavy Contention

Best-effort
Challenge 1: Heavy Contention

Best-effort

Job 1
Worker 1 ...... Worker n PS

Job 2
Worker 1
Worker 2
......
Worker n

Job 3
Worker 1 ...... Worker n PS

Switch
Challenge 1: Heavy Contention

Best-effort
Challenge 1: Heavy Contention

Best-effort
Challenge 2: Incomplete Aggregation

Job 2
Worker 1
Worker 2
......
Worker n

Job 3
Worker 1
......
Worker n
PS

Job 1
Worker 1
......
Worker n
PS

Switch
Challenge 2: Incomplete Aggregation
Challenge 2: Incomplete Aggregation

Job 2
Worker 1
Worker 2
......
Worker n

Job 3
Worker 1
......
Worker n
PS

Job 1
Worker 1
......
Worker n
PS

Switch
Challenge 2: Incomplete Aggregation
Challenge 2: Incomplete Aggregation

Job 3
Worker 1 ...... Worker n PS

Job 1
Worker 1 ...... Worker n PS

Job 2
Worker 1
Worker 2
......
Worker n

Switch

a_1

a_2

......

a_n
Challenge 2: Incomplete Aggregation

Job 1
Worker 1 ...... Worker n PS

Job 2
Worker 1 ...... Worker n PS

Job 3
Worker 1 ...... Worker n PS

Switch

a_1

a_2 + a_3 + ... + a_n
Challenge 2: Incomplete Aggregation

Job 1
Worker 1 ...... Worker n PS

Job 2
Worker 1 ...... Worker n

Job 3
Worker 1 ...... Worker n

Switch

\[ a_1 + a_2 + a_3 + \ldots + a_n \]
Challenge 2: Incomplete Aggregation

Job 2
Worker 1  ......  Worker n  PS

Job 3
Worker 1  ......  Worker n  PS

Job 1
Worker 1  ......  Worker n  PS

Switch

\[ a_1 + a_2 + a_3 + \ldots + a_n \]
Challenge 2: Incomplete Aggregation

Job 1
Worker 1 ...... Worker n PS

Job 2
Worker 1 ...... Worker n PS

Job 3
Worker 1 ...... Worker n PS

Switch

\[ a_1 + a_2 + a_3 + \ldots + a_n \]
Challenge 2: Incomplete Aggregation

Job 1
Worker 1 ...... Worker n PS

Job 2
Worker 1 ...... Worker n PS

Job 3
Worker 1 ...... Worker n PS

\[a_1 + a_2 + \ldots + a_n + a_1\]
Inter-Rack Aggregation

• Aggregation at every layer of network topology
  • Nondeterministic routing, i.e., ECMP
Inter-Rack Aggregation

• Aggregation at every layer of network topology
  • Nondeterministic routing, i.e., ECMP
• Support two-level aggregation at ToR switches
  • Workers and PS(es) locate in different racks
Inter-Rack Aggregation

• Aggregation at every layer of network topology
  • Nondeterministic routing, i.e., ECMP
• Support two-level aggregation at ToR switches
  • Workers and PS(es) locate in different racks
Inter-Rack Aggregation

- Aggregation at every layer of network topology
  - Nondeterministic routing, i.e., ECMP
- Support two-level aggregation at ToR switches
  - Workers and PS(es) locate in different racks
Inter-Rack Aggregation

- Aggregation at every layer of network topology
  - Nondeterministic routing, i.e., ECMP
- Support two-level aggregation at ToR switches
  - Workers and PS(es) locate in different racks
Inter-Rack Aggregation

- Aggregation at every layer of network topology
  - Nondeterministic routing, i.e., ECMP
- Support two-level aggregation at ToR switches
  - Workers and PS(es) locate in different racks
  - Scale up to 1024 workers
Additional Challenges
Additional Challenges

• Rethink reliability
  • Recovery from packet loss
  • Ensure exact once aggregation
  • Memory leak: aggregators are reserved forever, but not used
Additional Challenges

• Rethink reliability
  • Recovery from packet loss
  • Ensure exact once aggregation
  • Memory leak: aggregators are reserved forever, but not used

• Rethink congestion control
  • N flows merged into one flow communication
  • Drop congestion signal, i.e., ECN
Additional Challenges

• Rethink reliability
  • Recovery from packet loss
  • Ensure exact once aggregation
  • Memory leak: aggregators are reserved forever, but not used

• Rethink congestion control
  • N flows merged into one flow communication
  • Drop congestion signal, i.e., ECN

• Improve the floating point computation
  • Convert gradients to 32-bit integer at workers by a scaling factor
  • Aggregation overflow at switch
ATP Implementation and Evaluation

• Implementation
  • Replace the networking stack of BytePS at the end host
  • Use P4 to implement the in-network aggregation service at Barefoot Tofino switch

• Evaluation
  • **Setup:** 9 servers, each with one GPU, one 100G NIC
  • **Baseline:** (BytePS + TCP, BytePS+ RDMA) x (Nto1, NtoN), SwitchML, Horovod+RDMA, Horovod+TCP
  • **Metrics:** Training Throughput, Time-to-Accuracy
  • **Workloads:** AlexNet, VGG11, VGG16, VGG19, ResNet50, ResNet101, and ResNet152
Single Job Performance

![Graph showing training throughput for various models and methods]
Single Job Performance

ATP is comparable to, and outperforms the state-of-the-art approaches. ATP gets larger performance gains on network-intensive workloads (VGG) than the computation-intensive workloads (ResNet).
Multiple Jobs: dynamic (ATP) vs static

• 3 VGG16 Jobs
• Static approach evenly distributes aggregators to jobs
• PTA: the number of the aggregators to make each job to achieve the peak aggregation throughput
Multiple Jobs: dynamic (ATP) vs static

- 3 VGG16 Jobs
- Static approach evenly distributes aggregators to jobs
- PTA: the number of the aggregators to make each job to achieve the peak aggregation throughput
Multiple Jobs: dynamic (ATP) vs static

• 3 VGG16 Jobs
• Static approach evenly distributes aggregators to jobs
• PTA: the number of the aggregators to make each job to achieve the peak aggregation throughput
Multiple Jobs: dynamic (ATP) vs static

- 3 VGG16 Jobs
- Static approach evenly distributes aggregators to jobs
- PTA: the number of the aggregators to make each job to achieve the peak aggregation throughput
Multiple Jobs: dynamic (ATP) vs static

- 3 VGG16 Jobs
- Static approach evenly distributes aggregators to jobs
- PTA: the number of the aggregators to make each job to achieve the peak aggregation throughput

When switch memory is sufficient, ATP’s dynamic ≈ static
When switch memory is insufficient, ATP’s dynamic > static
Multiple Jobs: dynamic (ATP) vs static

- 3 VGG16 Jobs
- Static approach evenly distributes aggregators to jobs

More evaluations about **packet loss recovery overhead, time-to-accuracy, congestion control** in various scenarios.

When switch memory is sufficient, ATP’s dynamic ≈ static
When switch memory is insufficient, ATP’s dynamic > static
Summary

• A network service that supports best-effort, dynamic in-network aggregation aimed at multi-rack, multi-tenant

• Co-design end-host and switch logic
  • Reliability
  • Congestion control
  • Dealing with floating point

Opensource: https://github.com/in-ATP/ATP
ATP: In-network Aggregation for Multi-tenant Learning

Chonlam Lao*, Yanfang Le*, Kshiteej Mahajan, Yixi Chen, Wenfei Wu, Aditya Akella, Michael Swift

Tsinghua University    University of Wisconsin-Madison

Thank You!

Opensource: https://github.com/in-ATP/ATP