“Data Operation In Network” (DOIN)
Use Cases

COIN Interim meeting: Sep 2022, Online
Motivation

- Recent research has shown that **network devices undertake some computing tasks** can greatly improve the overall network and application performance in some typical scenarios.

- Their implementation is mainly based on the network devices which are programmable, e.g., using **P4 or other languages**.

- We argue that for some specific scenarios (see next 3 slides) it is useful to **provide an explicit and general way** in the data/control-plane to **signal the computation to be performed**.
**NetReduce – AI/HPC Scenario**

- Existing Weak points of distributed system:
  1) The nodes among distributed nodes will generate more traffic in order to reach consensus
  2) CPU and GPU chip throughput is insufficient, therefore packet loss is caused

---

**Traditional way:**

- "push + pull" way
- PS aggregate the gradients, thus PS will suffer from in-cast issue.
- PS can easily become a bottleneck

- An all-reduce operation performs reductions on data across nodes and writes the result to each node.

**Drawbacks:**
- 1) the overhead of data transfer is doubled for non-power-of-two case;
- 2) the communication pattern involved may lead to network contention

---

The increase in the number of servers does not lead to a linear increase in server performance.
NetReduce – AI/HPC Scenario

- Comparing with the host oriented solutions, in-network aggregation approaches like SwithML[1], SHARP [2] and NetReduce[3] could potentially reduce nearly half the aggregation data by offloading gradients aggregation from the host to the network switch.
- **Switches do aggregation tasks.**

In CNN-based CV and transformer-based NLP tasks, NetReduce is **1.7x faster and 1.5x faster**, respectively, and has better scalability than ring all-reduce.


NetLock

- **Traditional**: The lock manager is a centralized point managing all the locks for a cluster. The performance of lock manager is limited to the speed of SSD/HDD.
- When the load increases, in the case of database transactions processed on a single node, the lock manager becomes a **major performance bottleneck**.
- **bottleneck**: consuming nearly 75% of transaction time.

- **For DOIN**: The lock manager can be abstracted as **Compare And Swap (CAS) or Fetch Add (FA) operations**.

The test results in NetLock[1] show that the lock manager running on a switch is able to answer 100 million requests per second, nearly 10 times more than what a lock server can do.

---

NetSequencer

- Centralized sequencers are useful building blocks for a variety of distributed systems, such as GTM, used for ordering transactions in distributed database systems.
- A centralized sequencer can be the bottleneck in high-performance distributed systems.
- **Bottleneck:** Throughput and Round-Trip Time

For DOIN:
- Switches could realize the sequencer[1] by using a "Fetch-and-Add" operation, which can easily achieve hundreds of millions of operations per seconds and latency in the order of microseconds.

Compared with Gbps-level throughput of servers, network devices have **Tbps-level throughput** and **line-rate processing capabilities**

---

In-Network Operations and Data

- The network devices cannot support complex computing operations, which otherwise might affect the forwarding performance of data plane.
- The core idea of the existing scenario is to offload some “bottleneck” computing operations to the network devices in order to improve the system performance.
- Compared with customizing the computation case by case, it is helpful to define "simple and basic" operators that could be (re)used for different scenarios.
- An explicit and general mechanism is required to tell the switch what to do and on what traffic/packets.

<table>
<thead>
<tr>
<th>Use Case</th>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NetReduce</td>
<td>Sum value (SUM)</td>
<td>The network device sums the collected parameters together and outputs the result</td>
</tr>
<tr>
<td>NetLock</td>
<td>Compare And Swap or Fetch-and-Add</td>
<td>By comparing the request value with the status of its own lock, the network device sends out whether the host has the acquired lock. Through the CAS and FA, host can implement shared and exclusive locks.</td>
</tr>
<tr>
<td></td>
<td>(CAS or FA)</td>
<td></td>
</tr>
<tr>
<td>NetSequencer</td>
<td>Fetch-and-Add (FA)</td>
<td>The network device offers a counter service and provides a monotonically increasing sequence number for the host.</td>
</tr>
</tbody>
</table>
Unified General Approach

In order to perform the DOIN operation, we need the following features:

- A DOIN Network can route the computing packets to the computing devices.
- A DOIN operation indicator that tells the device what to do.
- A structured data description allowing fast packet processing.
Next Steps

- Further diving into the use case and requirement analysis
- Initiating a list of operators that could be used for various use cases
- Starting to draft a data plane solution
- Calling for collaboration
Q & A