Towards AI-Oriented High Computing Power Data Center Network

Ennan Zhai

Director of Network Research
Alibaba Cloud
Cloud processes data via computing power
More computing power is needed as data grows
Example: Large language model training

GPT Language Model Training
175 Billion Parameters

M6 Language Model Training
1,000 Billion Parameters

<table>
<thead>
<tr>
<th>Training Type</th>
<th>Cluster Configuration</th>
<th>Training Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finetune</td>
<td>A100 80G*8</td>
<td>One week</td>
</tr>
<tr>
<td>Reward Model</td>
<td>A100 90G*24</td>
<td>One week</td>
</tr>
<tr>
<td>RL Training ChatGPT</td>
<td>A100 80G*160</td>
<td>One week</td>
</tr>
</tbody>
</table>

A100 80G*512-2048 GPUs/Task
Training time: multiple weeks
GPU bandwidth: 800Gb-1.6Tb-3.2Tb
RDMA (Remote Direct Memory Access)

Traditional Network

Server: Initiator
- Buffer
- Application
- Buffer
- Sockets
- Buffer
- Transport Protocol Driver

Server: Target
- Buffer
- Application
- Buffer
- Sockets
- Buffer
- Transport Protocol Driver

RDMA Network

Server: Initiator
- Buffer
- Application
- Buffer
- Sockets
- Buffer
- Transport Protocol Driver

Server: Target
- Buffer
- Application
- Buffer
- Sockets
- Buffer
- Transport Protocol Driver
Network performance is the bottleneck

In the large language model training, data parallelism and global gradient sync require high communication bandwidth

Lower communication speed results in a lower cluster linear expansion ratio, significantly increasing cost
Alibaba Cloud's Solutions
The essence of network performance

Network Performance = Speed of Packet Transmission

- Server and network are separate
- Server can only obtain coarse-grained network information (e.g., RTT and ECN)

- Server and network are co-designed
- Server can get fine-grained, accurate, and customizable network information

Congestion Control

Path Scheduling
The essence of network performance

- Server and network are separate
- Server can only obtain coarse-grained network information (e.g., RTT and ECN)

- Server and network are co-designed
- Server can get fine-grained, accurate, and customizable network information

Network Performance = Speed of Packet Transmission
- Congestion Control
- Path Scheduling

How we did it?
The essence of network performance

Network Performance = Speed of Packet Transmission

- Precise feedback-based congestion control
- Bandwidth-guaranteed path selection algorithm

Programmable data plane

Congestion Control
Path Scheduling
The essence of network performance

Network Performance = Speed of Packet Transmission

- Precise feedback-based congestion control
- Bandwidth-guaranteed path selection algorithm

Congestion Control
Path Scheduling

Programmable data plane

Improving network performance via end-to-end, server-network co-optimization
3 Technical Challenges & Solutions
The essence of network performance

Network Performance = Speed of Packet Transmission

- Precise feedback-based congestion control
- Bandwidth-guaranteed path selection algorithm

Programmable data plane

Improving network performance via end-to-end, server-network co-optimization
Programming data plane

The network can be programmed to run customized packet processing logics with line rates.

```plaintext
table routing {
  key = { ipv4.dstAddr : lpm; }
  actions = { drop; route; }
  size = 2048;
}
control ingress() {
  apply {
    routing.apply();
  }
}
```
Technical challenges in programming data plane

Challenge 1: Data plane is hard to program
Packet processing logic is hard to be implemented due to heterogeneous hardware constraints

Challenge 2: Data plane programming is hard to be correct
It is hard to guarantee the correctness of packet processing logic. Can we make it bug free?
Challenge 1: Data plane is hard to program
Packet processing logic is hard to be implemented due to heterogeneous hardware constraints

Lyra: A Cross-Platform Language and Compiler for Data Plane Programming on Heterogeneous ASICs

Jiaqi Gao†‡, Ennan Zhai†‡, Hongqiang Harry Liu‡, Rui Miao‡, Yu Zhou‡*, Bingchuan Tian‡*, Chen Sun‡
Dennis Cai†, Ming Zhang‡, Minlan Yu§
† Alibaba Group  ‡ Harvard University  § Tsinghua University  * Nanjing University

IR Generation

Code Analysis
(Most Challenging)

SMT Solver

Predicate Block Arrangement

Predicate Code Generation

Actual Code on Target Hardware

Lyra Code

Topology
Hardware Types

Intermediate Representations

Predicate Block DAG

Challenge 2: Data plane programming is hard to be correct
It is hard to guarantee the correctness of packet processing logic. Can we make it bug free?

Building a practical data plane formal verifier with high usability.
The essence of network performance

Network Performance = Speed of Packet Transmission

- Congestion Control
- Path Scheduling

- Precise feedback-based congestion control
- Bandwidth-guaranteed path selection algorithm

Improving network performance via end-to-end, server-network co-optimization
### HPCC: High Precision Congestion Control


*Alibaba Group*, *Harvard University*, *University of Cambridge*, *Massachusetts Institute of Technology*

<table>
<thead>
<tr>
<th>DCQCN</th>
<th>HPCC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rate-based control</td>
<td>Rate-based control + sending window</td>
</tr>
<tr>
<td>ECN-based back off</td>
<td>INT-based precise back-off</td>
</tr>
<tr>
<td>Heuristic adjustment</td>
<td>Accurate, provable adjustment</td>
</tr>
</tbody>
</table>

High throughput + stability: quickly converge for utilizing free capacity or avoiding congestion

High bandwidth + low latency: maintaining a close-to-100% utilization with a close-to-zero queue
Technical challenges in HPCC

Challenge 1: Feedback delay
Pkt/ACK may get delayed

Challenge 2: Overreaction
Different ACKs may bring overlapping feedback

- Pkt/ACK may get delayed
- Diff ACKs bring overlapping feedback

Graph showing feedback delay and overreaction:
- No congestion: Feedback within RTT
- Congestion: Feedback delayed (Can be >> T)
- Overlap indicating overreaction

Pipe volume=6
Estimating inflight bytes

Use total inflight bytes to measure congestion

- Each flow needs to estimate the total inflight bytes
- Use INT to estimate the total inflight bytes

\[
\text{total inflight bytes} \approx \frac{qlen + txRate \times T}{\text{available in INT}}
\]

Each sender estimates independently in a distributed way
The essence of network performance

Network Performance = Speed of Packet Transmission

- Precise feedback-based congestion control
- Bandwidth-guaranteed path selection algorithm

Improving network performance via end-to-end, server-network co-optimization
### Predictable vFabric on Informative Data Plane

Shuai Wang*,†, Kaihui Gao*,†, Kun Qian†, Dan Li*,†, Rui Miao†, Bo Li‡, Yu Zhou†, Ennan Zhai†, Chen Sun‡, Jiaqi Gao†, Dai Zhang†, Binzhong Fu†, Frank Kelly‡, Dennis Cai†, Hongqiang Harry Liu‡, Ming Zhang†

*Tsinghua University †Alibaba Group ‡Zhongguancun Laboratory §University of Cambridge

---

#### Burst traffic causes tail latency spikes

#### Load imbalance hinders abstraction of ideal pipe

<table>
<thead>
<tr>
<th>Systems</th>
<th>Strict Bandwidth Isolation</th>
<th>Convergence Speed</th>
<th>Low Latency</th>
<th>Multipath Awareness</th>
<th>Topology Requirement</th>
<th>Network Device Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Tenant-level Guarantee</td>
<td>Work Conservation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QJUMP[20], Chameleon[54]</td>
<td>✗</td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
<td>Priority queues</td>
</tr>
<tr>
<td>SecondNet [21], Oktopus [12], CloudMirror[39], Silo[28]</td>
<td>✓</td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
<td>None</td>
</tr>
<tr>
<td>Seawall[51], FairCloud[44]</td>
<td>✗</td>
<td>✓</td>
<td>10~200ms</td>
<td>✗</td>
<td></td>
<td>None</td>
</tr>
<tr>
<td>NetShare[38]</td>
<td>✗</td>
<td>✓</td>
<td>10~200ms</td>
<td>✗</td>
<td></td>
<td>None</td>
</tr>
<tr>
<td>ElasticSwitch[45]</td>
<td>✓</td>
<td>✓</td>
<td>10~200ms</td>
<td>✗</td>
<td></td>
<td>None</td>
</tr>
<tr>
<td>Trinity[26]</td>
<td>✓</td>
<td>✓</td>
<td>10~200ms</td>
<td>✗</td>
<td>Ideal assumption</td>
<td>Priority queues, ECN</td>
</tr>
<tr>
<td>FairCloud[44] (PS-P)</td>
<td>✓</td>
<td>✓</td>
<td>10~200ms</td>
<td>✗</td>
<td>Ideal assumption</td>
<td>Per-tenant queue</td>
</tr>
<tr>
<td>Hadrian[13]</td>
<td>✓</td>
<td>✓</td>
<td>5~10ms</td>
<td>✗</td>
<td>Ideal assumption</td>
<td>Tree topology</td>
</tr>
<tr>
<td>Proteus[60], EyeQ[29], HUG[16], GateKeeper[49]</td>
<td>✓</td>
<td>✓</td>
<td>5~10ms</td>
<td>✗</td>
<td>Ideal assumption</td>
<td>Programmable</td>
</tr>
<tr>
<td>PicNIC[37]</td>
<td>✓</td>
<td>Partial</td>
<td>5~10ms</td>
<td>✗</td>
<td>Host side</td>
<td>Congestion/loss-free fabric</td>
</tr>
<tr>
<td>µFAB (Our work)</td>
<td>✓</td>
<td>✓</td>
<td>&lt;1ms</td>
<td>✓</td>
<td>✓</td>
<td>None</td>
</tr>
</tbody>
</table>

---

No guarantee on latency

No guarantee on bandwidth
**Predictable vFabric on Informative Data Plane**

Shuai Wang††, Kaihui Gao†, Kun Qian†, Dan Li††, Rui Miao†, Bo Li†, Yu Zhou†, Ennan Zhai†, Chen Sun†, Jiaqi Gao†, Dai Zhang†, Binzhang Fu†, Frank Kelly‡, Dennis Cai†, Hongqiang Harry Liu†, Ming Zhang†

*Tsinghua University †Alibaba Group ‡Zhongguancun Laboratory §University of Cambridge

---

**Bandwidth guarantee**  A virtual fabric quickly provisions the guaranteed bandwidth if traffic demand is sufficient

**Bounded tail latency**  End-to-end latency between vNICs is always bounded to a low level

---

**Challenge 1: Hard to achieve latency guarantees**

Existing CC heuristically evolves flow rate to achieve full network utilization

---

**Challenge 2: Hard to achieve bandwidth guarantees**

Bandwidth isolation in multi-path architectures is not orthogonal to load balancing
Reconstructing DCN service with programmability

uFAB-E:
- Send probes with fetch core information back
- Schedule packets to paths
- Control sending rate of each path

uFAB-C:
- Extract VF information in probes
- Summarize total bandwidth subscription on each link
- Piggyback the preceding information with INT

Sub-ms to perceive the path’s quality and converge to demanded bandwidth
Reconstructing DCN service with programmability

- Send probes with fetch core information back
- Schedule packets to paths
- Control sending rate of each path

- Extract VF information in probes
- Summarize total bandwidth subscription on each link
- Piggyback the preceding information with INT

Sub-ms to perceive the path’s quality and converge to demanded bandwidth

Proportional sharing:

\[ r = \frac{\phi}{\sum \phi} \times C \]

What if some senders have insufficient traffic?

\[ R = \frac{\phi}{\sum \phi} \times \sum R \times C \]

\( r \) serves as the lower bound, \( R \) serves as the expected rate

Deterministic rate adjustment & path selection
The essence of network performance

Network Performance = Speed of Packet Transmission

- Precise feedback-based congestion control
- Bandwidth-guaranteed path selection algorithm

Programmable data plane

Improving network performance via end-to-end, server-network co-optimization
The essence of network performance

Network Performance = Speed of Packet Transmission

- Precise feedback-based congestion control
- Bandwidth-guaranteed path selection algorithm

Programmable data plane

Reliability/stability problems are critical

Improving network performance via end-to-end, server-network co-optimization
Simulation-based verification for cluster updates

Network updating is dangerous: the majority of outages in our AI cluster was caused by updates.
Example: Vendor-specific behavior

- Router R1
  - AS: 100
  - Network: 10/8, 20/8
  - Peers: R2
  - R1 to R2 egress
  - Policy:
    - If any: add community 920

- Router R2
  - AS: 200
  - Peers: R1, R3
  - R2 to R3 ingress
  - Policy:
    - If prefix == 20/8: add community 920

- Router R3
  - AS: 300
  - Peers: R2, R4
  - R3 to R4 ingress
  - Policy:
    - If community != 920: deny

- Router R4
  - AS: 400
  - Peers: R3
  - R3 to R4 ingress
  - Policy:
    - If community != 920: deny

BGP updates:
- 10/8, 20/8

<table>
<thead>
<tr>
<th>Prefix</th>
<th>AS Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>10/8</td>
<td>300, 200, 100</td>
</tr>
<tr>
<td>20/8</td>
<td>300, 200, 100</td>
</tr>
</tbody>
</table>
Example: Vendor-specific behavior

- **R1** (Vendor A)
  - router r1
  - as 100
  - nw 10/8
  - nw 20/8
  - peer r2
  - r1 to r2 egress
  - policy r1 to r2:
    - if any:
      - add community 920

- **R2** (Vendor B)
  - router r2
  - as 200
  - peer r1
  - peer r3

- **R3** (Vendor A)
  - router r3
  - as 300
  - peer r2
  - r2 to r3 ingress
  - peer r4
  - policy r2 to r3:
    - if prefix == 20/8:
      - add community 920

- **R4** (Vendor A)
  - router r4
  - as 400
  - peer r3
  - r3 to r4 ingress
  - policy r3 to r4:
    - if community != 920:
      - deny

**BGP updates**
- 10/8, 20/8

**Vendor-specific behavior**
- Remove community

**Prefix** | **AS Path**
---|---
20/8 | 300, 200, 100

... | ...

... | ...

**Prefix** | **AS Path**
---|---
20/8 | 300, 200, 100

... | ...

... | ...
How to address the model faithfulness challenge?

**Key insight:**
- Using production network RIBs as a reference to debug simulated model
How to address the model faithfulness challenge?

**Key insight:**
- Using production network RIBs as a reference to debug simulated model

**Strawman solution:**
- Localizing vendor-specific behaviors by comparing real and simulated RIBs

---

Real-world routing tables

<table>
<thead>
<tr>
<th>R1</th>
<th>R2</th>
<th>R3</th>
<th>R4</th>
</tr>
</thead>
<tbody>
<tr>
<td>prefix</td>
<td>as</td>
<td>path</td>
<td>prefix</td>
</tr>
<tr>
<td>10/8</td>
<td>i</td>
<td></td>
<td>10/8</td>
</tr>
<tr>
<td>20/8</td>
<td>i</td>
<td></td>
<td>20/8</td>
</tr>
</tbody>
</table>

Simulated routing tables

<table>
<thead>
<tr>
<th>R1</th>
<th>R2</th>
<th>R3</th>
<th>R4</th>
</tr>
</thead>
<tbody>
<tr>
<td>prefix</td>
<td>as</td>
<td>path</td>
<td>prefix</td>
</tr>
<tr>
<td>10/8</td>
<td>i</td>
<td></td>
<td>10/8</td>
</tr>
<tr>
<td>20/8</td>
<td>i</td>
<td></td>
<td>20/8</td>
</tr>
</tbody>
</table>
How to address the model faithfulness challenge?

Key insight:
- Using production network RIBs as a reference to debug simulated model

Strawman solution:
- Localizing vendor-specific behaviors by comparing real and simulated RIBs

<table>
<thead>
<tr>
<th></th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
<th>R4</th>
</tr>
</thead>
<tbody>
<tr>
<td>prefix</td>
<td>as path</td>
<td>prefix</td>
<td>as path</td>
<td>prefix</td>
</tr>
<tr>
<td>10/8</td>
<td>i</td>
<td>10/8</td>
<td>100</td>
<td>10/8</td>
</tr>
<tr>
<td>20/8</td>
<td>i</td>
<td>20/8</td>
<td>100</td>
<td>20/8</td>
</tr>
</tbody>
</table>

Real-world routing tables

However, using normal RIBs cannot localize the correct VSBs!

Simulated routing tables

Diff
How to address the model faithfulness challenge?

Key insight:
• Using production network RIBs as a reference to debug simulated model

Our solution:
• We combine all the attributes of a route relevant for routing into an extended RIB
• The first mismatch between real and simulated ext-RIBs is vendor-specific behavior
How to address the model faithfulness challenge?

Key insight:
• Using production network RIBs as a reference to debug simulated model

Our solution:
• Hoyan combines all the attributes of a route relevant for routing into an extended RIB
• The first mismatch between real and simulated ext-RIBs is vendor-specific behavior
Device behavior model tuner

• Eight VSBs detected
  - Default ACL, route policy, removing private AS, etc.;
  - VSB localization within 10 lines of configuration

![Diagram showing CDF against Prefix accuracy (%): Pre-depl. of tuner and 2 Months after]
AI-Oriented Network in Alibaba Cloud

Serverless PAI PasS

PaaS

IaaS

Infrastructure Operation
4 Future Directions
Specialized hardware is the key enabler

- **Switch**
  - High throughput

- **SoC SmartNIC**
  - Flexible

- **FPGA SmartNIC**
  - Customizable

Diagram showing:
- Network bandwidth & Data growth
- Compute-Network Gap
- CPU performance

- 10s Tb/s
- 200 – 400 Gb/s
- 200 Gb/s
- 50 Gb/s
Embrace heterogeneity

Deploy network functions on all programmable devices in the entire cloud network
Embrace heterogeneity and build uniform interface

Developers only interact with the uniform interface and ignore the hardware details

Lyra: A Cross-Platform Language and Compiler for Data Plane Programming on Heterogeneous ASICs (SIGCOMM’20)

Vela: Host-Side Uniform Programming Platform for Network Processing
5 Conclusion
• Server-network co-design end-to-end performance improvement

• High-performance hardware programming to fundamentally accelerate AI training

The era of AI has arrived!

It is the last word to develop cutting-edge technology and stand at the commanding heights of technology