ClickINC: In-network Computing as a Service in Heterogeneous Programmable Data-center Networks

Wenquan Xu, Zijian Zhang, Yong Feng, Haoyu Song, Zhikang Chen, Wenfei Wu, Guyue Liu, Bin Liu
Caveats

- INC in academia ≠ COIN in IETF
- Mostly boring details except the motivation and high-level ideas
The Evolution of Networking

Network: a dumb pipe

- Data plane flexibility
- Memory + calculation
- Line-rate packet processing

Can network help with computation?

Programmable ASIC, FPGA, NP, SmartNIC ...

Data plane programmable switch
Prevalent INC Applications

- **Key-value store**\(^{[1]}\)
- **DDoS attack defense**\(^{[2]}\)
- **ML parameter aggregation**\(^{[3]}\)
- **SQL query acceleration**\(^{[4]}\)
- **ML model inference**\(^{[5]}\)

1. Developers need to program their own INC from scratch:
   - INC is strongly coupled to the devices, hard to generalize
   - Different apps have different performance demands & data characteristics
   - Hard to reuse

2. INC developer needs also be the network operator:
   - Needs to develop a complete forwarding/processing program
   - Takes care of network details:
     - Packet parsing
     - Protocol handling
     - Correctness of forwarding rules
   - Closely involved in the deployment and network operation

3. INC development is challenging:
   - Heterogeneous devices: architecture, resources, language
   - Complex topology: especially when multiple paths are supported
   - Device resource and capability limitations:
## Related Works

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPDK [Intel]</td>
<td>Y: unify program interface</td>
</tr>
<tr>
<td>DOCA [Nvidia]</td>
<td>N: cross-device program orchestration</td>
</tr>
<tr>
<td>Flightplan [NSDI’ 20]</td>
<td>Y: program orchestration on heter. devices</td>
</tr>
<tr>
<td></td>
<td>N: automatic program partition</td>
</tr>
<tr>
<td>Lyra [SIGCOMM’ 20]</td>
<td>Y: automatic program partition</td>
</tr>
<tr>
<td></td>
<td>N: large-scale deployment, smartNICs</td>
</tr>
<tr>
<td>ClickINC [SIGCOMM’ 23]</td>
<td>Decouples network operation with INC developing</td>
</tr>
<tr>
<td></td>
<td>Isolates development of different INCs</td>
</tr>
<tr>
<td></td>
<td>Automates incremental deployment and support large-scale scenario</td>
</tr>
</tbody>
</table>

- For network operator
  - Monolithic program: limit to single user
  - Low-level abstraction
  - Small-scale deployment
**INC as a Service**

With the goal of deploying INC in one click

- Unified data plane: OBD
- Conceal device, topology, language, etc.
- Decouple network and INC
- INC program as plugins
Programming API

User Programming:

• Python-like language

• Three modes:
  • Template configuration
  • Modular programming
  • Advanced user-define programming
ClickINC Architecture

- **Compiler:**
  - Convert program to IR
  - Translate IR to target platform

- **Allocator:**
  - Place program on devices

- **Manager:**
  - Merge INC into main program
  - Remove INC program
  - Update resources
INC Program Placement

Map a program to pipeline stage/cores on one or more devices:
- Problem modeling: ILP or SMT
  - Constraints: 1) topology; 2) hardware limitation; 3) resource size
  - Targets: 1) serving traffic; 2) occupied resources; 3) across-device overhead
- Current solutions: ILP solver (P4all), SMT solver (Lyra)
  - Inapplicable in large-scale scenario and for multi-path traffic

Pruned-based Dynamic programming:
- Simplifying topology for fat-tree and leaf-spine topology
- Dynamic programming on each sub-tree, and combine the solutions
- Pruning method: reduce searching space
INC Program Management

- Merge INC to the main program:
  - Graph based method
  - Add annotations

- Remove INC:
  - By annotations
**Emulator & testbed**

- Tofino: bf-sde
- Trident4: BCM-SIM
- FPGA/NFP smartNIC: behavior model

- DPDK server
- Tofino switches
- SN1000: FPGA smartNIC
Experiment Results

ClickINC makes use of resources on heterogeneous and multiple devices

Modular programming abstraction allows more efficient INC development

Less placement time with equal optimality solution compared to SMT solver
Experiment Results

Strong scalability to the number of devices

Incremental deployment has less affected traffic, device and other INCs
Conclusion

Contributions:
1. Propose the concept of INC as a Service
2. Top-down framework for program developing and deploying: One big INC abstractions, agile programming model, efficient program placement algorithm.
3. Heterogeneous-device emulation and testbed

On-going researches:
1. Automatically set parameters for user-written programs
2. Expand the template for common INC applications
3. More kinds of programmable network devices will be supported
4. Placement algorithm will be improved to support more complex topology and traffic scenarios