ParaGraph: An application-simulator interface and toolkit for hardware-software co-design

Mikhail Isaev[1], Nic McDonald[2],
Jeff Young[1], Rich Vuduc[1]

[1]Georgia Institute of Technology, [2]Nvidia

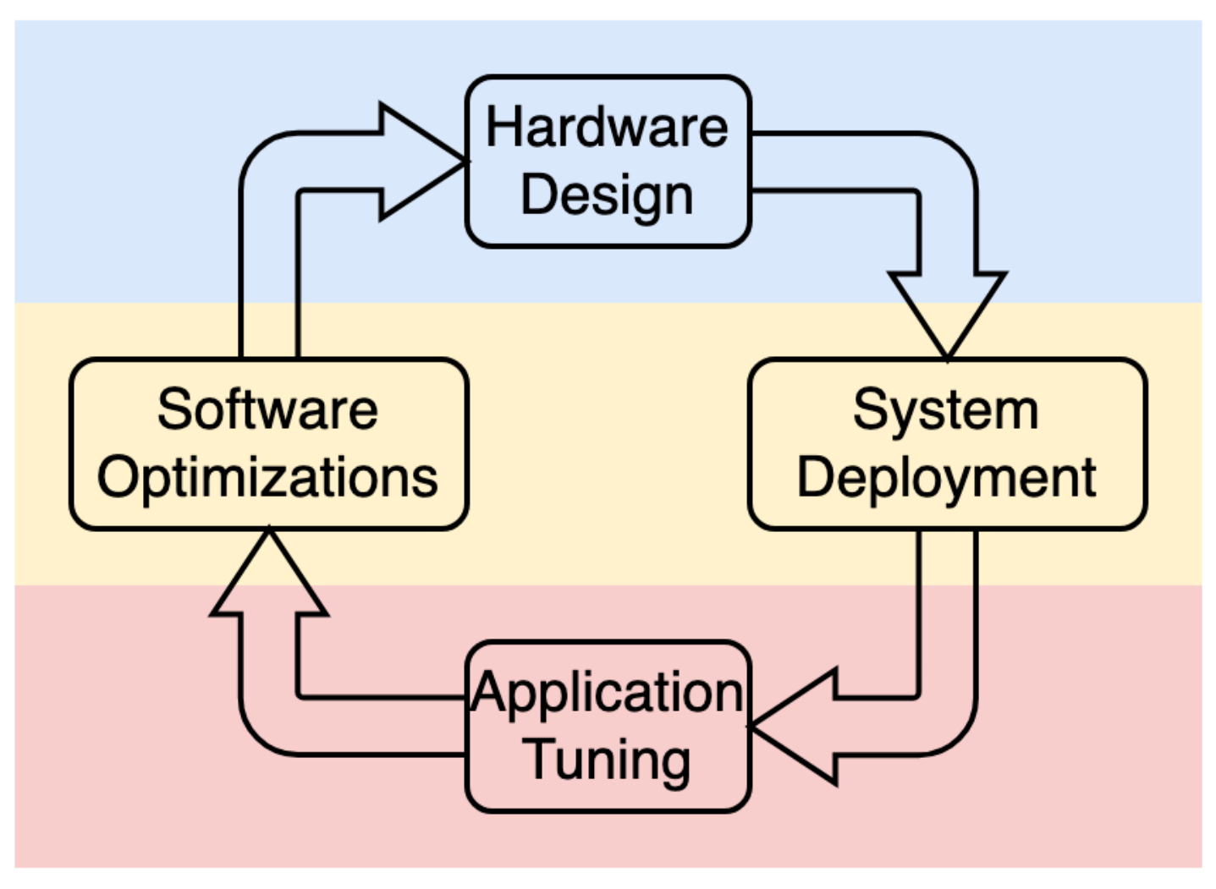
Research Problem

System co-design crosses many domain boundaries

Most HW/SW co-design studies fall under one of the categories:

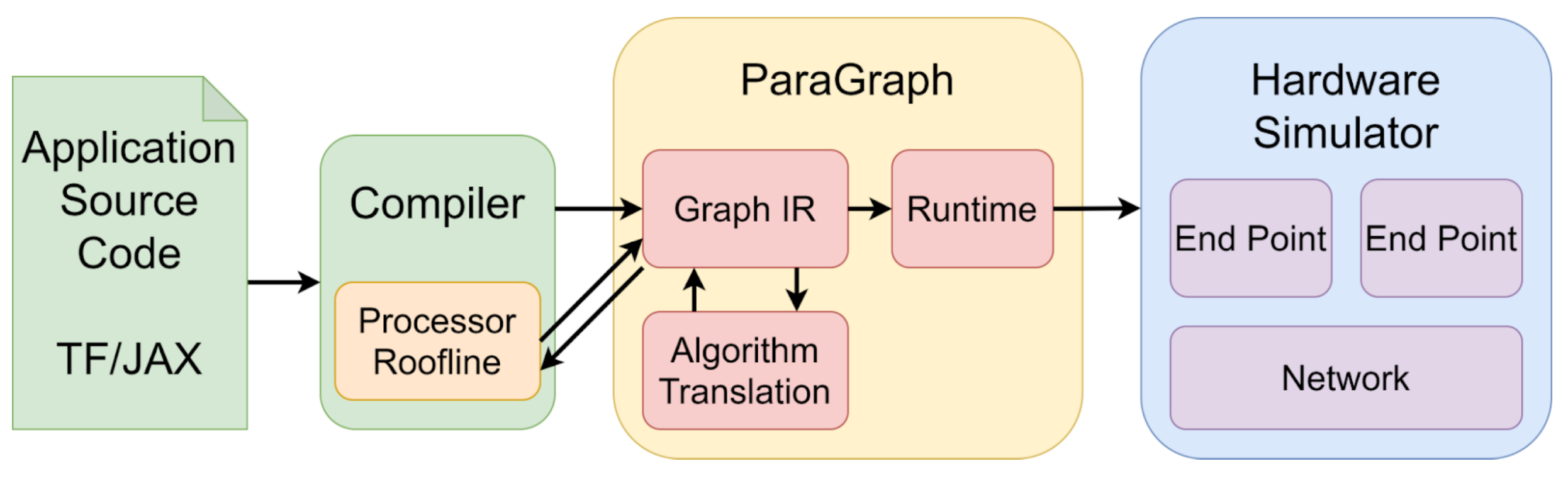
- Optimizing **software** on **existing hardware**;
- Designing **future hardware** systems with limited and **simplistic application** models.

Lack of infrastructure to model both future **hardware AND applications** with appropriate fidelity



System development cycle crosses **hardware**, **software**, and **application** domains

ParaGraph - our take on HW/SW co-design toolchain



ParaGraph – a toolchain to support HW/SW co-design of future large-scale systems built with accelerated hardware for distributed high-performance computing and deep learning applications

ParaGraph decouples workload modeling from the underlying simulation infrastructure

ParaGraph Workflow

(a) Pseudocode

```
import tensorflow as tf
input, target = load_data(64)
model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(1024,)),
  tf.keras.layers.Dense(10)
])
loss_fn = tf.keras.losses.MeanAbsoluteError()
model.compile(loss=loss_fn)
model.fit(input, target)
```

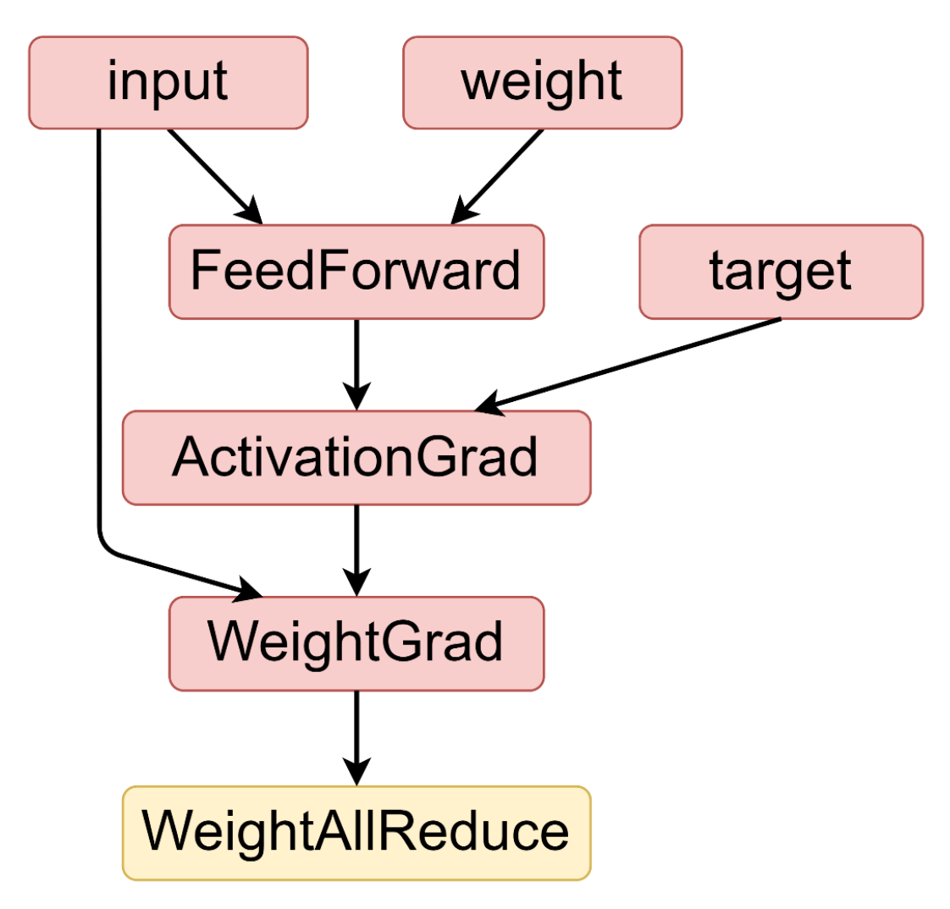
(b) HLO IR

```
%input.1 = f32[64,1024]{0,1} parameter(0)
%weight.2 = f32[1024,10]{1,0} parameter(1)
%target.3 = f32[64,10]{1,0} parameter(2)
%FeedForward.4 = f32[64,10]{1,0} dot(f32[64,1024]{0,1} %input.1, f32[1024,10]{1,0} %weight.2),
        lhs_contracting_dims={0}, rhs_contracting_dims={1}
%ActivationGrad.5 = f32[64,10]{1,0} add(f32[64,10]{1,0} %FeedForward.4, f32[64,10]{1,0} %target.3)
%WeightGrad.6 = f32[1024,10]{1,0} dot(f32[64,1024]{0,1} %input.1, f32[64,10]{1,0} %ActivationGrad.5),
        lhs_contracting_dims={0}, rhs_contracting_dims={0}
%WeightAllReduce.7 = f32[1024,10]{1,0} all-reduce(f32[1024,10]{1,0} %WeightGrad.6),
        replica_groups={{0,1,2}}
```

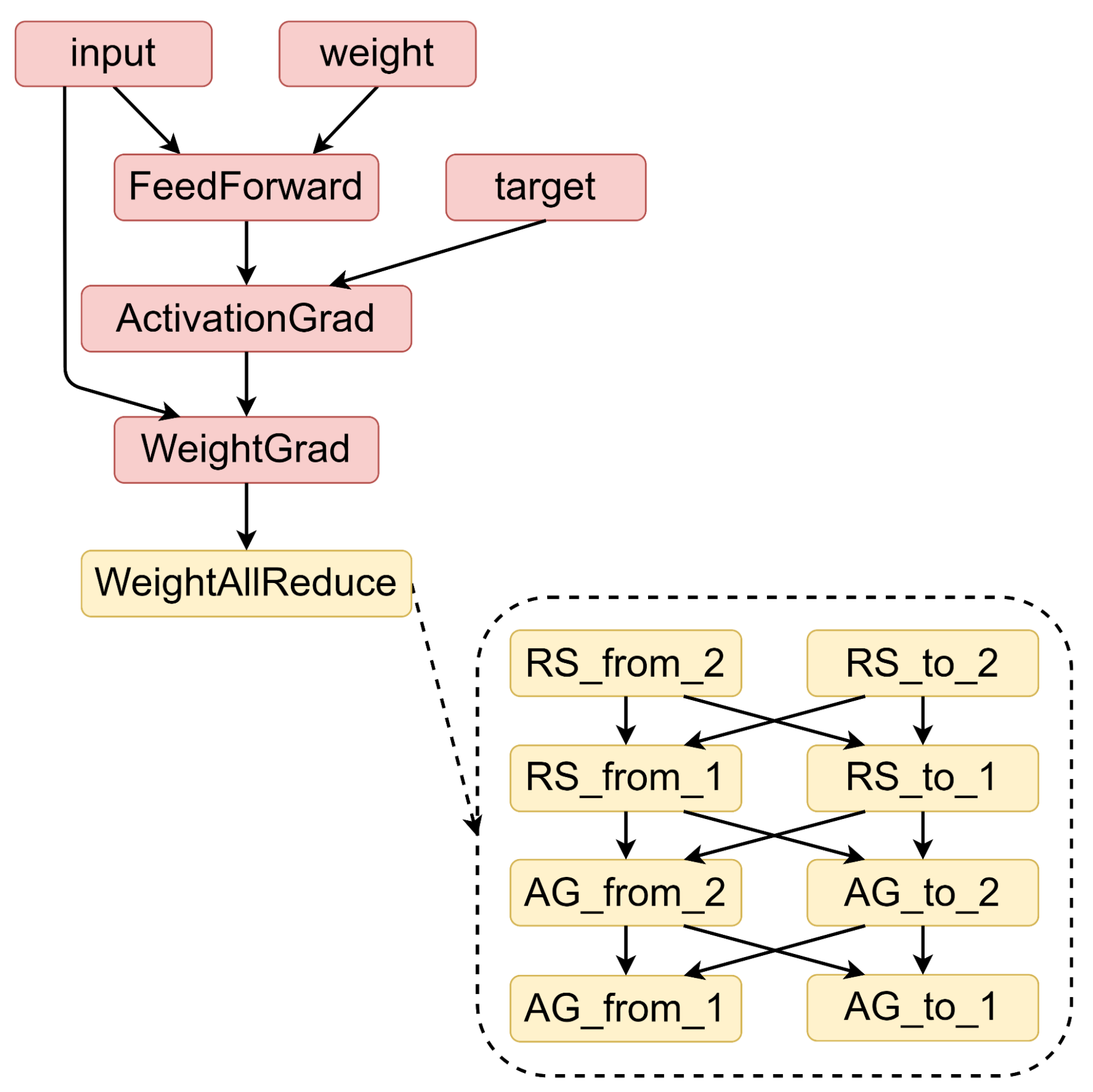
(c) ParaGraph IR

```
%input.1 = delay(), bytes_out=262144, seconds=0.250
%weight.2 = delay(), bytes_out=40960, seconds=0.039
%target.3 = delay(), bytes_out=2560, seconds=0.002
%FeedForward.4 = delay(%input.1, %weight.2), bytes_in=303104,
        bytes_out=2560, flops=1310720, seconds=1.25
%ActivationGrad.5 = delay(%FeedForward.4, %target.3), bytes_in=5120,
        bytes_out=2560, flops=640, seconds=0.007
%WeightGrad.6 = delay(%input.1, %ActivationGrad.5), bytes_in=264704,
        bytes_out=40960, flops=655360, seconds=0.625
%WeightAllReduce.7 = all-reduce(%WeightGrad.6), bytes_out=40960
        communication_groups={{0,1,2}}
```

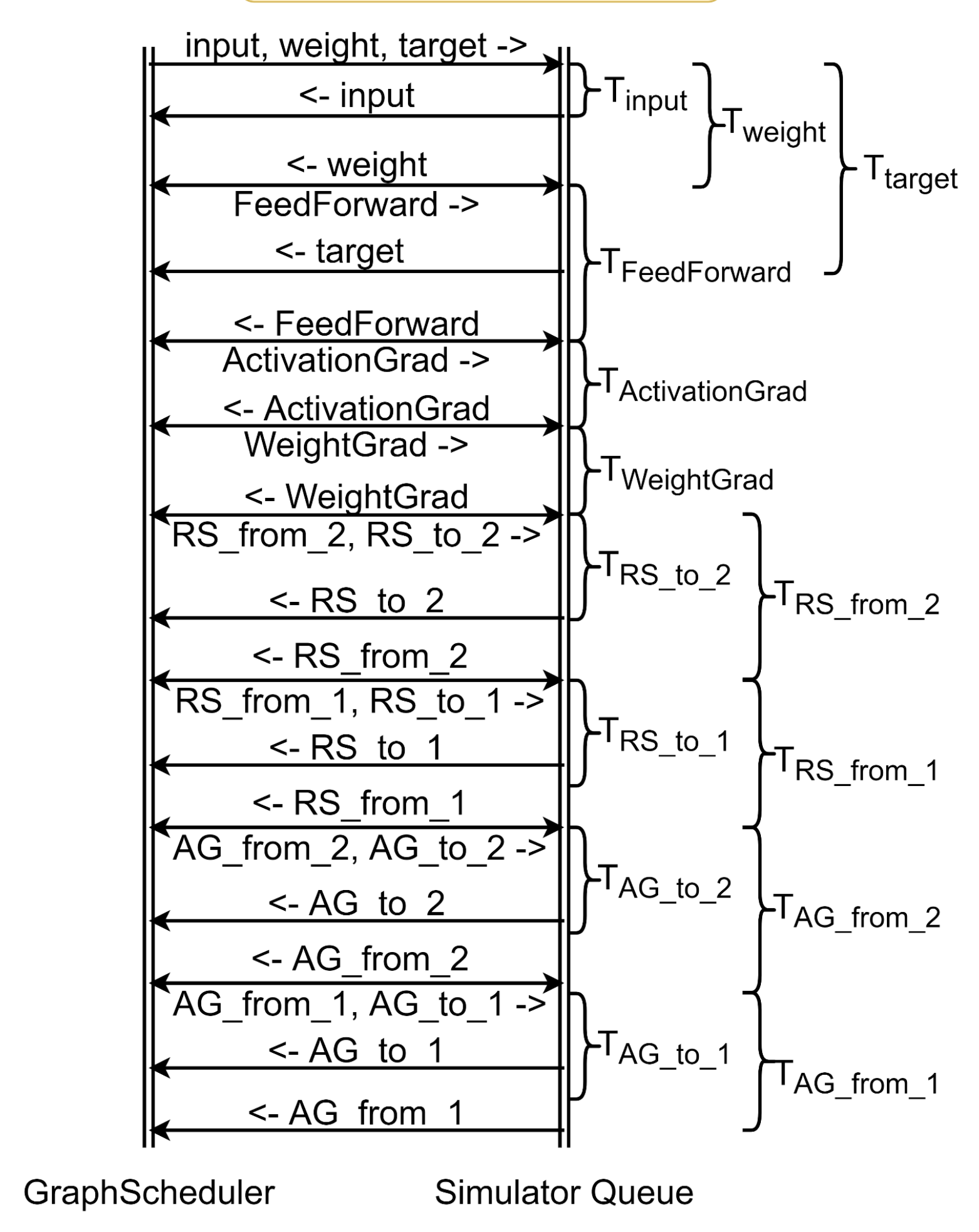
(d) ParaGraph/HLO graph



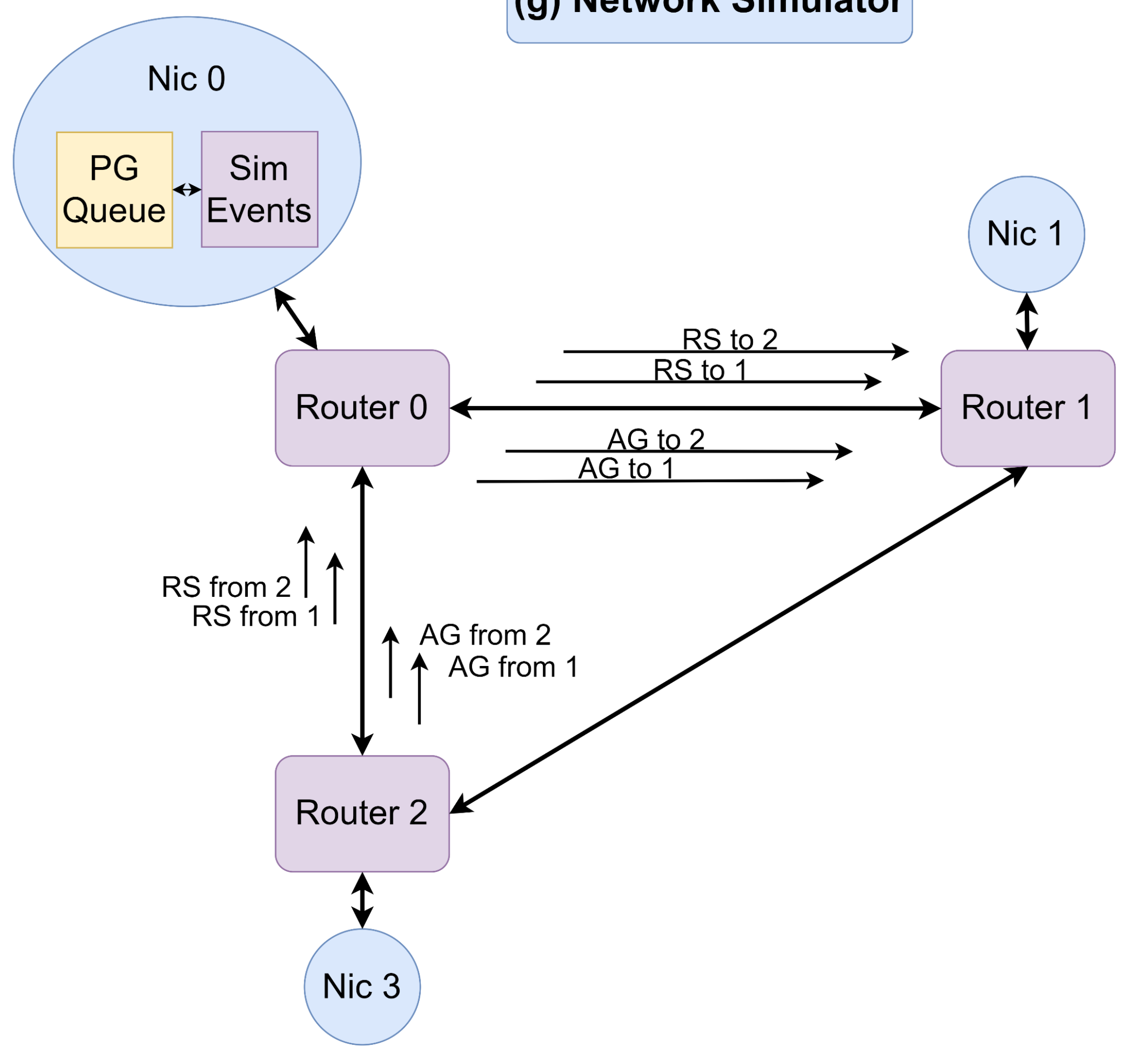
(e) ParaGraph graph after translation



(f) ParaGraph API



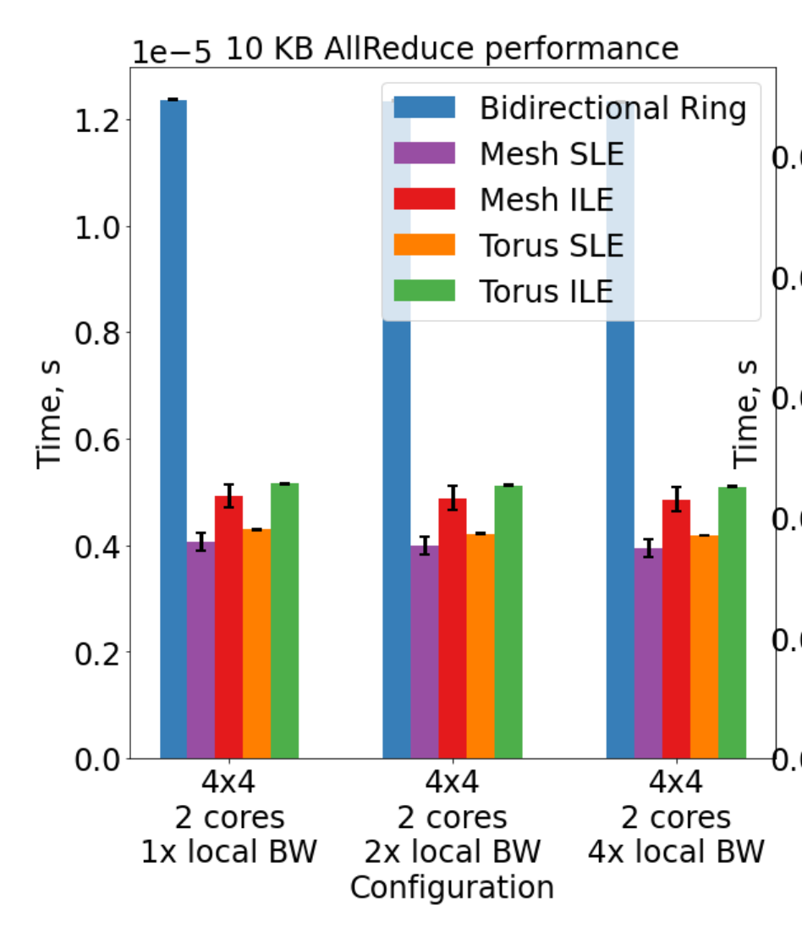
(g) Network Simulator

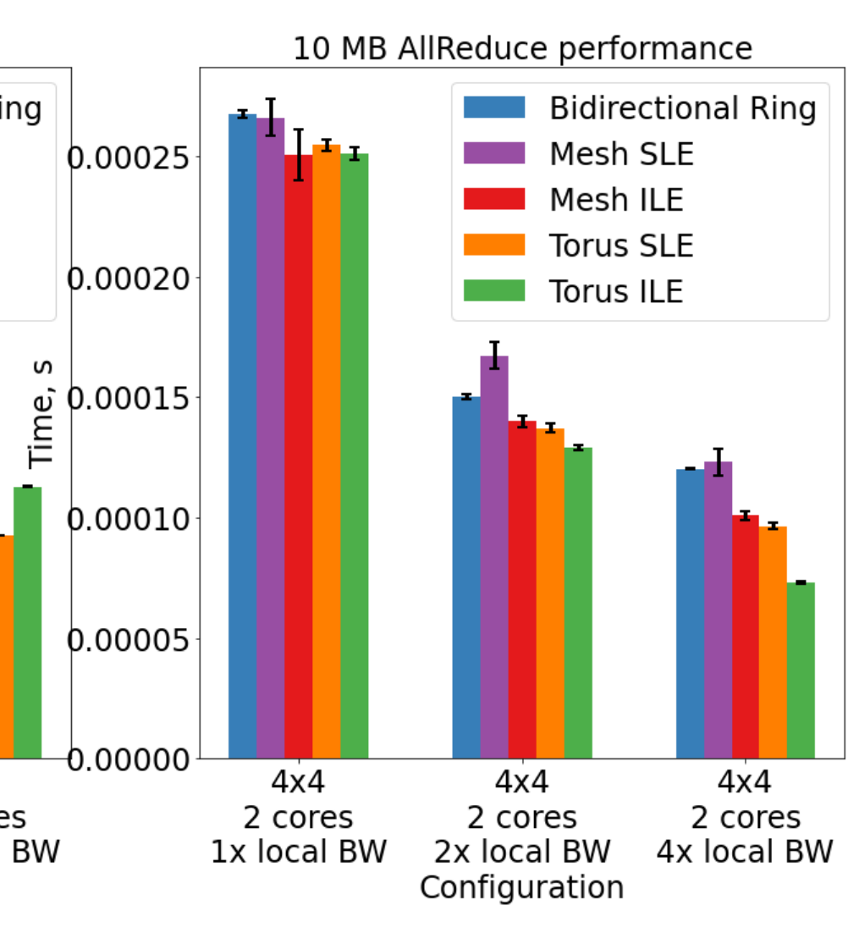


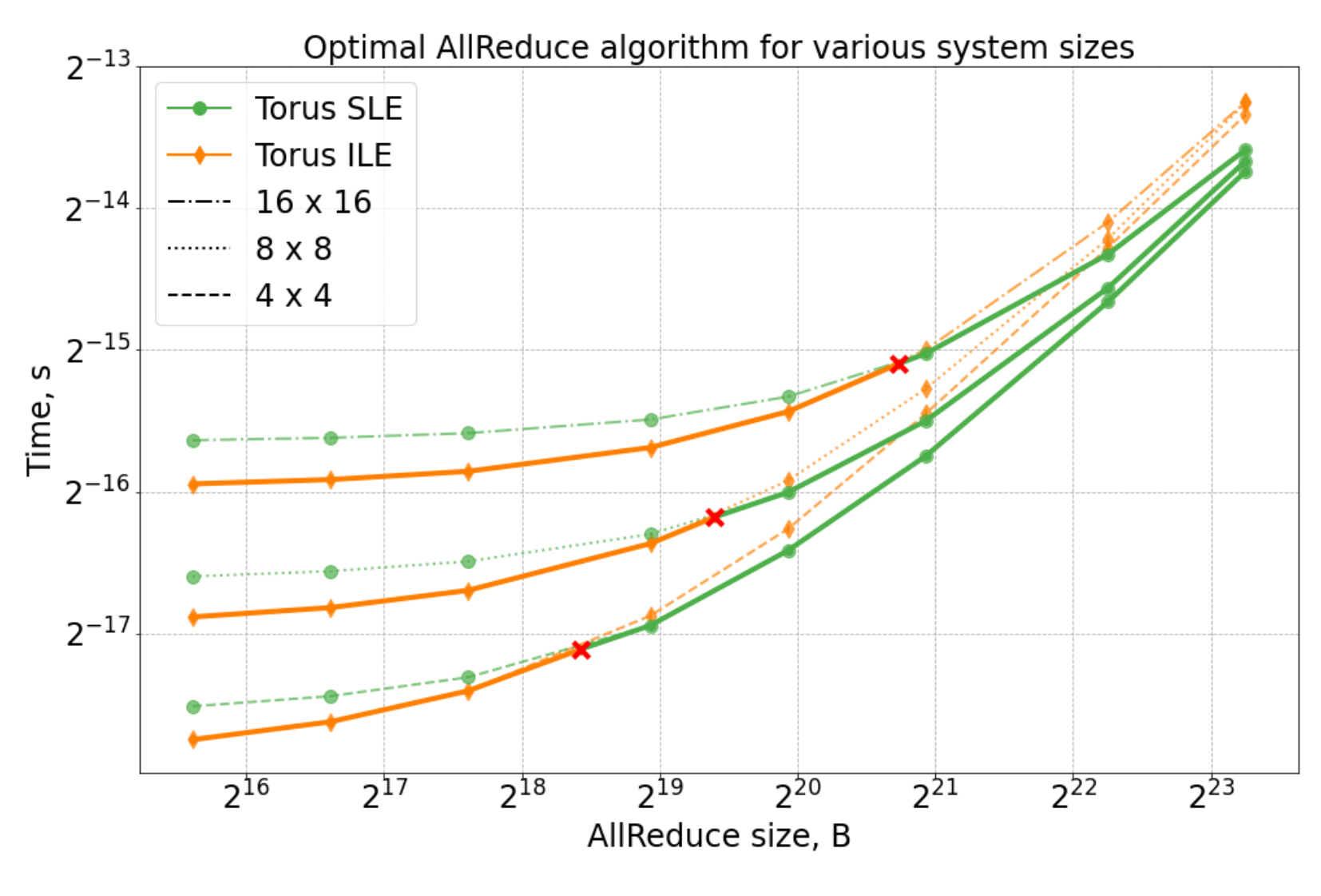
Case Study 1: optimal AllReduce search

ParaGraph allows SW engineers to model system-level SW, such as communication libraries, before system deployment

With ParaGraph we navigate landscape of SW and HW parameters simultaneously







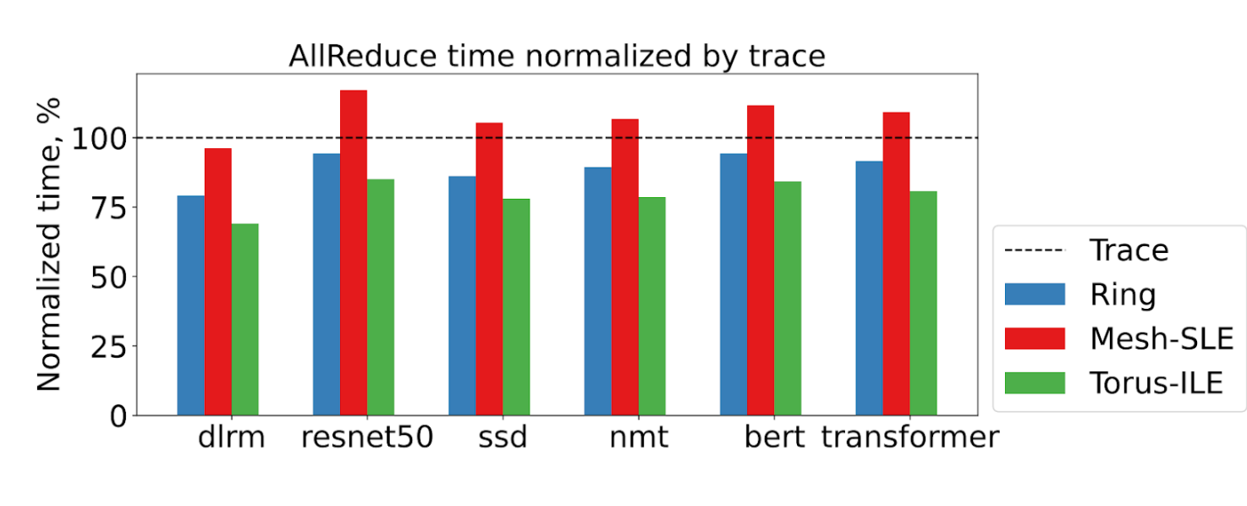
Case Study 2: model accuracy analysis

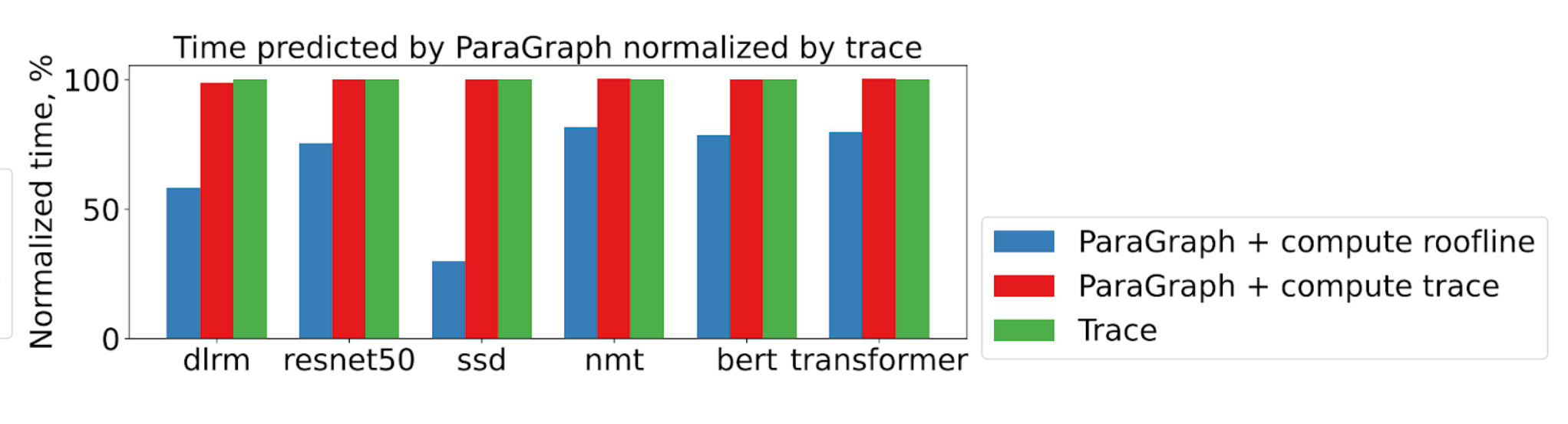
ParaGraph provides high fidelity application models to HW engineers

Validated against MLPerf training v0.7 trace from 64-cores TPUv3 system

Network DES model matches the trace

Compute roofline provides longer modeled time

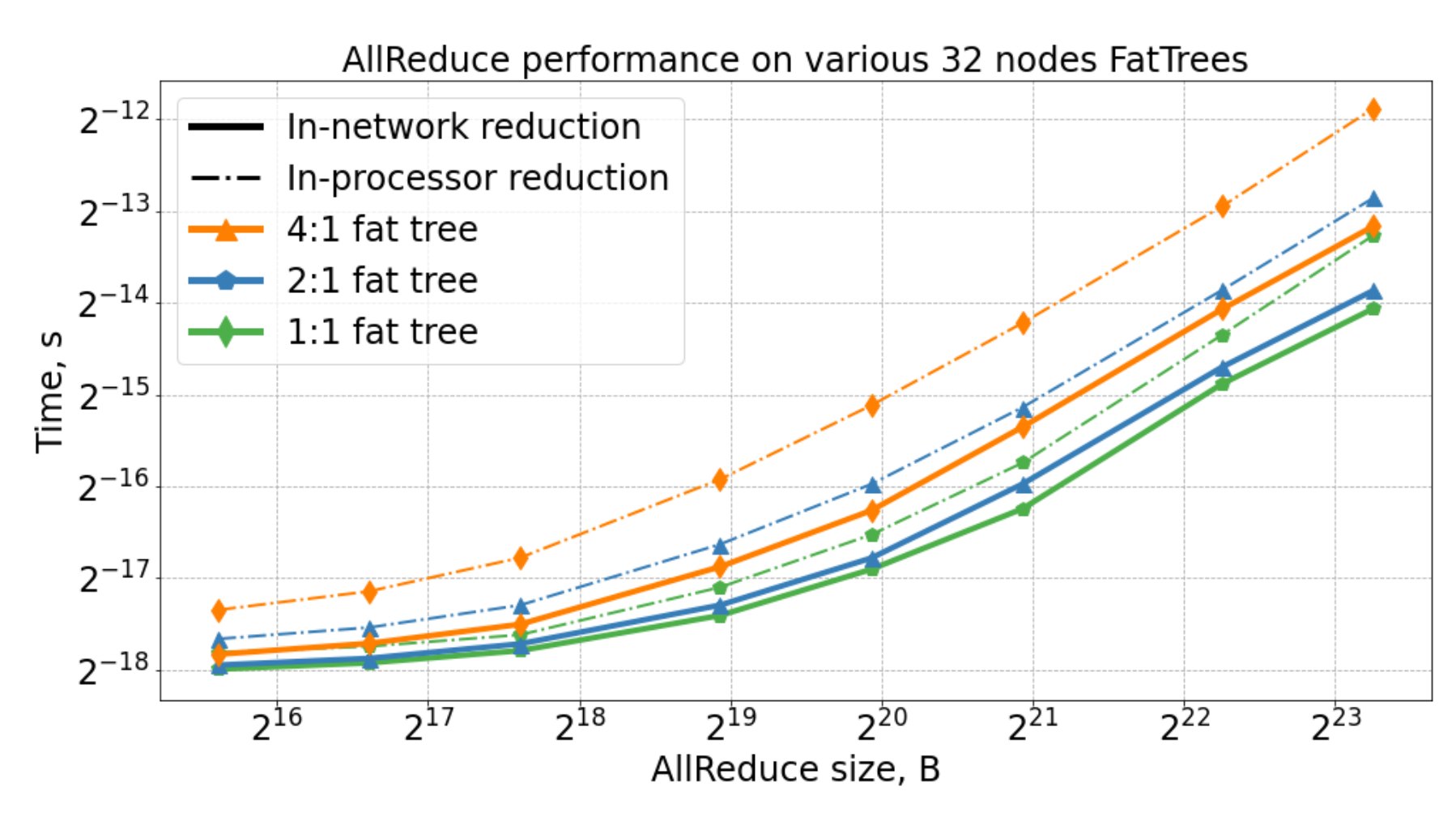


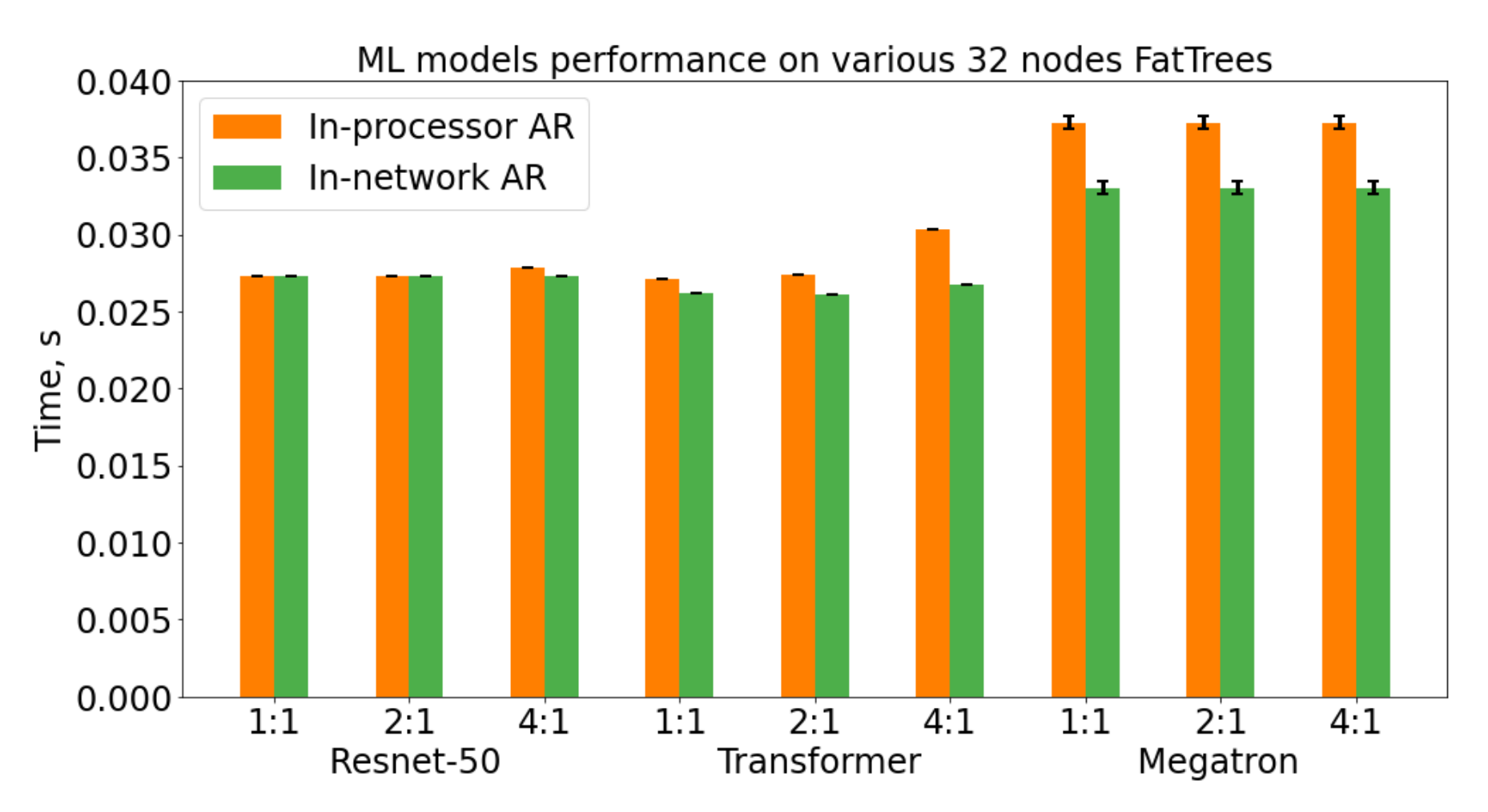


Case Study 3: in-network reduction analysis

ParaGraph allows HW engineers test future designs, such as network switch with in-network reductions, not only on generated traffic, but also on real applications

ParaGraph can help uncover performance bottleneck of specific applications before the chip designs are fabricated and systems are deployed





Conclusion

ParaGraph is a versatile co-design tool that allows mixing and matching components to construct various end-to-end simulation workflows

ParaGraph supports multiple network simulators and several front-end libraries via XLA compiler. In future we plan to provide even more frontends and backends

Find us on https://github.com/paragraph-sim

Acknowledgement

Portions of this work were performed during internships at Google and Nvidia.