Data-Affinity Acceleration Design for Large-Scale Network Simulation
面向数据设计的大规模网络仿真

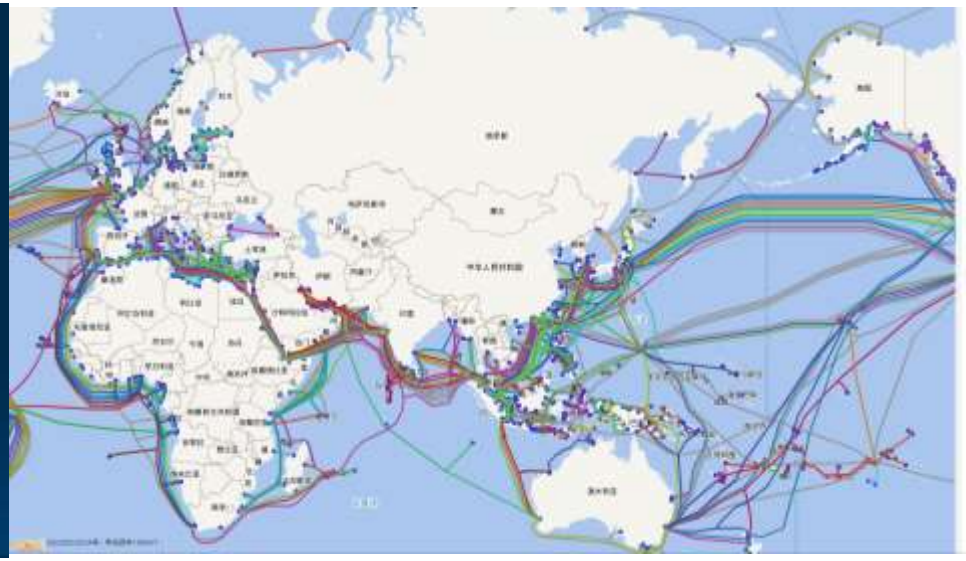
高凯辉

中关村实验室

目录

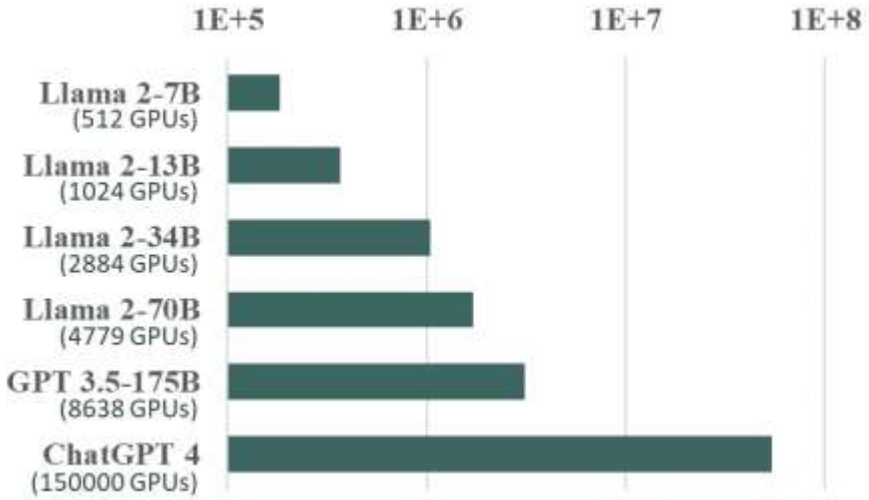
- 离散事件仿真(Simulation)技术概述
- 面向数据设计的网络**数据面仿真**研究
- 面向数据设计的网络**控制面仿真**研究
- 面向数据设计的**AI智算网络仿真**研究

网络规模持续增长



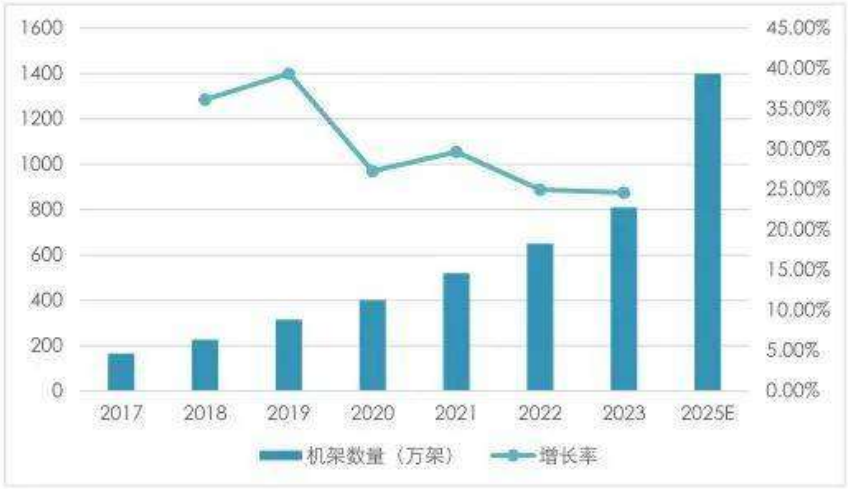
运营商骨干网普遍包括：**一万多台**核心路由器、**三万多条**链路

全球互联网AS规模已超过**10万**



智算中心内：训练LLM模型所需的A100 GPU小时数

图表 1：近年我国数据中心机架规模及增速



数据来源：中国通服数字基建产业研究院

为了支撑LLM的算力需求，**智算中心网络的规模也在迅速增长**！

以AI智算网络为例

- AI智算网络架构的设计空间非常复杂，其他网络类似

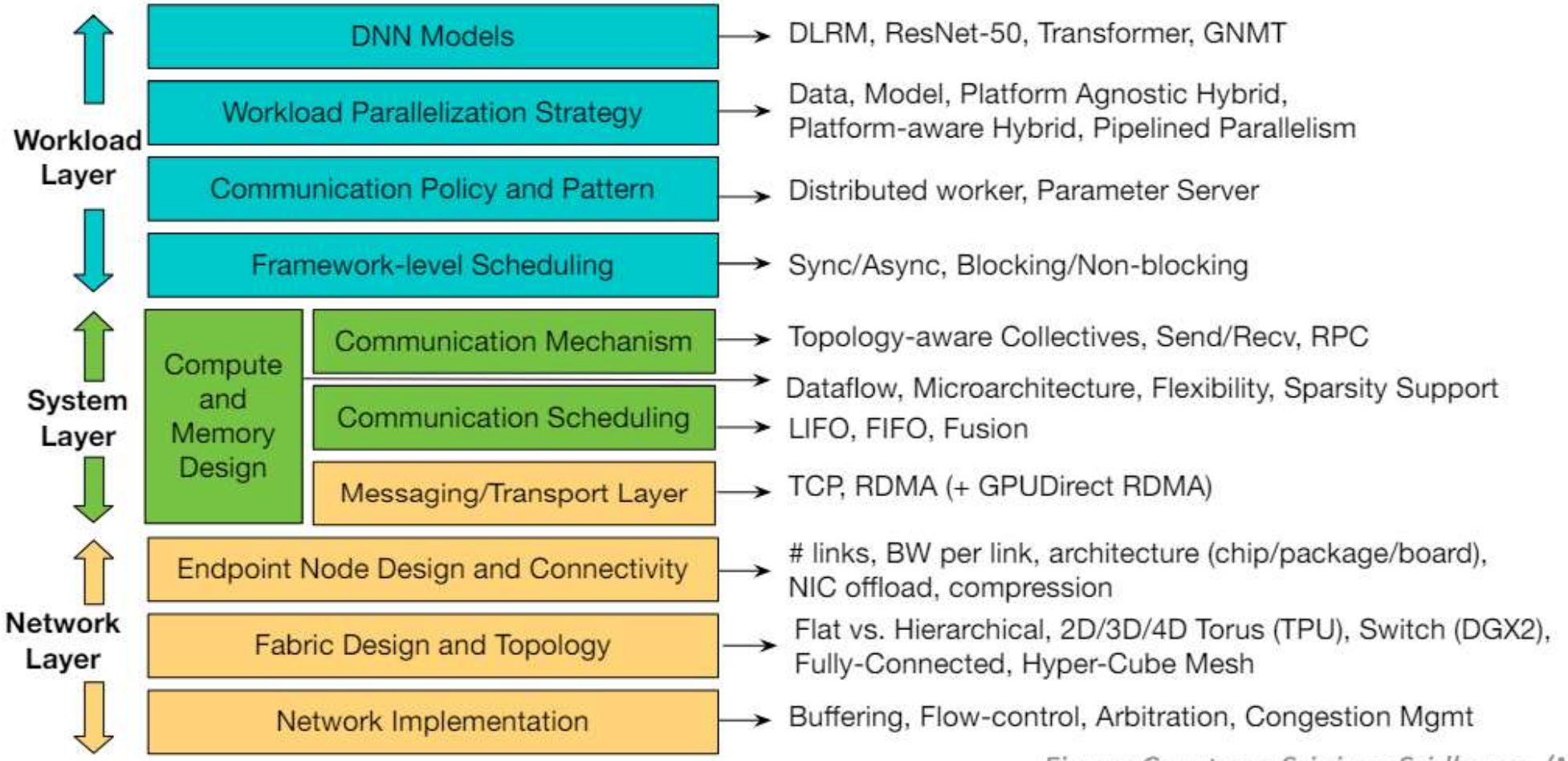


Figure Courtesy: Srinivas Sridharan (Meta)

网络设计/运维需要仿真技术（Simulation）

- 基于**物理试验床**的实验成本高、耗时长
- 由于网络拥塞等细粒度事件，**基于AI的网络性能近似模型**不准确
- **仿真技术Simulation**

建模已存在/不存在的复杂系统

任意粒度（包级别、流级别）

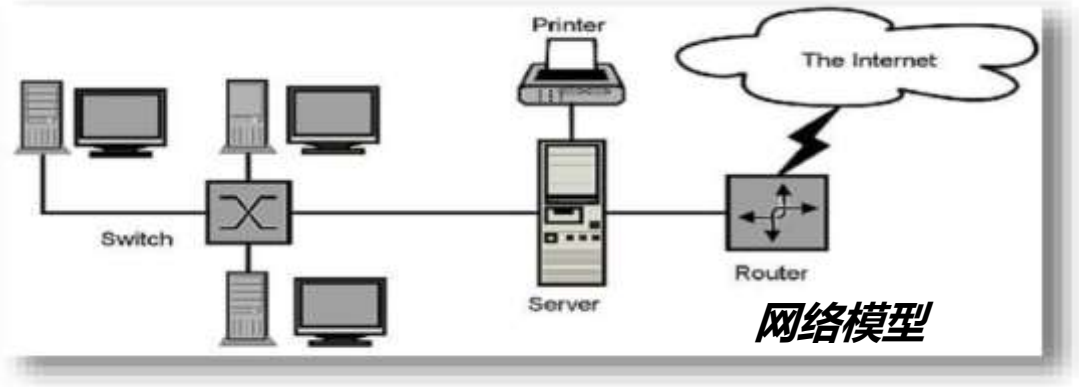
任意规模

任意扩展

任意观测

仿真介绍

- “Simulation is the process of designing a model of a real system and conducting experiments with this model for the purpose of either understanding the behavior of the system and/or evaluating various strategies for the operation of the system.”

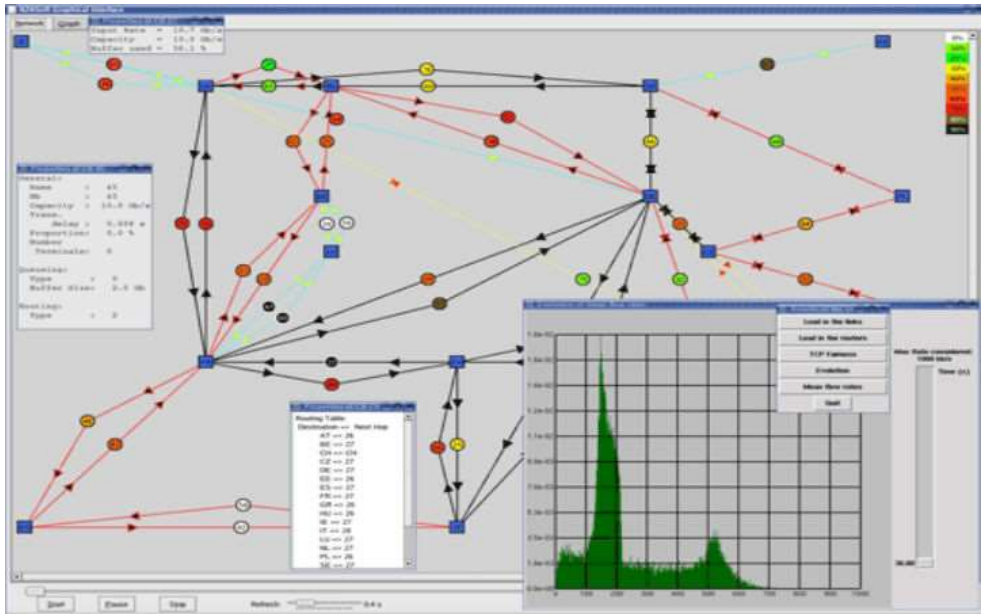


- 用处
  - ◆对复杂系统进行详细建模
  - ◆为观察到的行为建立理论或假设
  - ◆使用模型来预测未来的行为
  - ◆验证所提出的新系统、新方法

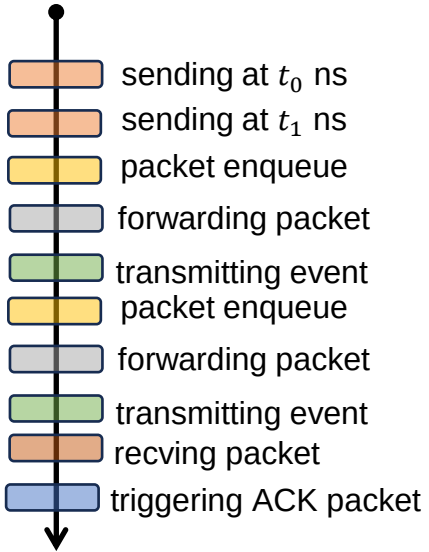
离散事件仿真介绍

- A **discrete-event simulation (DES)** models the operation of a system as a (discrete) sequence of events in time. Each event occurs at a particular instant in time and marks a change of state in the system. Between consecutive events, no change in the system is assumed to occur; thus the simulation time can directly jump to the occurrence time of the next event, which is called **next-event time progression**. —— From ***Wikipedia***

- 事件驱动
  - 每个事件都提供对下一个事件的引用(例如:使用指针)
- 仿真结束
  - 当没有更多事件时，或在指定的时间



离散事件仿真执行过程

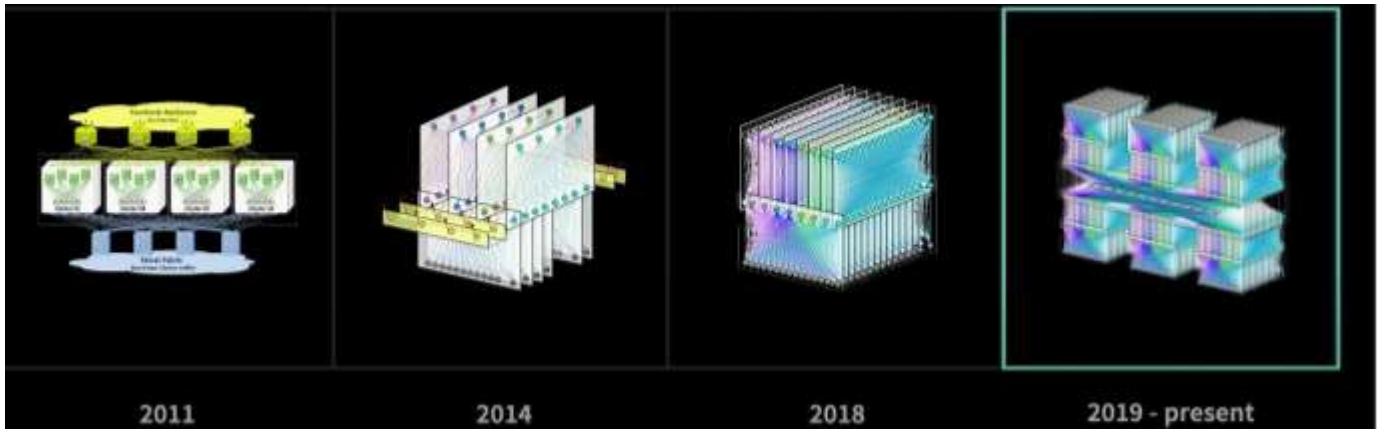


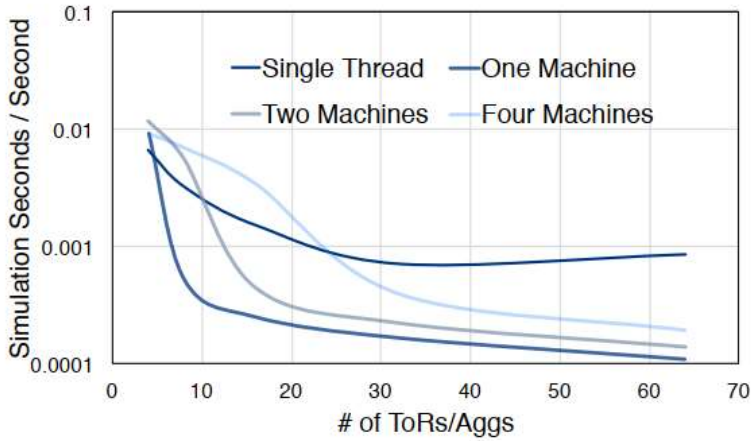
以NS-3为代表

"ns" family of simulators awarded the 2020 ACM SIGCOMM Networking Systems Award

已有DES仿真器性能较差

- 现代网络的规模使得NS-3等工具的性能问题愈发严重





OMNeT++（串行、并行）在各DCN拓扑上的性能

- 需要满足严格正确性保证，进程间同步开销使得DES并行效率极低
  - OMNeT++ 需要**至少九天的时间**来仿真常规DCN的1秒
  - NS-3 具有相似的性能

目录

- 离散事件仿真(Simulation)技术概述
- **面向数据设计的网络数据面仿真研究**
- 面向数据设计的网络控制面仿真研究
- 面向数据设计的AI集群网络仿真研究

DONS [SIGCOMM 23]

Fast and Affordable Discrete Event Network Simulation with Automatic Parallelization

快速且低成本的网络离散事件仿真系统

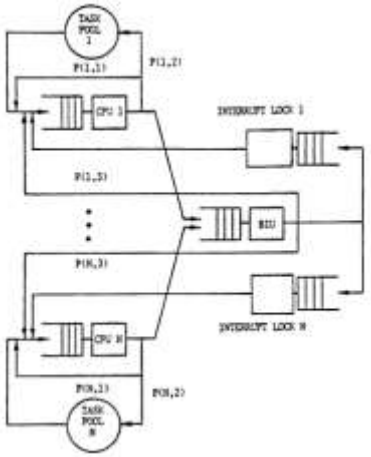
Kaihui Gao

Li Chen, Dan Li, Vincent Liu, Xizheng Wang, Ran Zhang, Lu Lu

相关技术

1. 连续时间仿真（Continuous-time simulation）

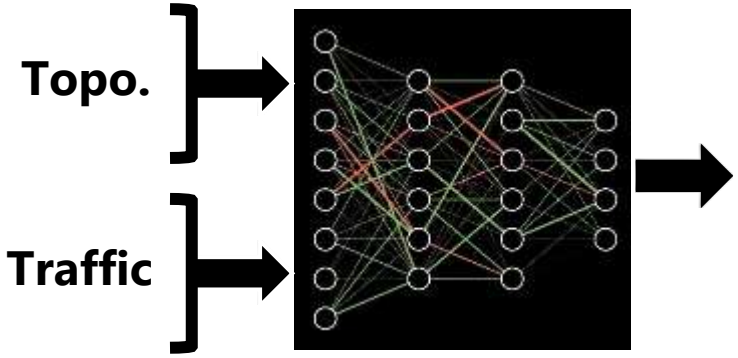
◆ 排队论，网络演算，控制论等技术



- 可扩展性较好
- 忽略包级别事件
- 低保真度

2.基于深度学习的性能估计（AI-powered performance approximation ）

◆ RouteNet[SOSR'19], MimicNet[SIGCOMM'21], DeepQueueNet[SIGCOMM'22], 等

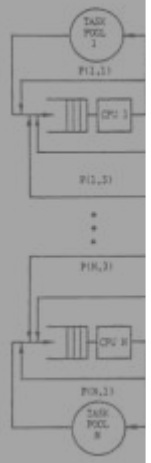


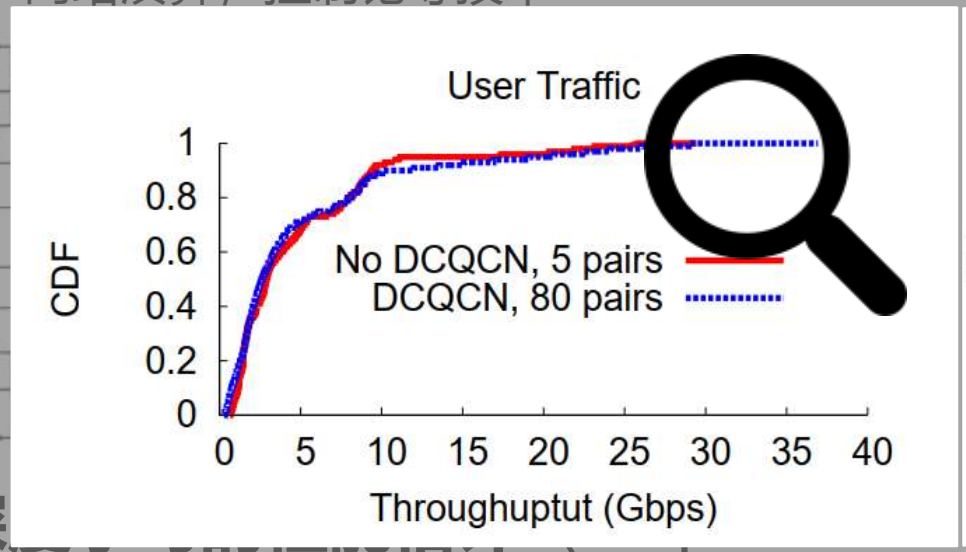
- 快速
- 需要GPU加速
- 固有的误差
- 扩展性仍然被DES限制

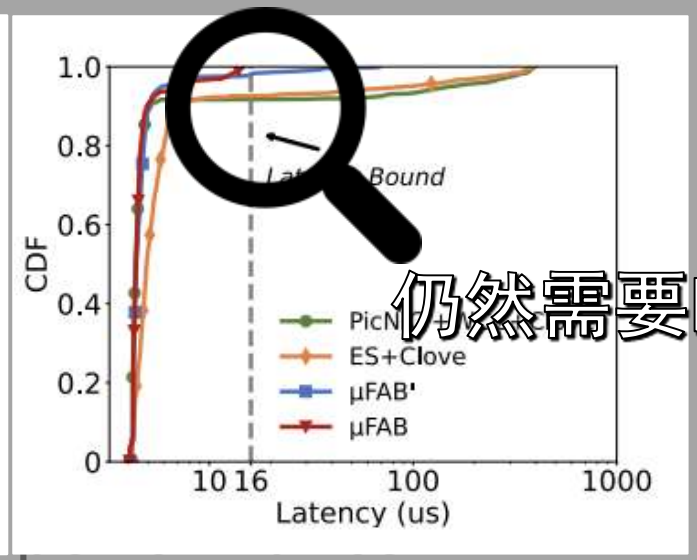
相关技术

1. 连续时间仿真（Continuous-time simulation）

- 排队论，网络演算，控制论等技术







仍然需要DES

2.基于深[illegible]

approximation)

- Route[illegible]

问题不在于DES仿真的计算负载，
而是计算资源未被充分使用

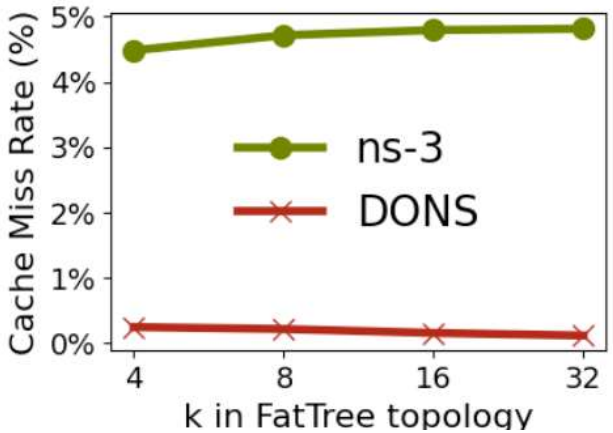
Topo.

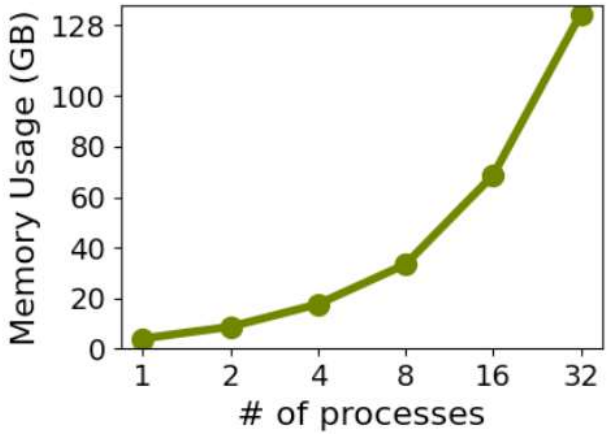
Traffic

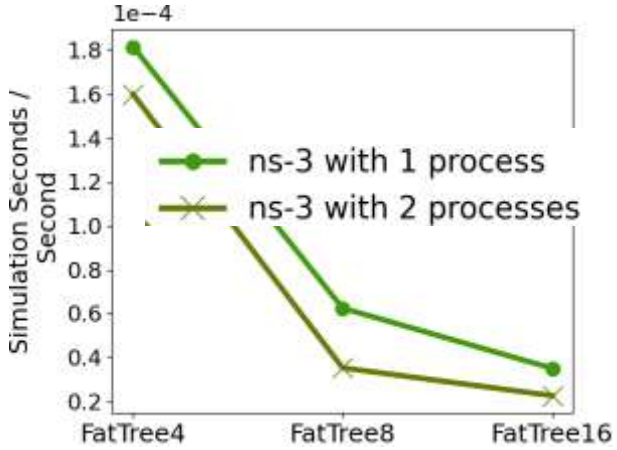
Metrics
(e.g., delay)

- [illegible]
- 固有的误差
- 扩展性仍然被DES限制

已有DES仿真器未充分利用多核CPU





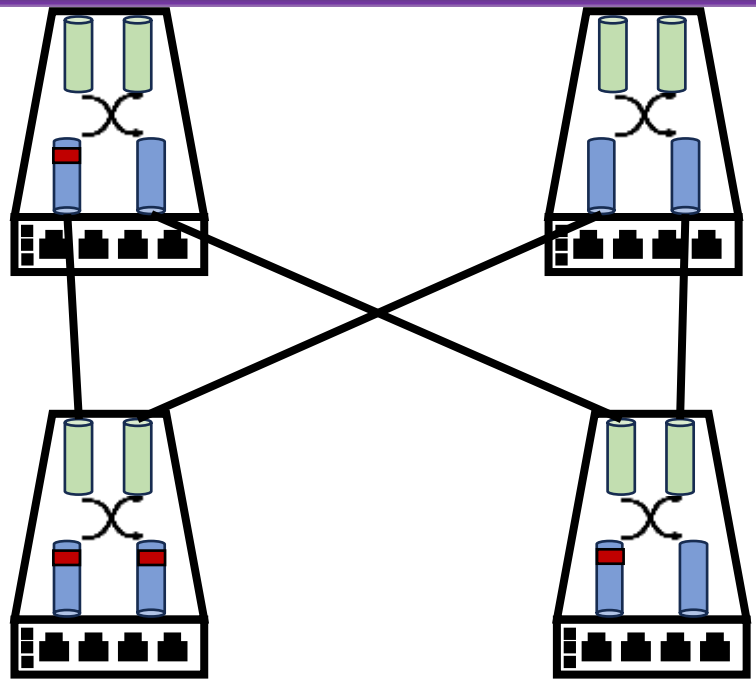


1. **High CPU cache miss rate:**
   ~5% 的L3 cache miss rate

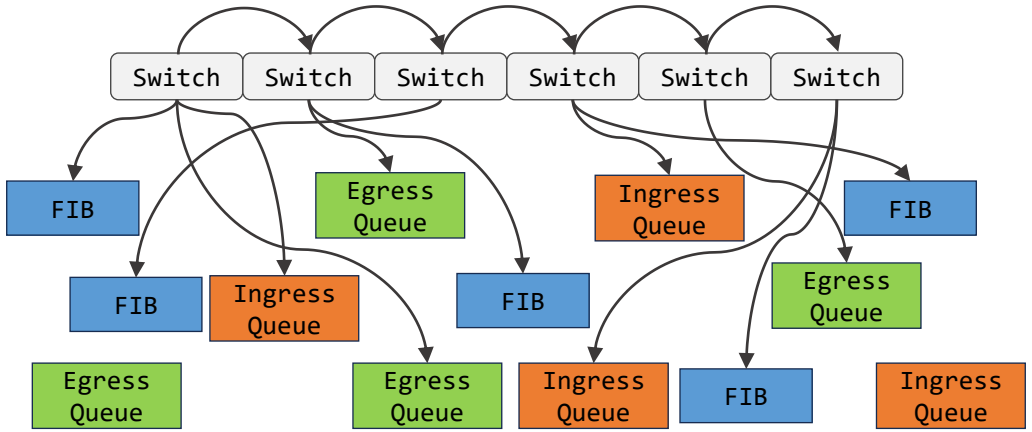
2. Poor memory efficiency:
使用32进程仿真Fattree (k=32)
需要**~5TB**

3. Low parallelization efficiency:
双进程并行比单进程**慢**
~43.5%

根本原因：面向对象设计的软件架构



```
// Object-oriented Design
class Switch {
    IngressQueue[] inqueues;
    EgressQueue[] outqueues;
    FIB fib_table;
};
void Switch::Forwarding(Ptr<Packet> p)
{
    int outport = fib_table.lookup(p.dst_ip);
    outqueues[outport].enqueue(p);
}
```



内存中的数据布局

- 主流编程范式
- 数据与逻辑组合成对象
- 数据散列在内存中

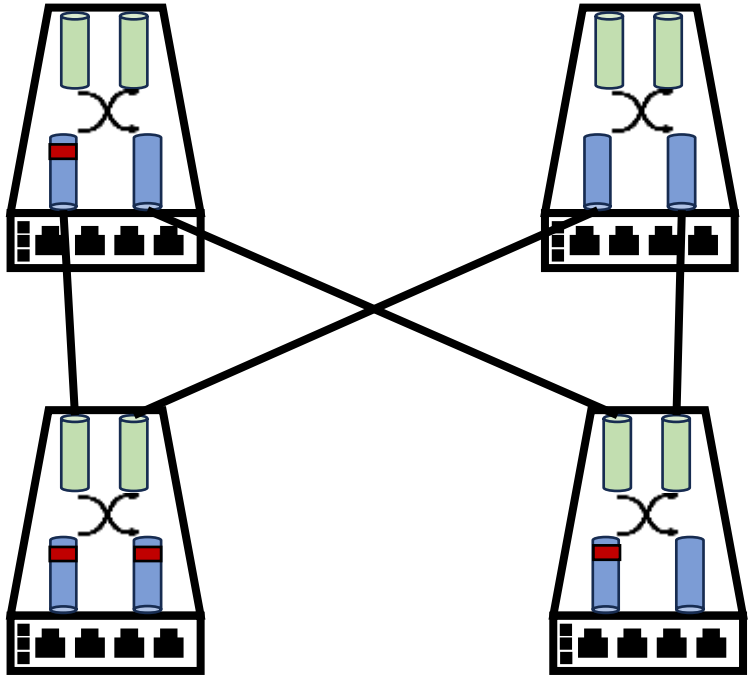
➢ **对CPU cache不友好**
➢ **难以自动并行**

提升OOD：DOD和ECS

- 视频游戏领域中的解决方案：**面向数据设计 Data-oriented Design (DOD)**
- DOD思想的实现：Entity Component System (ECS)架构

Madrona

- 使用DOD/ECS建模网络转发行为



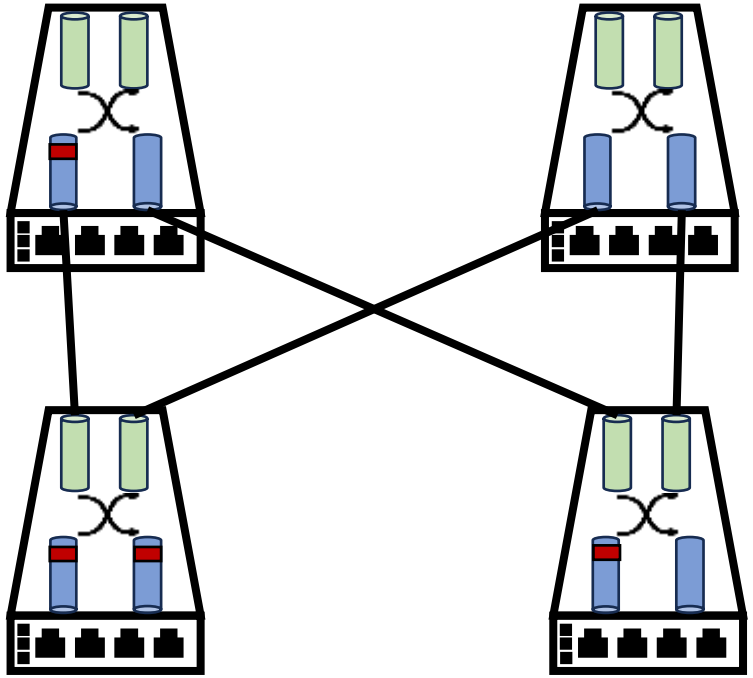

```
// Data-oriented Design
struct IngressQueue : Component { ... }
struct Ptr<EgressQueue> : Component { ... }
struct FIB : Component { ... }
```

提升OOD：DOD和ECS

- 视频游戏领域中的解决方案：**面向数据设计 Data-oriented Design (DOD)**
- DOD思想的实现：Entity Component System (ECS)架构

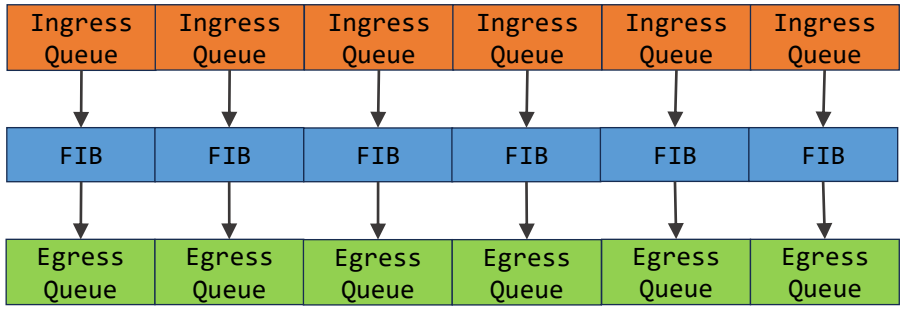
Madrona

- 使用DOD/ECS建模网络转发行为




```
// Data-oriented Design
struct IngressQueue : Component { ... }
struct Ptr<EgressQueue> : Component { ... }
struct FIB : Component { ... }
void Forwarding_System(IngressQueue inq, FIB
fib_table, Ptr<EgressQueue> outqueues) {
  foreach (var p in inq) {
    int index = fib_table.lookup(p.dst_ip);
    outqueues[index].enqueue(p);
  }
}
```

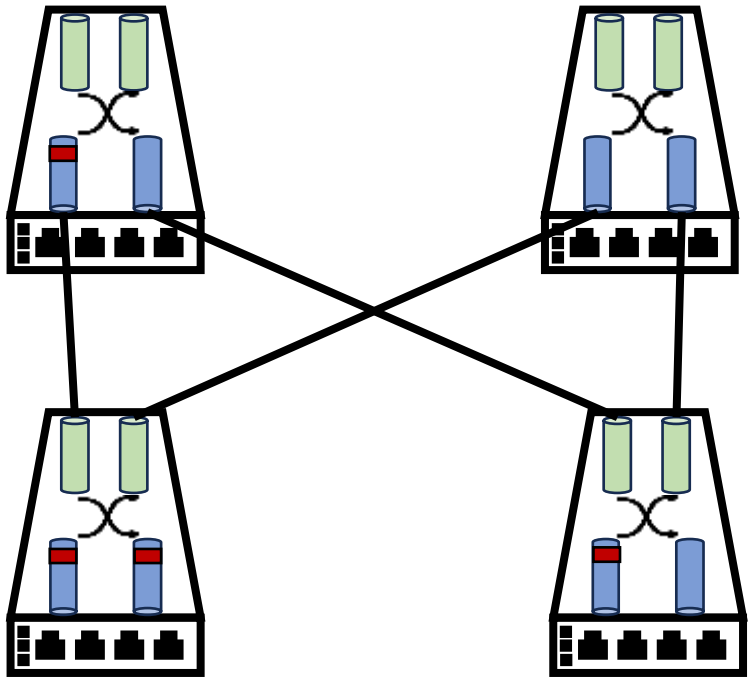
提升OOD：DOD和ECS



内存中的数据布局

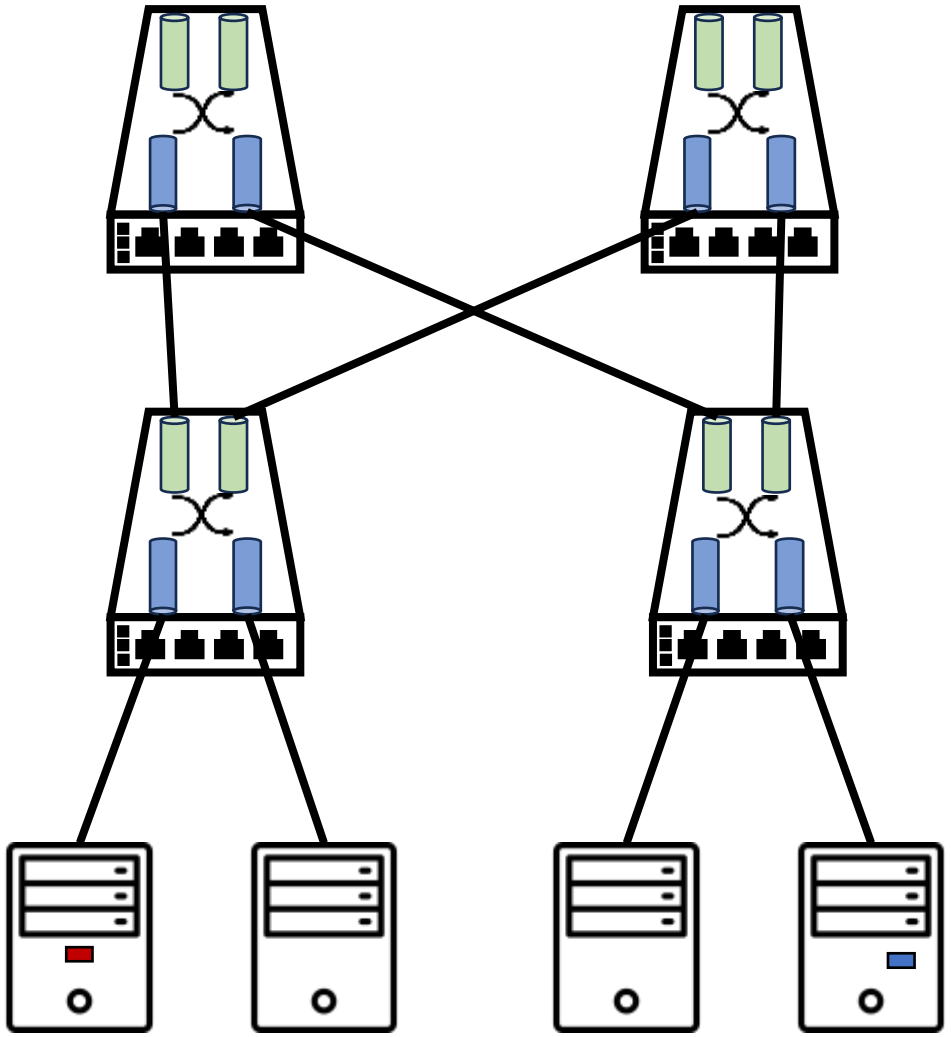
- 相同类型的数据存储在相邻的内存中
- 数据与逻辑完全解耦

➢ **对CPU cache非常友好**
➢ **容易自动并行**




```
// Data-oriented Design
struct IngressQueue : Component { ... }
struct Ptr<EgressQueue> : Component { ... }
struct FIB : Component { ... }
void Forwarding_System(IngressQueue inq, FIB
fib_table, Ptr<EgressQueue> outqueues) {
  foreach (var p in inq) {
    int index = fib_table.lookup(p.dst_ip);
    outqueues[index].enqueue(p);
  }
}
```

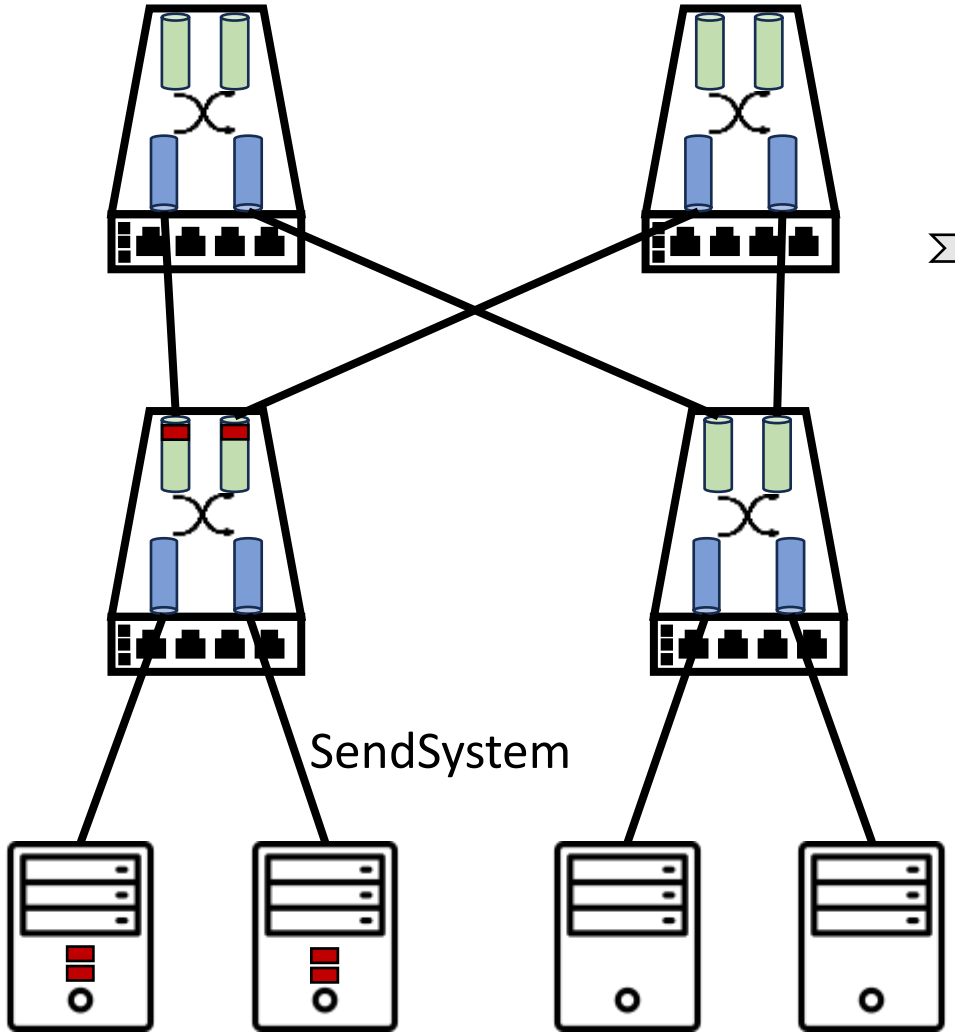
DONS：网络建模



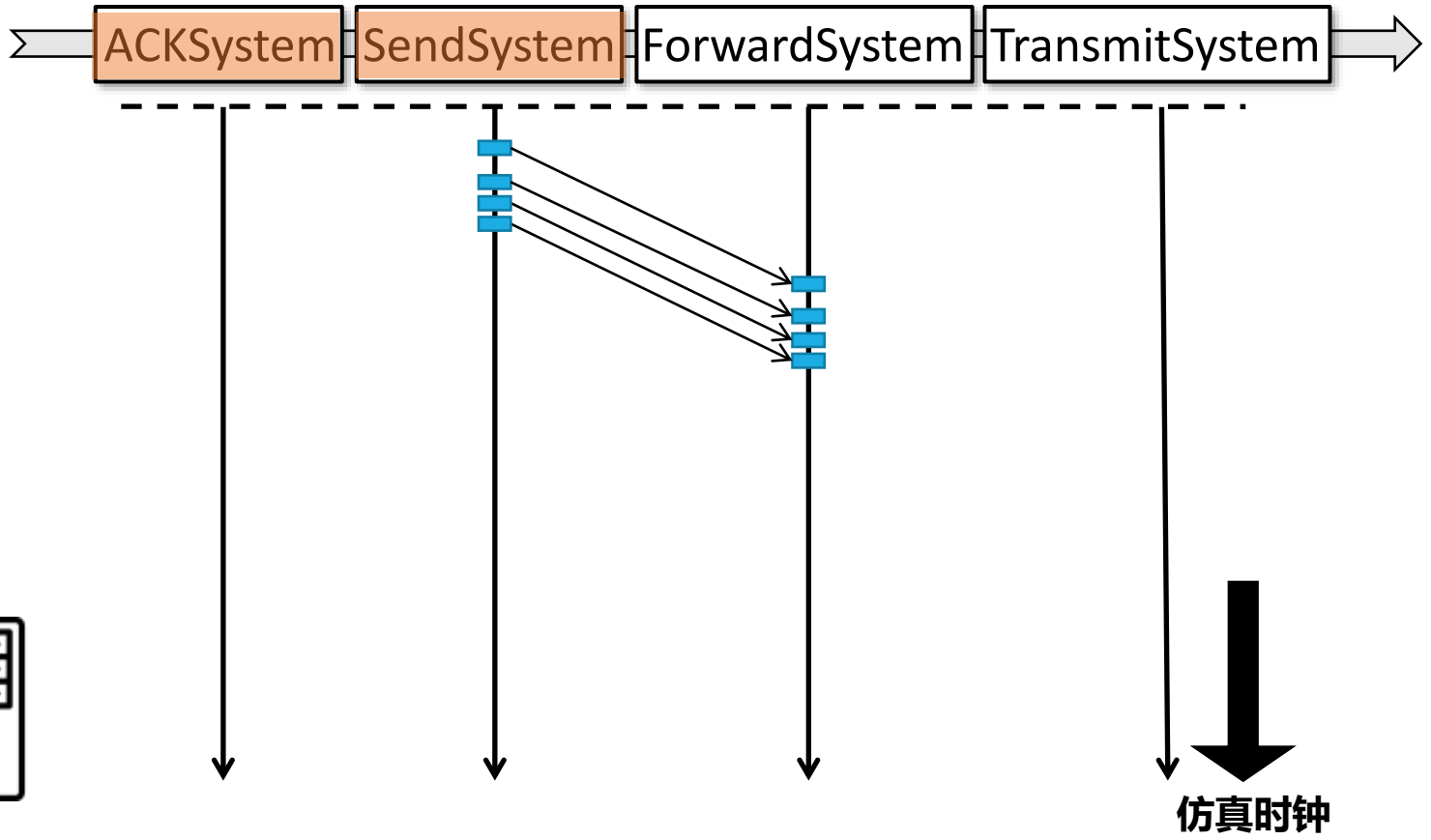
4 Entities	Components
Sender	IP, Flows, DCTCP, Stats
Receiver	IP, Buffer, RWND, Stats
IngressPort	MAC, Buffer, FIB, Stats
EgressPort	MAC, Buffer, RR, Stats

4 Systems	
SendSystem	generating *packets* in **Sender** and moving them to the connected **IngressPort**
ForwardSystem	forwarding the *packets* in the **IngressPort** to the corresponding **EgressPort**
TransmitSystem	transmiting the *packets* in **EgressPort** to the corresponding **IngressPort** or **Receiver**
ACKSystem	processing the *packets* received by the **Receiver** and triggering ACKs

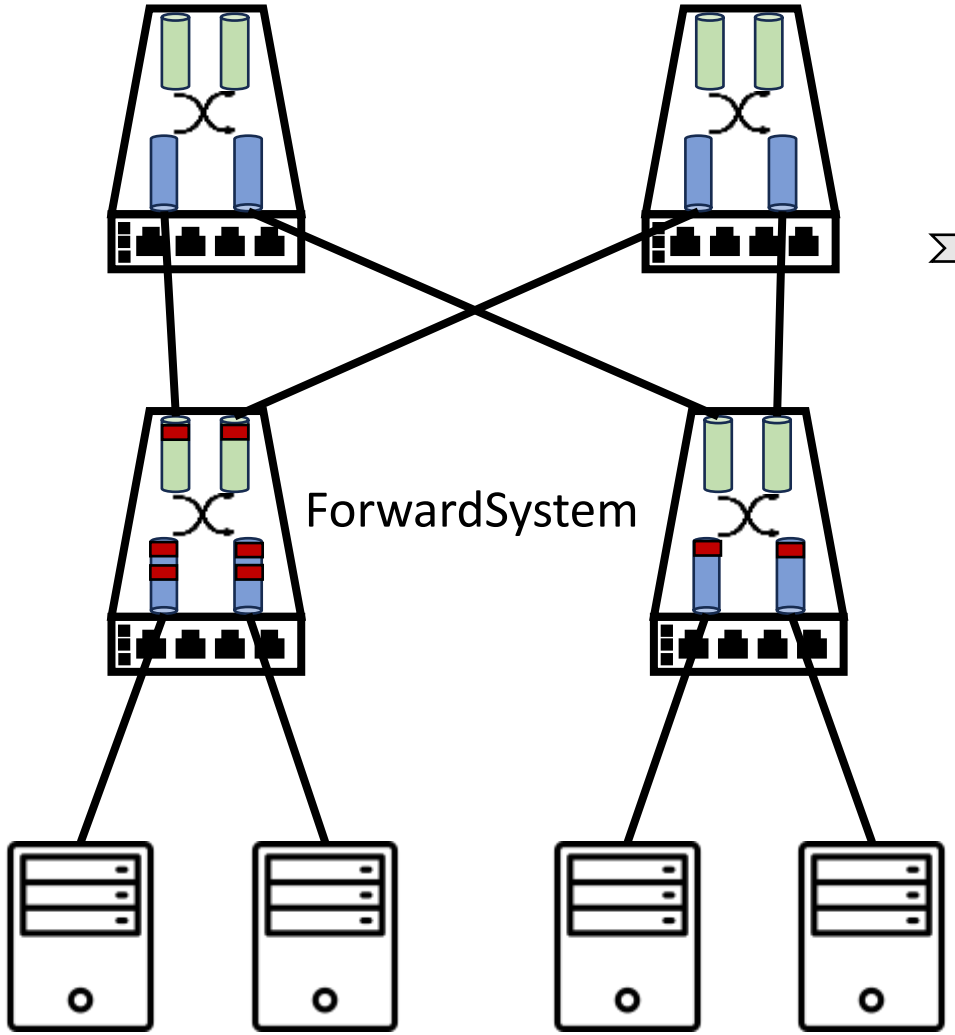
DONS：线程模型



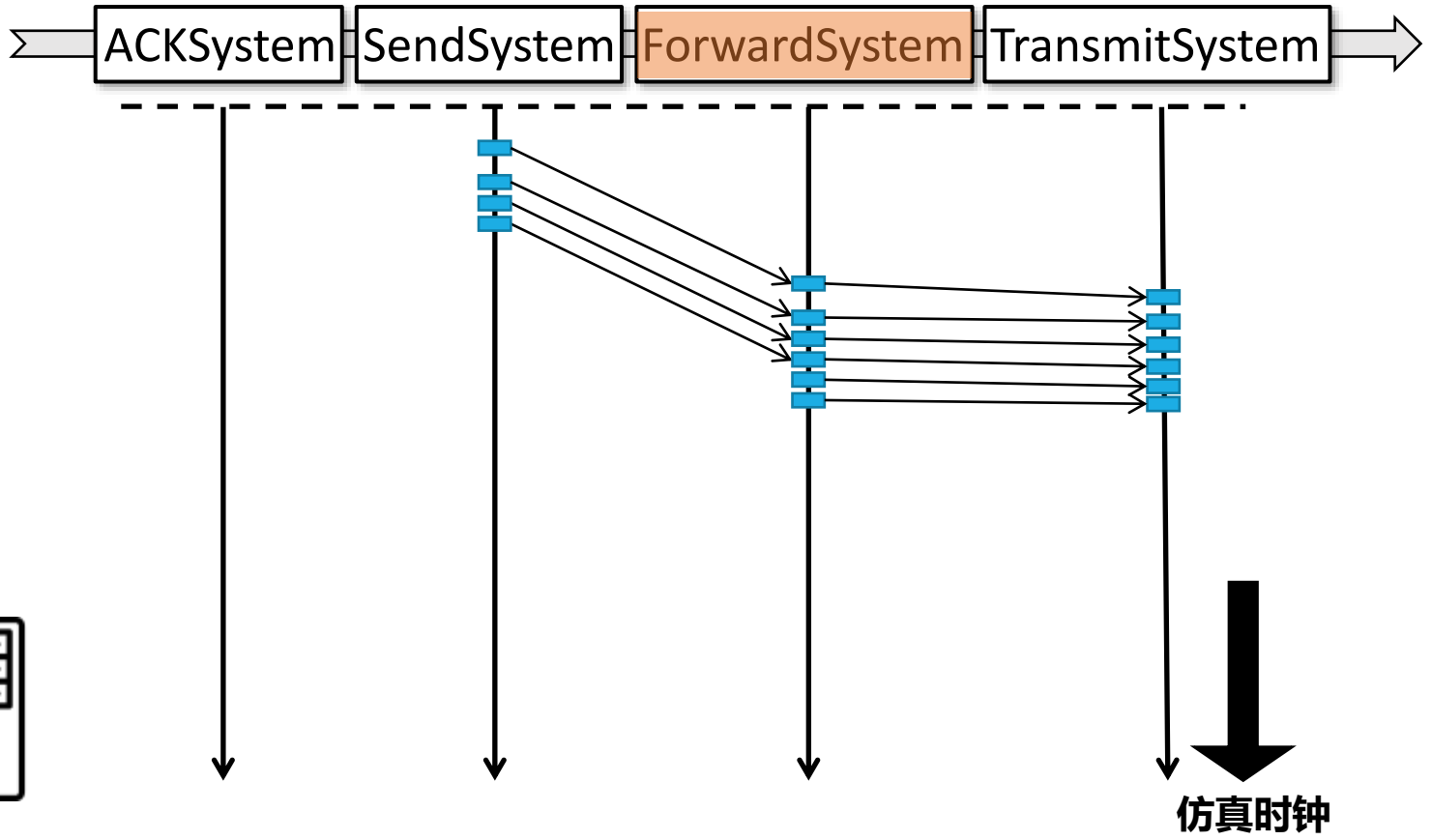
- 1. 顺序执行四个systems
- 2. 所有设备的同一system在多核CPU上并行执行



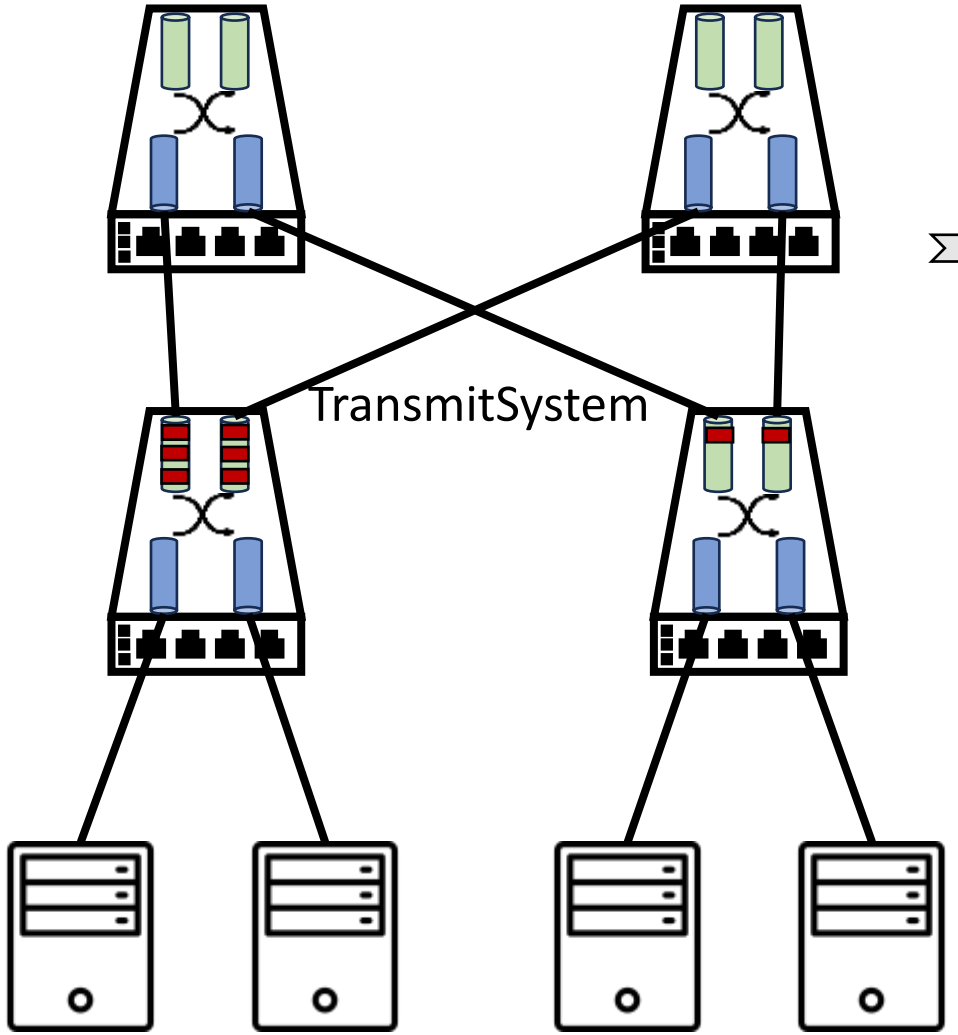
DONS：线程模型



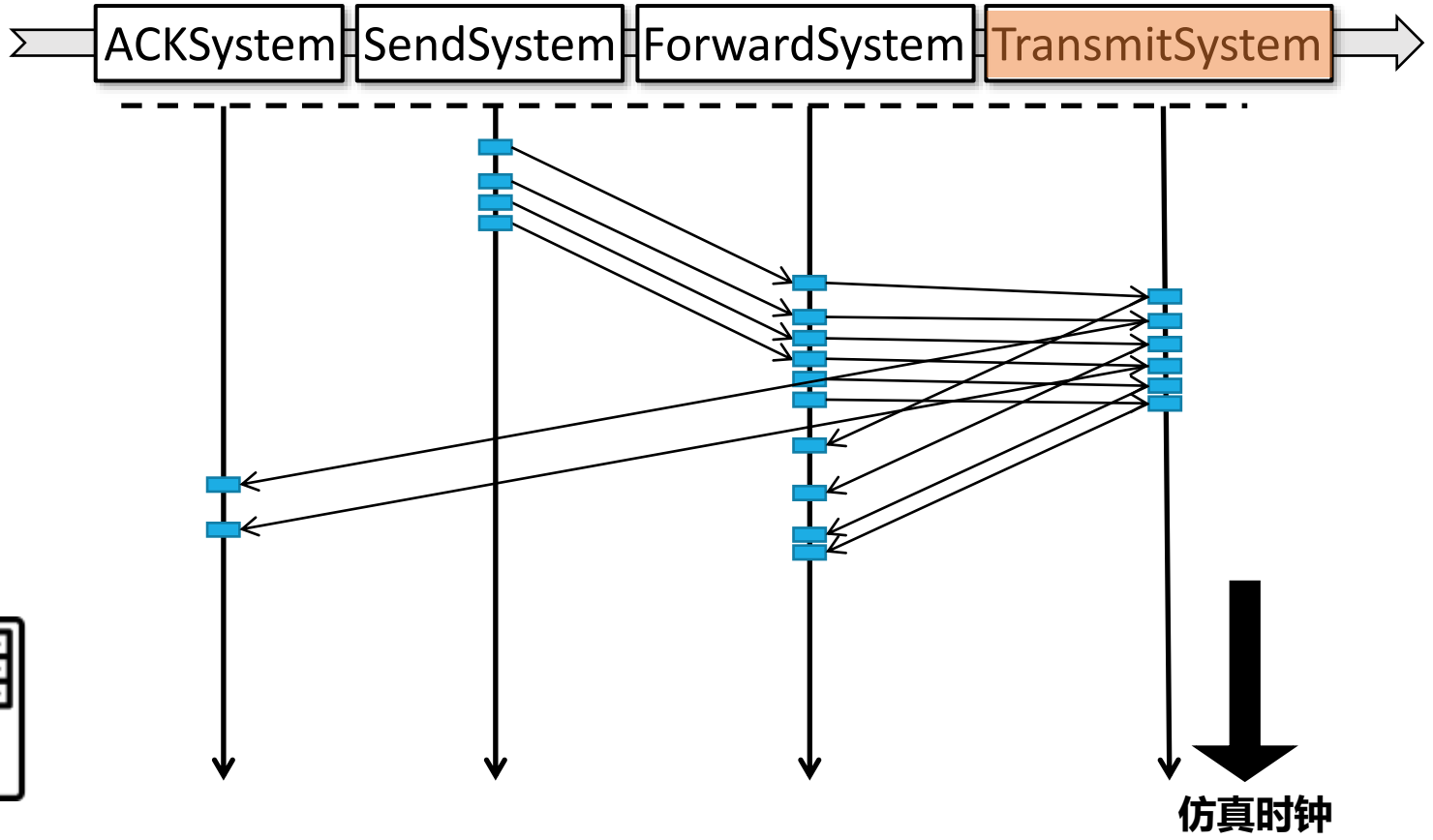
- 1. 顺序执行四个systems
- 2. 所有设备的同一system在多核CPU上并行执行



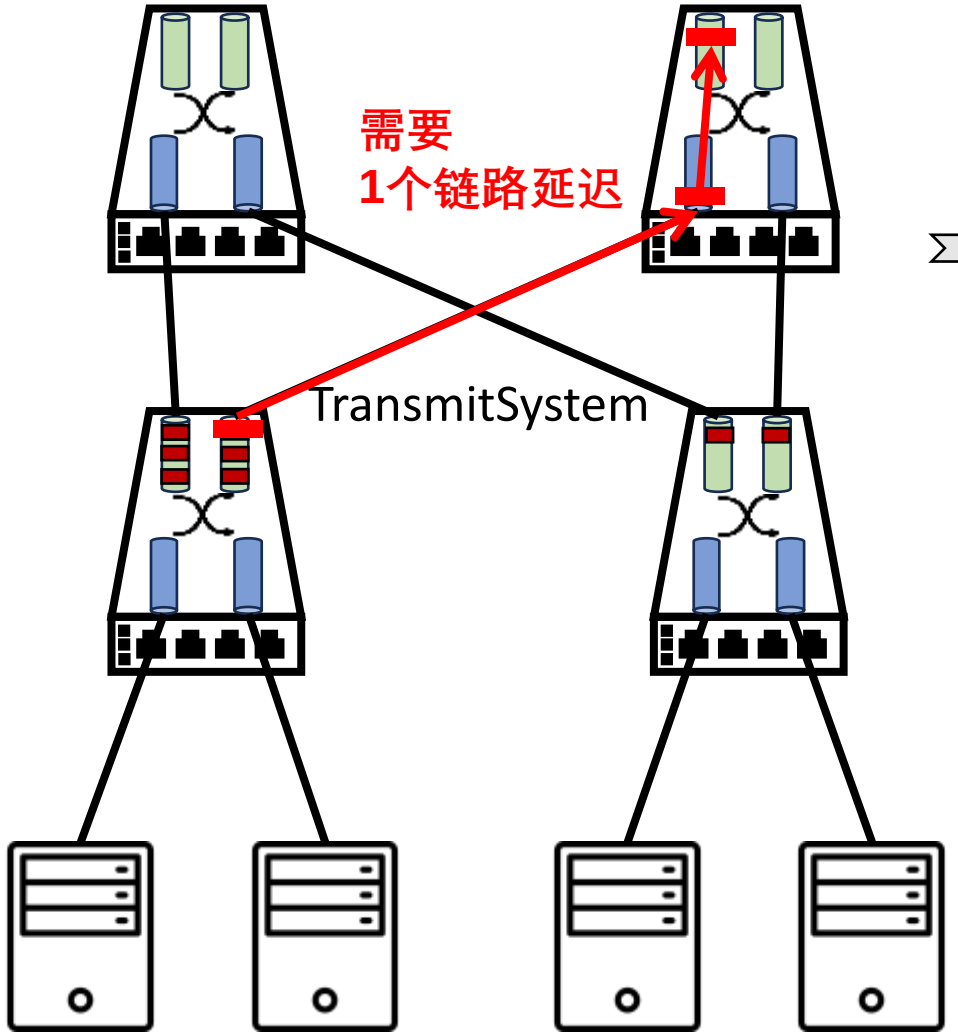
DONS：线程模型



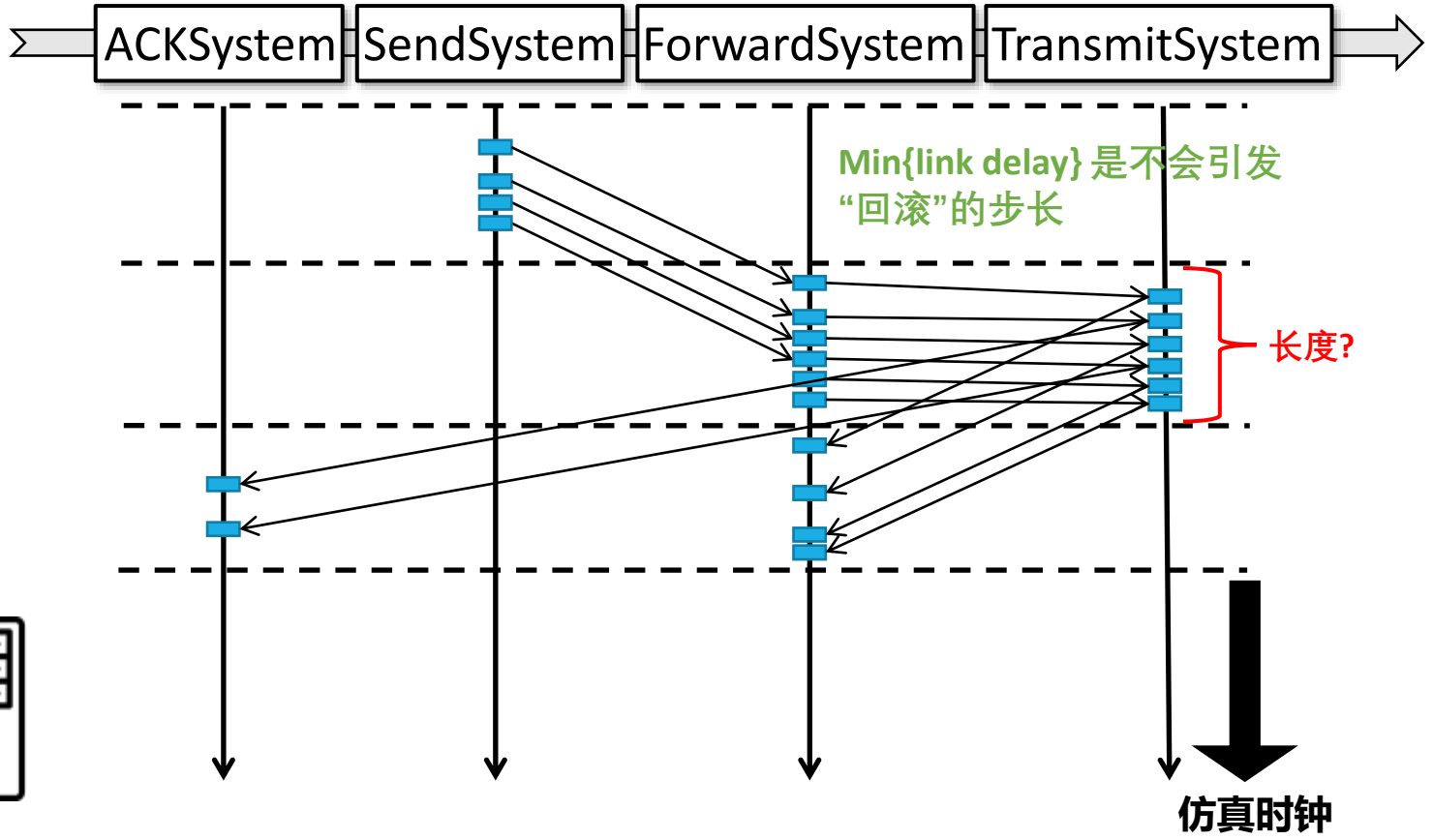
- 1. 顺序执行四个systems
- 2. 所有设备的同一system在多核CPU上并行执行



DONS：线程模型

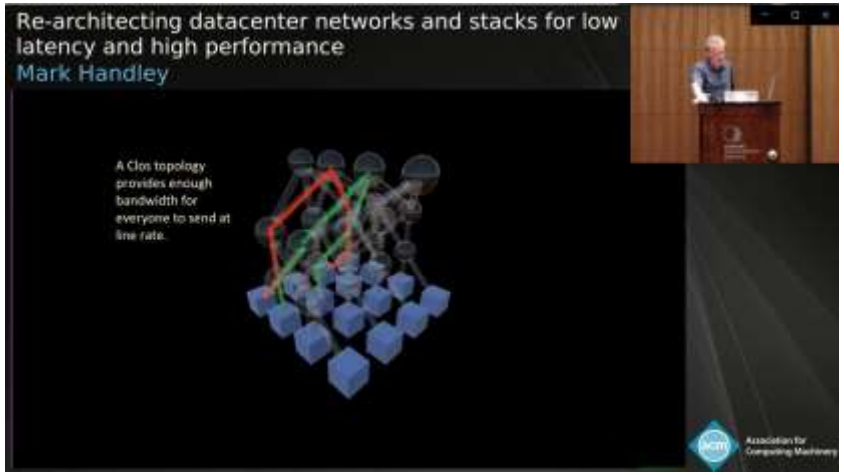


- 1. 顺序执行四个systems
- 2. 所有设备的同一system在多核CPU上并行执行

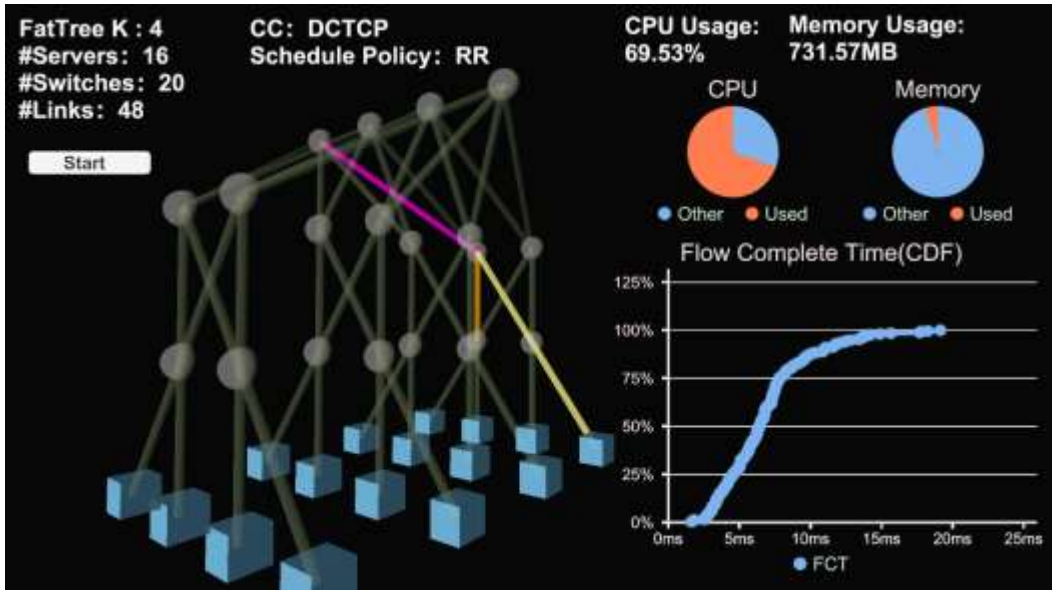


实现

- 受Mark Handley在SIGCOMM'17和SIGCOMM'19上的启发，我们基于Unity实现了DONS
- ~1万行代码，已开源[1]
- **发布DONS v1.0**，支持网络数据平面性能仿真：
  - UDP, TCP, DCTCP
  - IPv4, ECMP, RED
  - FIFO, RR, DRR, SP
- 优化技术包括：
  - Command buffer, Merge Sort, …



Mark Handley'在SIGCOMM'17上的演示



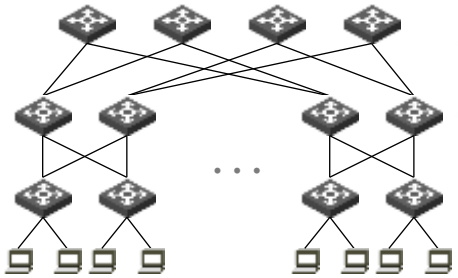
DONS前端

[1] https://github.com/dons2023/Data-Oriented-Network-Simulator

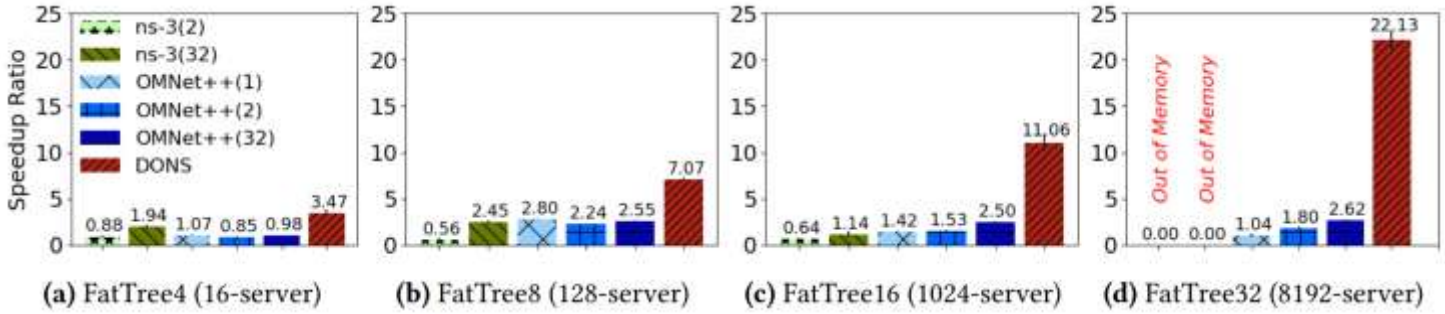
23

实验效果

仿真速度

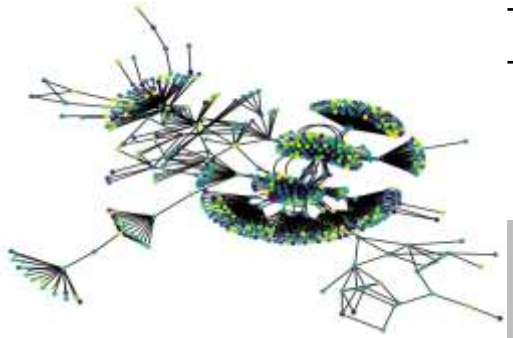


FatTree (K = 4, 8, 16, 32)



DONS相比于ns-3（单进程）加速了3倍 到 22倍

可扩展性



Large-scale WAN (13k routers) from a ISP

FatTree k = 64 (65k servers, 5k sw)

#Machines	Simulator	#GPUs	Time	Speedup	w_1
4	OMNeT++	0	9d 14h 24m	baseline	-
	DeepQueueNet	4	2h 56m	78.5X	0.43
	DONS	0	5h 27m	42.2X	0
8	OMNeT++	0	7d 19h 8m	baseline	-
	DeepQueueNet	8	1h 48m	104.1	0.46
	DONS	0	2h 53m	65.0X	0

9.5 天 → 5 小时

~8 天 → 3 小时

DONS相比于OMNeT++加速了42倍 到 65倍

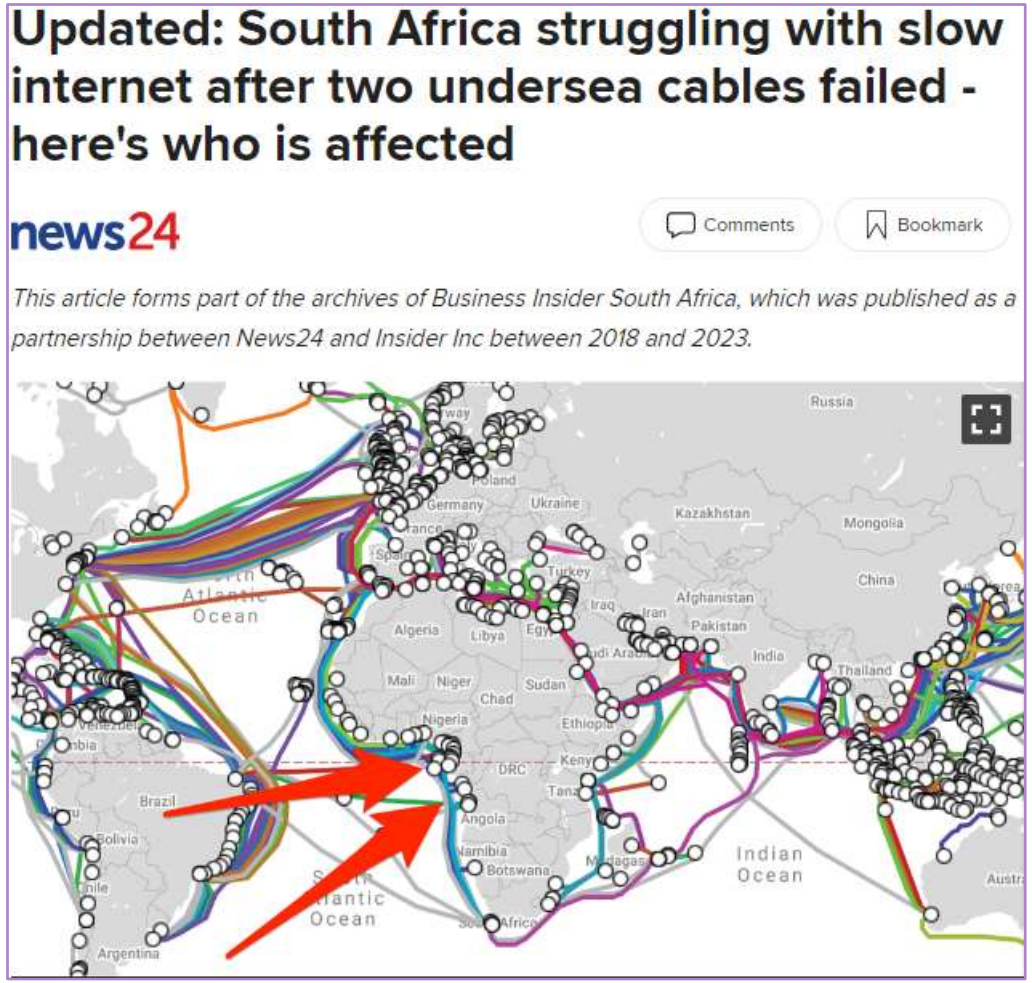
24

目录

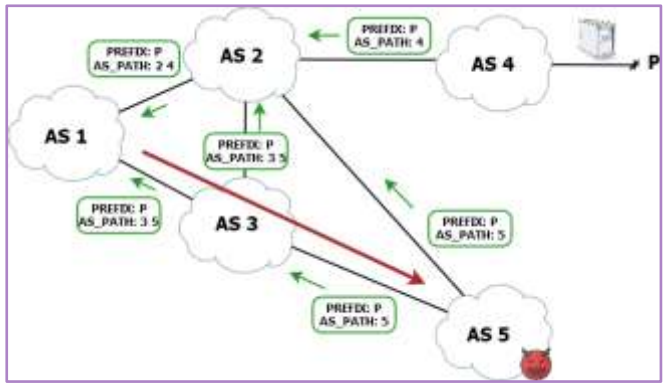
- 离散事件仿真(Simulation)技术概述
- 面向数据设计的网络数据面仿真研究
- 面向数据设计的网络**控制面仿真**研究
- 面向数据设计的AI集群网络仿真研究

网络控制平面仿真至关重要

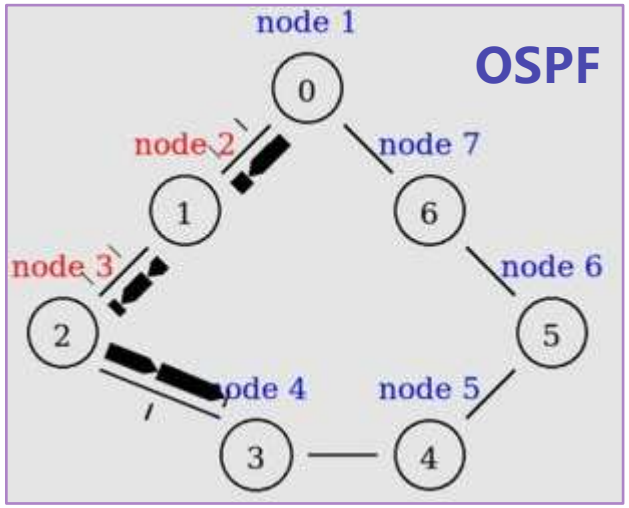
- 仿真故障对路由的影响

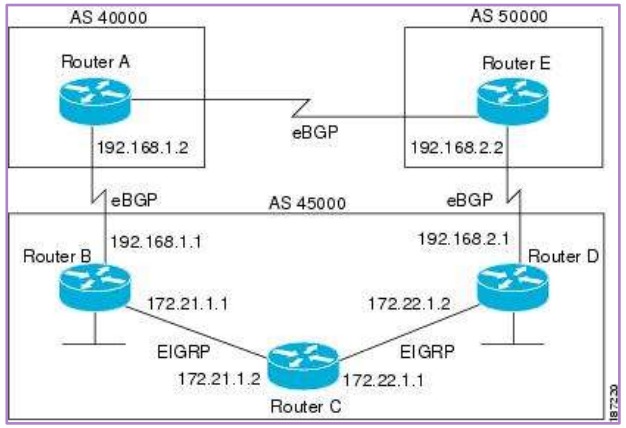


- 复现/预判路由安全事件（路由劫持/泄露）



- 验证控制平面相关配置





网络控制平面仿真技术分类

□网络仿真（Network Simulation）

◆如NS-2/3，OMNeT++，OPNET，QualNet，EXata，NetSim

◆协议运行时的资源消耗少（**轻量**）

◆ 默认为单进程单线程模式，**速度极慢**

◆ 多进程的并行效率很低，**可扩展性较差**

□网络模拟（Network Emulation）

◆Mininet, CORE, GNS-3, CrystalNet *[SOSP'17]*, mini-Internet *[SIGCOMM CCR'20]*, SEED *[HotNets'22]*

◆协议实现的**保真度较高**

◆ 协议运行时的资源消耗多（VM/docker)**可扩展性较差**

◆ 对网络性能（控制面处理能力、带宽）仿真**不准确**

□控制平面形式化验证（Control Plane Verification）

◆Batfish, Hoyan*[SIGCOMM'20]*, SRE *[SIGCOMM'22]*

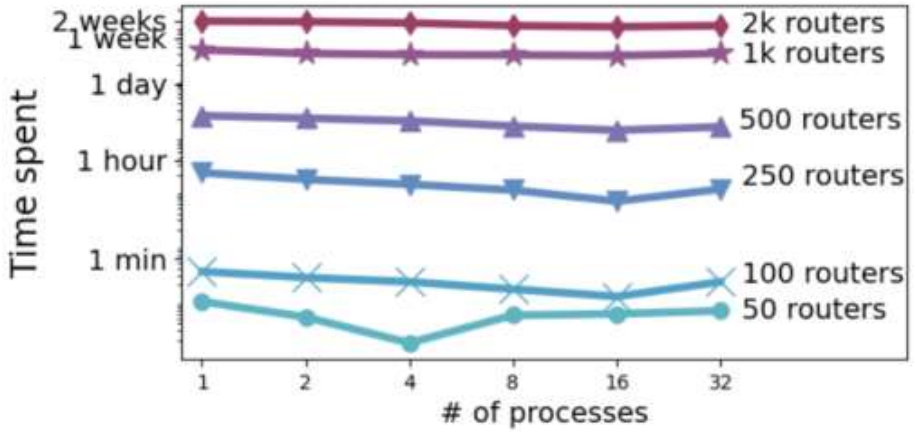
◆有数学**理论基础**，分析出所有故障case

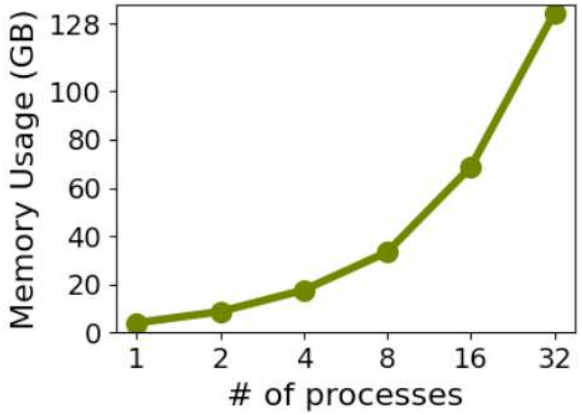
◆ 需要在完整的network snapshot上做分析，**难以动态地**模拟网络的变化

◆ 需要枚举所有可能的状态空间，**可扩展性较差**

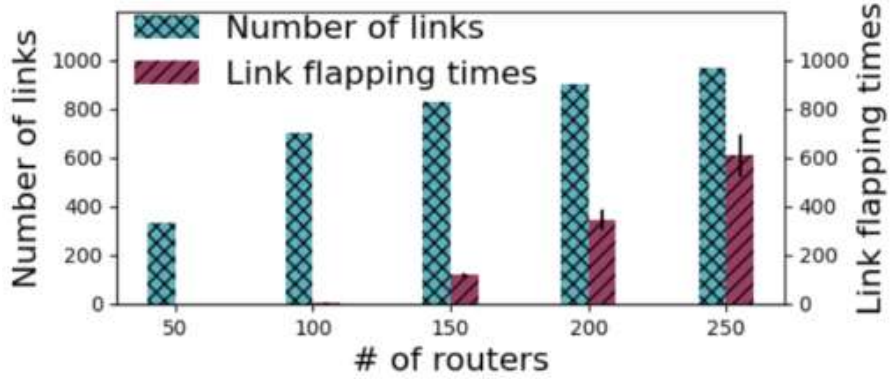
已有技术的可扩展性较差

- 已有仿真器的并行模式：每个进程分别仿真一个子网络
  - 同步开销高
  - 内存开销高





- 已有模拟器的并行模式：每个docker/VM运行一个虚拟路由器
  - 虚拟路由器的资源消耗高，难以集中优化
  - 对路由器的能力仿真不准确：发包能力、处理包能力



- 总结：无法仿真超过两千台路由器的网络

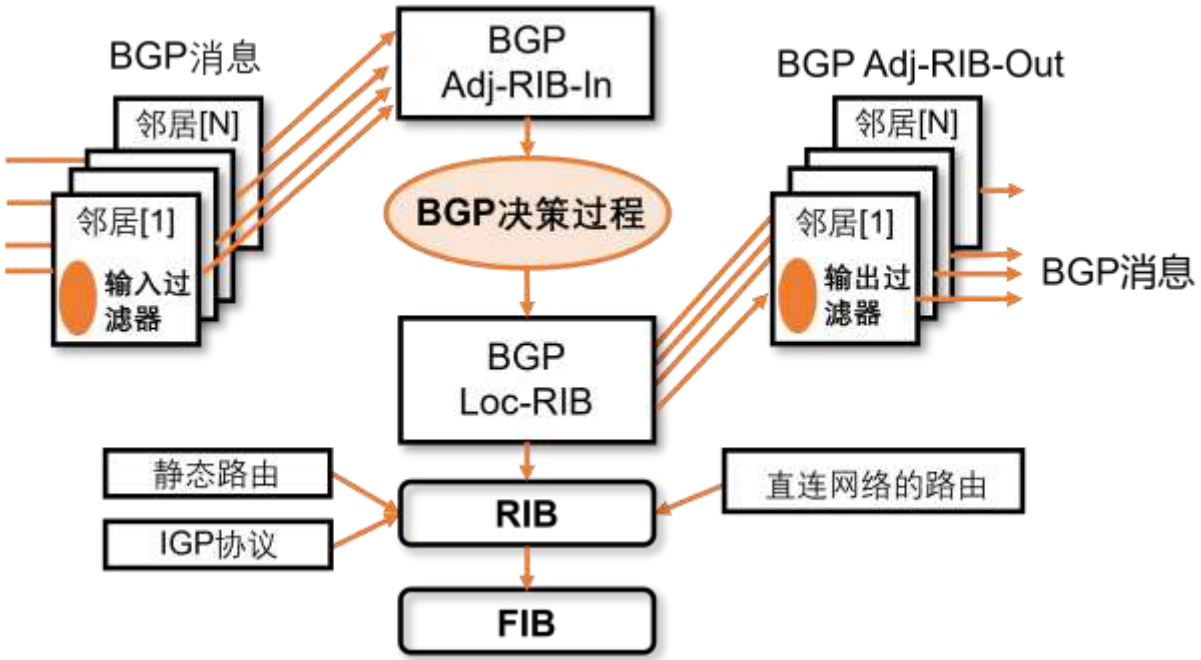
本质问题：并行模式粒度太粗，导致并行效率太低

核心设计

- 基于DOD思想设计面向数据的控制平面路由协议仿真系统，延续离散事件仿真的轻量性和准确性，并实现细粒度并行，提升可扩展性
- 目标：仅用单台CPU服务器即可仿真整个互联网AS级的行为
- 挑战：
  - 如何基于DOD建模众多控制面路由协议
  - 如何提升相对于真实网络的仿真保真度
  - 如何优化大规模网络仿真时的内存开销

关键设计#1：控制平面路由协议建模

- 以BGP为例

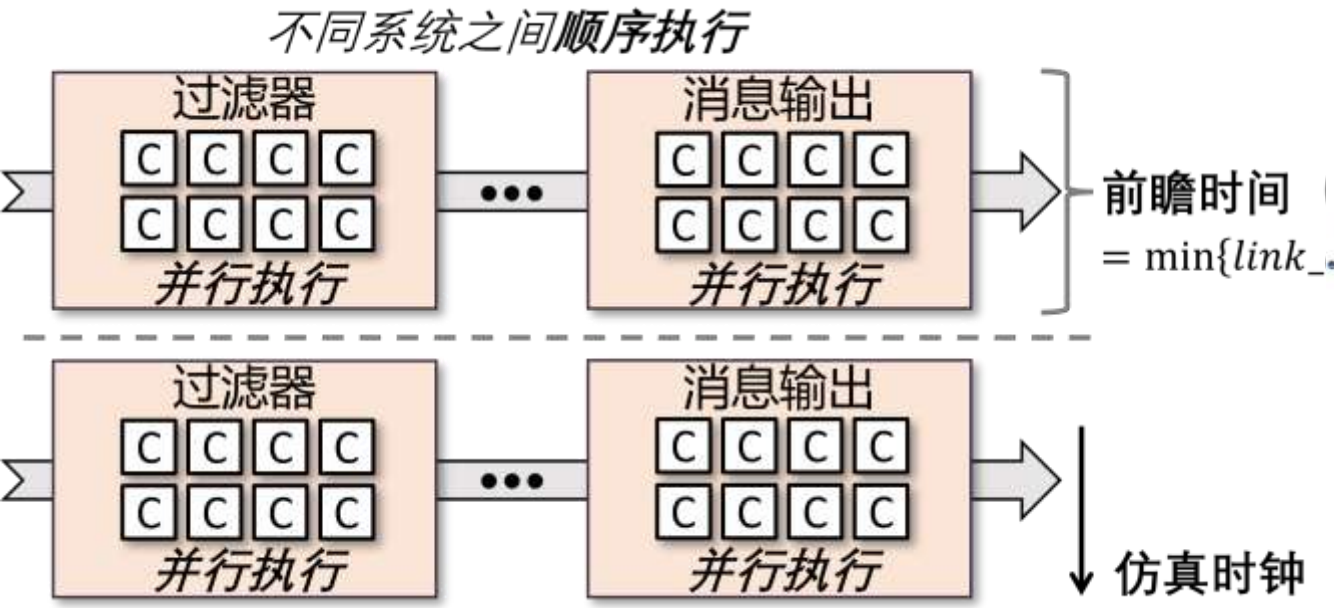


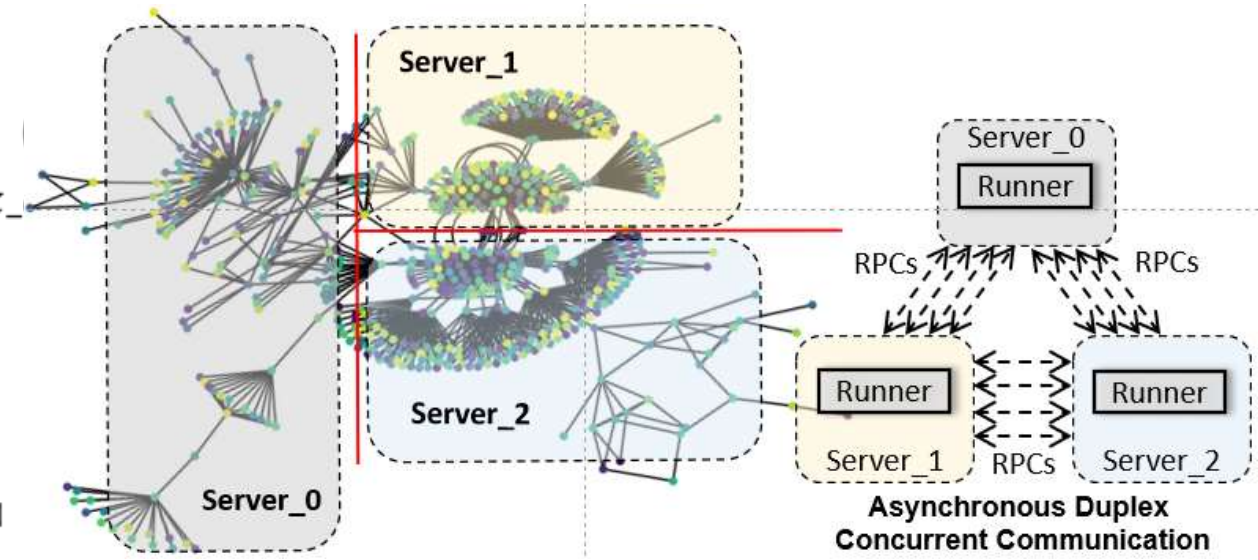
关键设计#2：支持真实网络配置

- 问题：如何在仿真器中高保真地模拟控制面协议、策略、配置
- Emulator是在Docker/VM中运行真实的设备固件（firmwares）

- 设计一种**配置parser**，支持将真实路由协议实现、配置和策略转换为DOCS中的逻辑与数据
  - 对于实现，参照RFC设计的，用实验证明
  - 对于配置，Batfish已经有分析工具，我们提出一个抽象的parser。【调研Batfish】
  - 对于policy，解析配置文件的语法树。基础：已知的policy能够准确的跑起来；进阶：未知的policy需要拟合，完全没有信息的，用同类别通用policy代表，best-effort
- 网络工程师可以使用与生产网络相同的管理工具和方法来与DOCS交互

关键设计#3：多线程并行与多机分布式并行





目录

- 离散事件仿真(Simulation)技术概述
- 面向数据设计的网络数据面仿真研究
- **面向数据设计的AI智算网络仿真研究**

AI智算网络设计空间

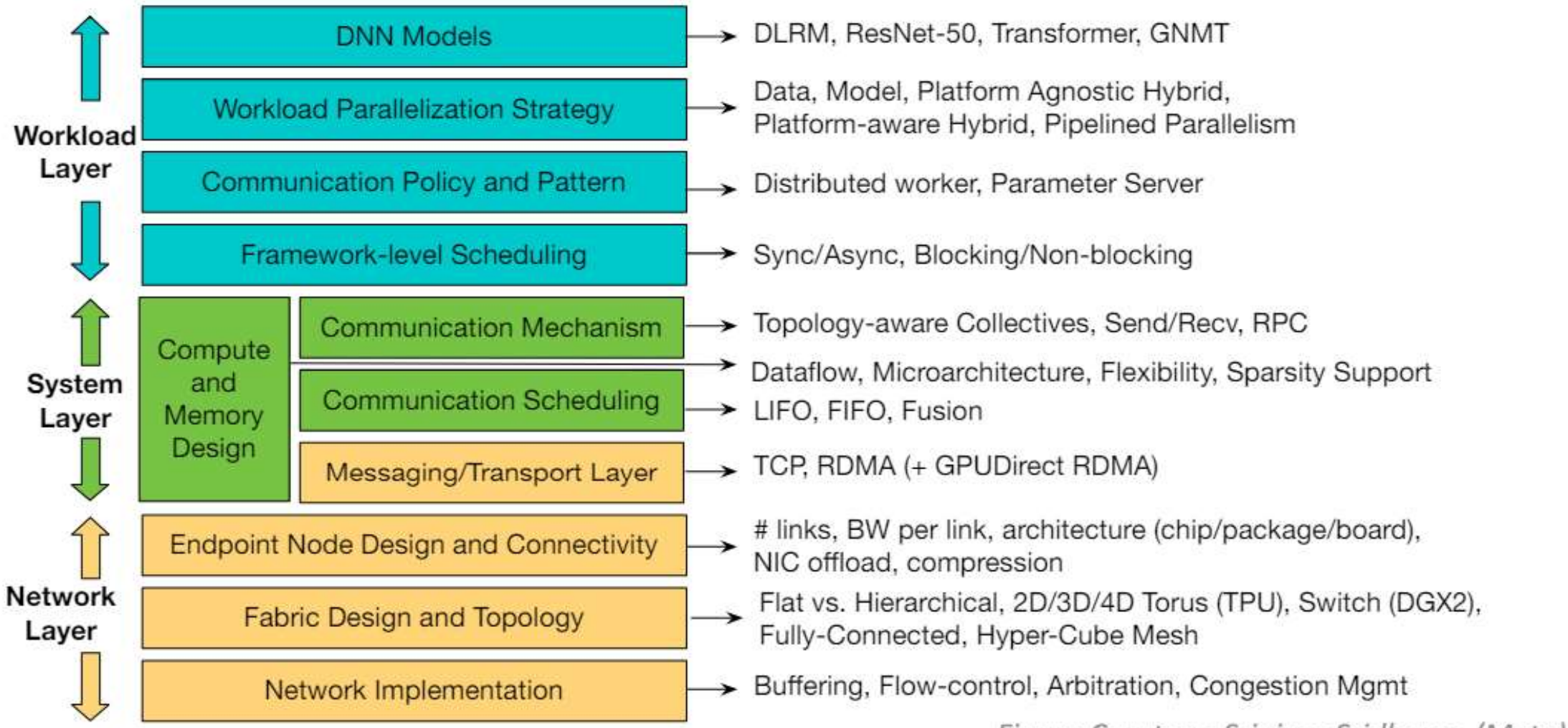
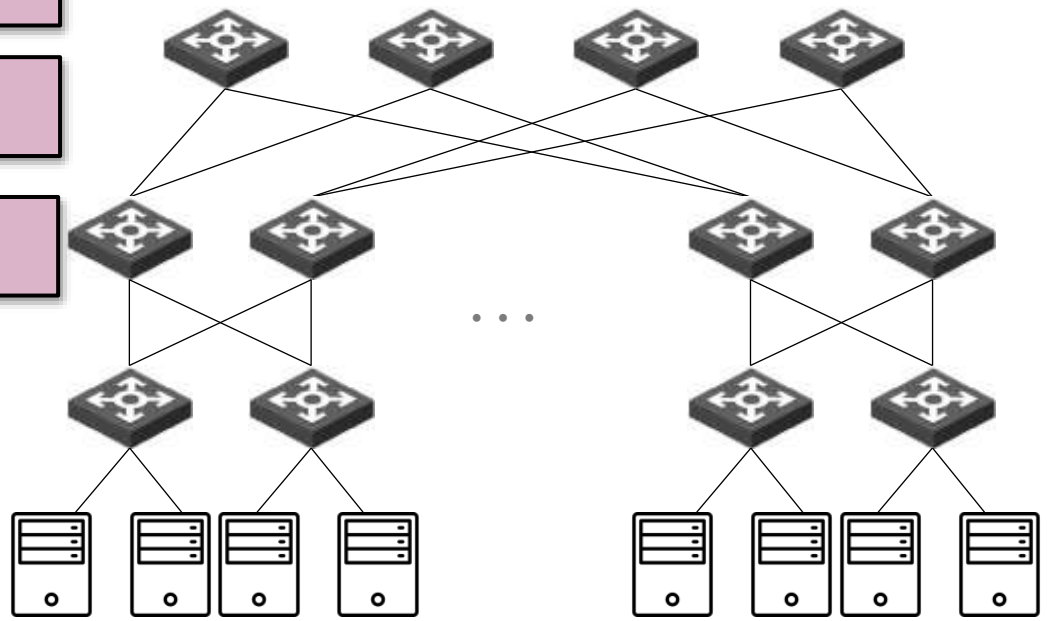


Figure Courtesy: Srinivas Sridharan (Meta)

AI智算网络对仿真的需求

- 从不同角度评估AI基础设施
  - GPU选择
  - 网络架构设计
  - 主机架构设计
- 以低成本测试各种优化技术
  - 参数调优
  - 新算法、新协议
- 一体化仿真引擎
  - 高保真度
  - 不是多个仿真引擎的简单组合

AI智算网络仿真对象



AI智算网络

- 路由、流量工程、容错
  BGP、ECMP
- 转发、流控
  PFC
- 网络拓扑结构
  FatTree、BCube

- 网络传输协议
  TCP、RDMA
- 拥塞控制算法
  DCTCP、DCQCN

AI通信协议栈

- GPU算力建模
- GPU显存建模

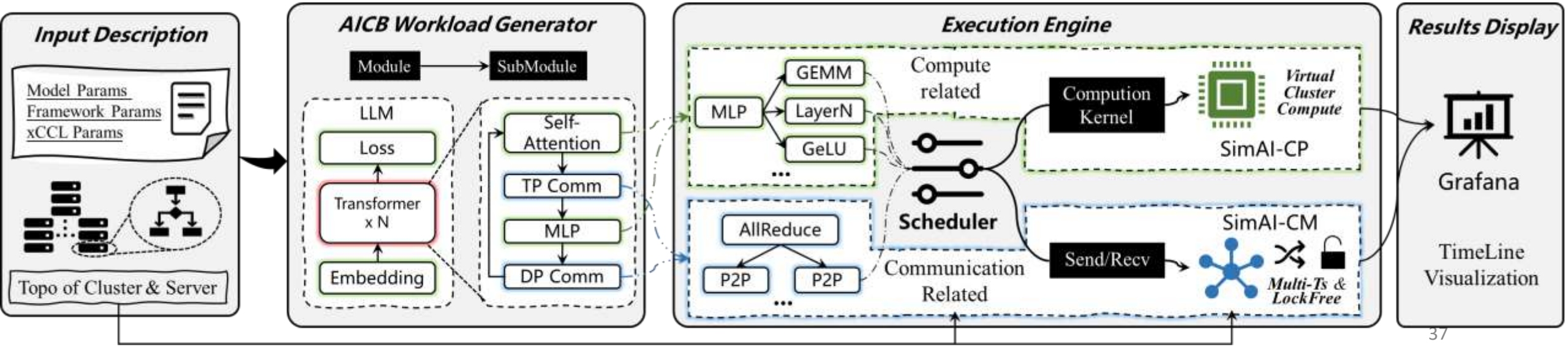
AI算力

AI Workload

- DNN模型结构 → 架构、层数、参数量
- 任务部署策略 → Job placement, Host utilization, Availability
- 训练并行策略 → 数据并行、模型并行、流水线并行、混合并行
- 集合通信机制 → Ring AllReduce、Parameter Server、RHD

面向大模型训练的仿真器SimAI

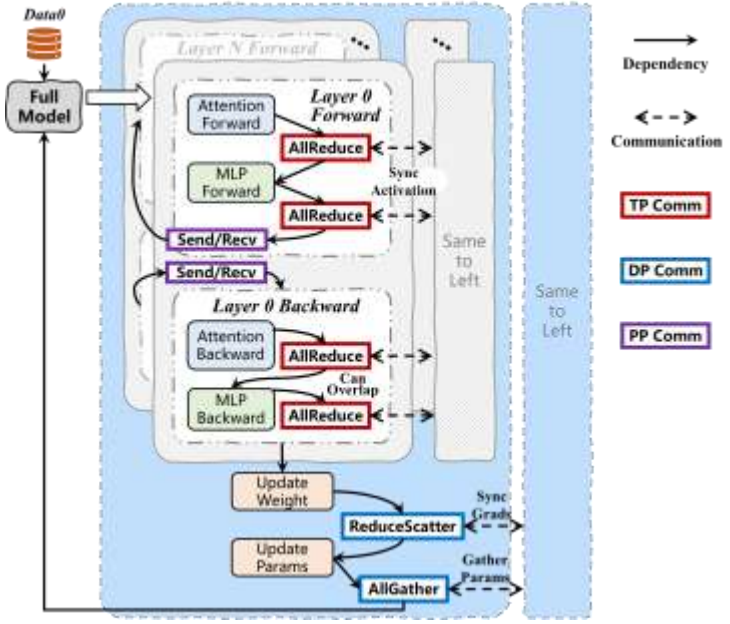
- SimAI的两大模块：模型训练的工作负载生成器； 基于网络仿真器的执行引擎
  - 工作负载生成器用于根据三类输入生成计算和通信负载： 1.集群参数（拓扑结构，GPU类型等）；2.模型参数（模型类型，并行策略等）；3.架构参数（训练框架，集合通信库等）
  - 执行引擎进一步执行计算/通信负载：1. 通信仿真模块中，集合通信被拆解为点对点通信，在网络仿真器中执行；2. 计算仿真模块拆解出计算算子，根据硬件算力推导计算时间

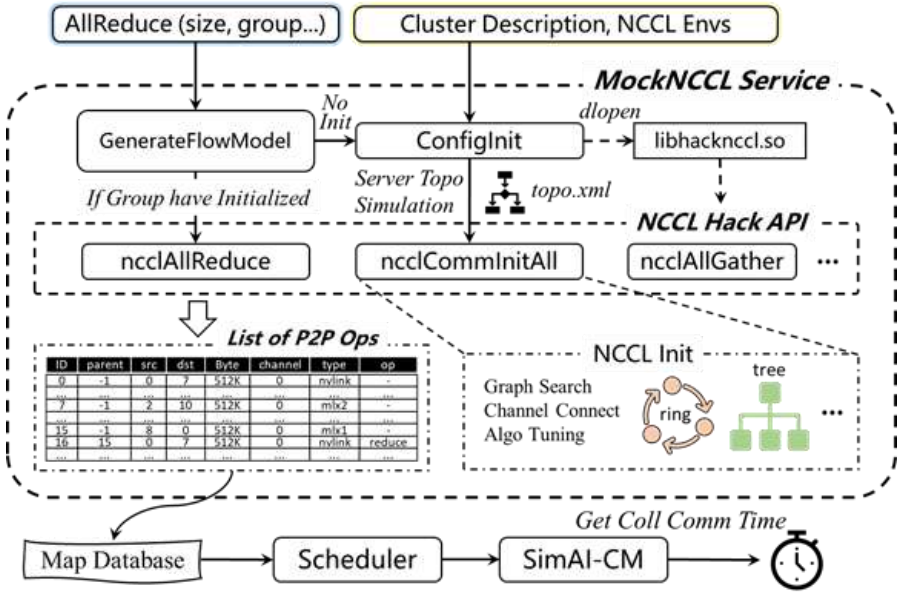


37

关键技术：负载精确生成 & 集合通信库模拟

- SimAI通过模拟模型训练过程中通信的生成和执行流程，实现对通信的精确仿真。关键在于：负载生成器计算出所有集合通信；集合通信库模拟产生点对点通信
  - 负载生成器将模型拆解到“层”的粒度（MLP等），根据模型参数计算每层的参数量，进而结合并行策略参数计算节点间的通信量
  - 集合通信库模拟基于大量专家经验，复现并模拟NCCL等通信库的算法和流程

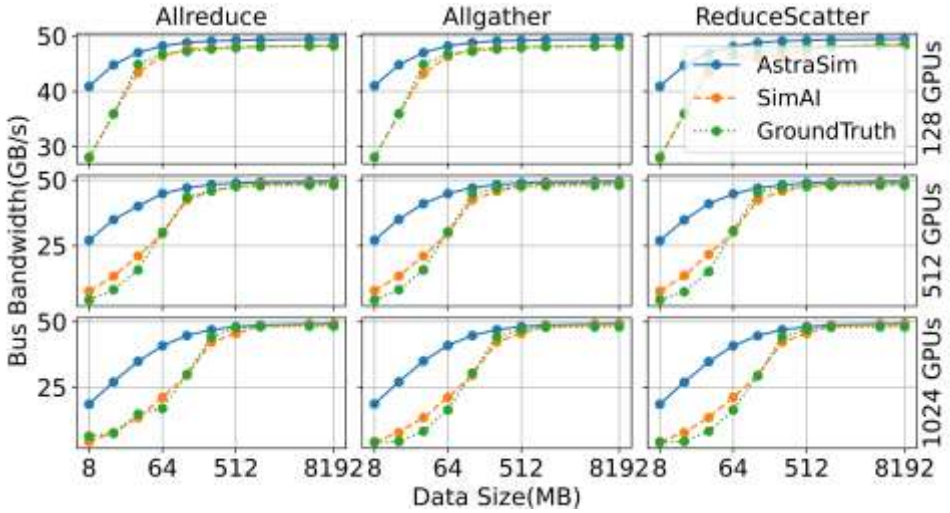




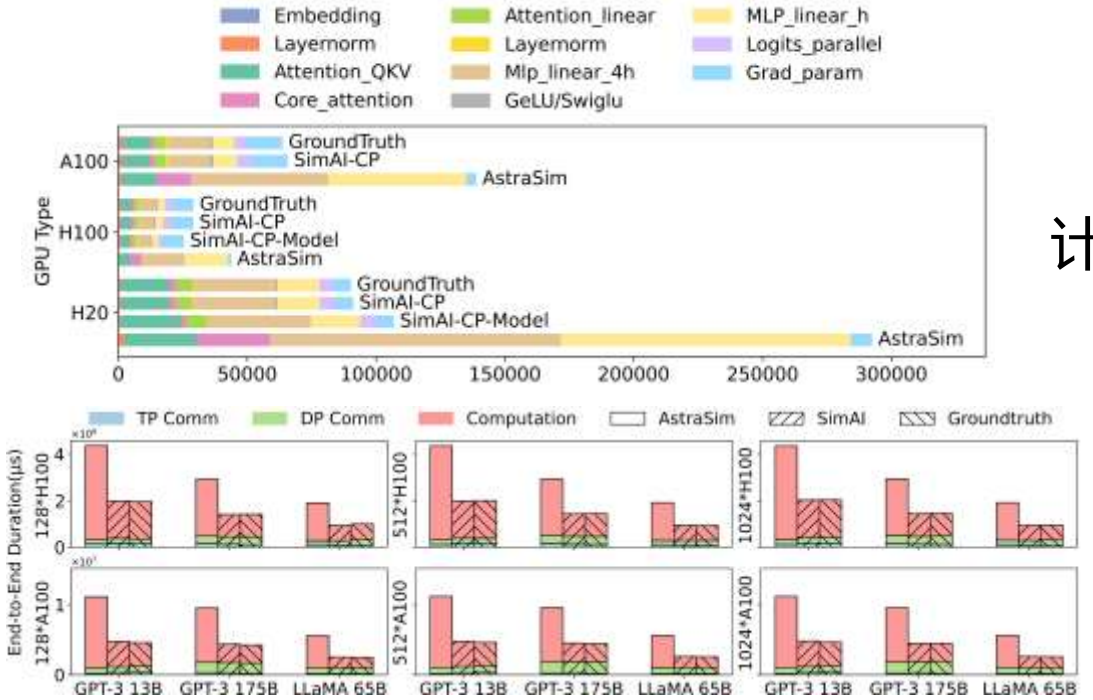
38

仿真性能评估

- SimAI的通信、计算、端到端仿真准确性均达到92%以上，较AstraSim取得巨大提升
  - 集合通信仿真中，SimAI能够拟合真实情况，尤其小规模通信中相比AstraSim提升20.4倍
  - 计算过程仿真中，SimAI对各GPU的仿真准确性达96-99%，相比AstraSim提升49-224%
  - 端到端过程中，SimAI仅有2%误差，相比AstraSim提升36.1倍



通信仿真性能评估

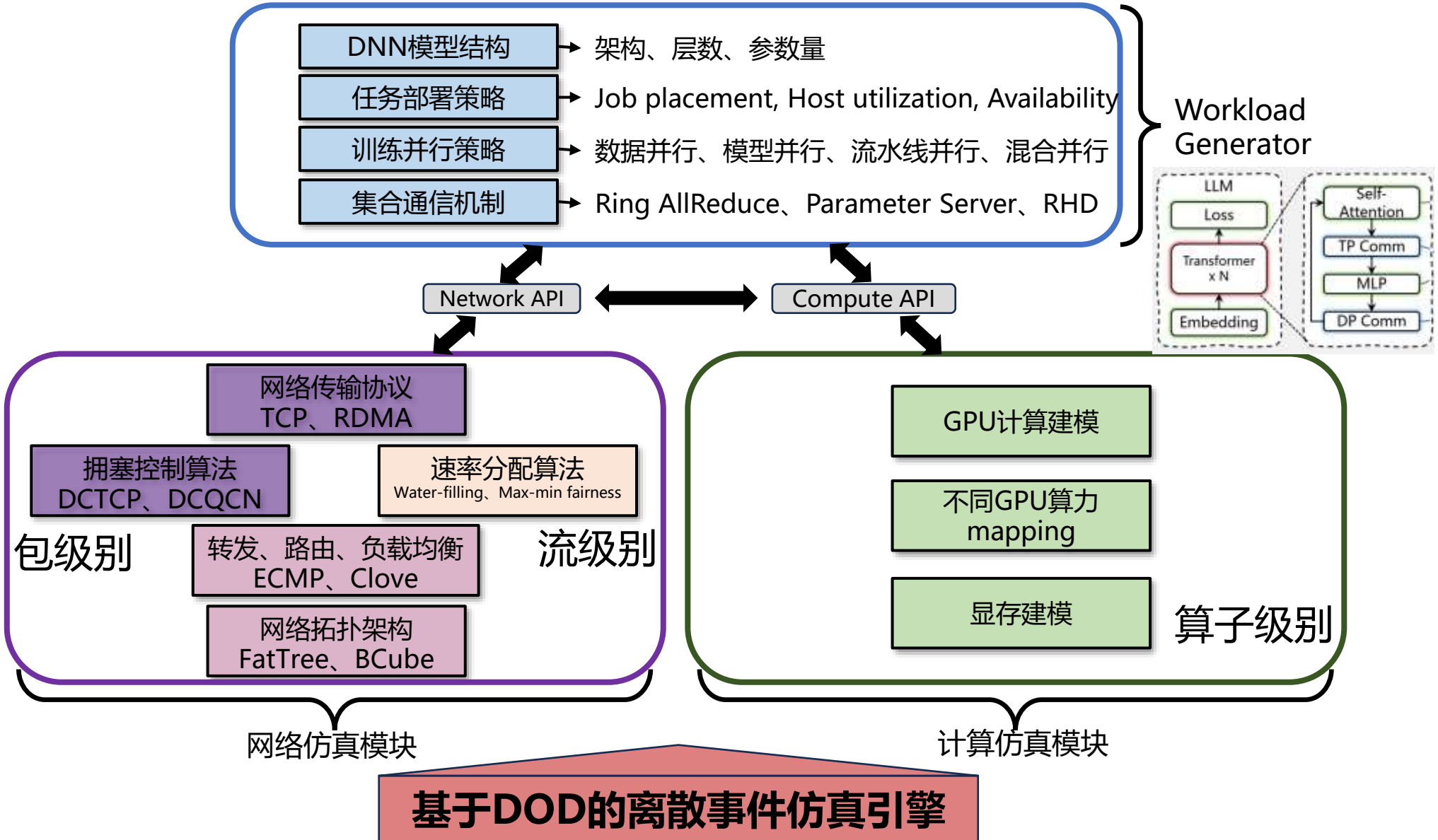


计算仿真性能评估

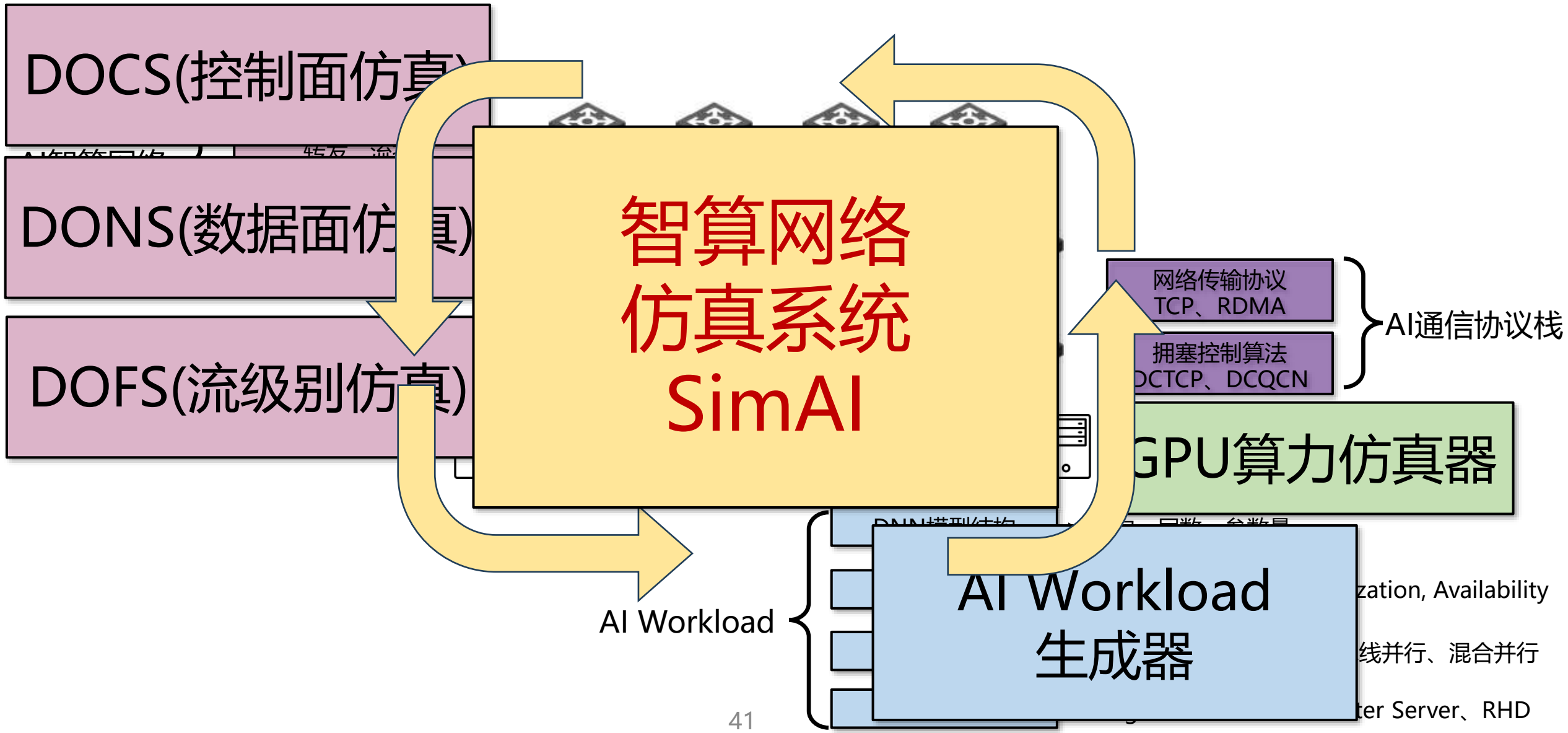
端到端性能评估

39

面向数据设计的AI智算网络仿真系统



总结


DOCS(控制面仿真)
DONS(数据面仿真)
DOFS(流级别仿真)
智算网络
仿真系统
SimAI
网络传输协议
TCP、RDMA
拥塞控制算法
DCTCP、DCQCN
AI通信协议栈
GPU算力仿真器
AI Workload
AI Workload
生成器
zation, Availability
线并行、混合并行
ter Server、RHD