CCF ChinaNet 2023

DONS：面向数据设计的网络离散事件仿真引擎

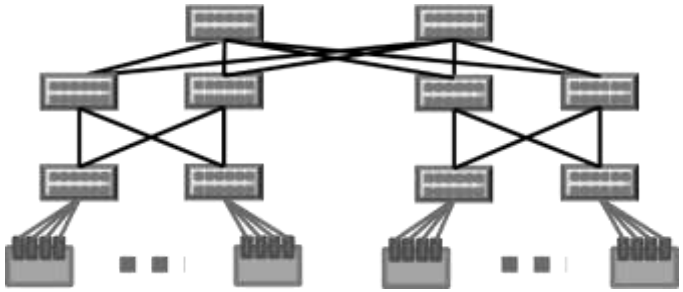
中关村实验室　高凯辉

2023年8月

研究背景

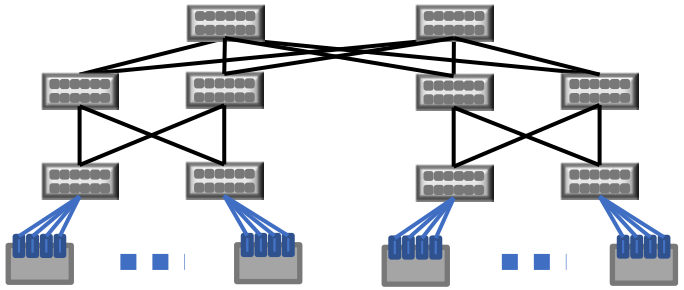
- **网络仿真**是验证网络创新设计（如新协议、新算法）的关键技术

- 网络规划
- 变更
- 新协议
- 新算法
- …

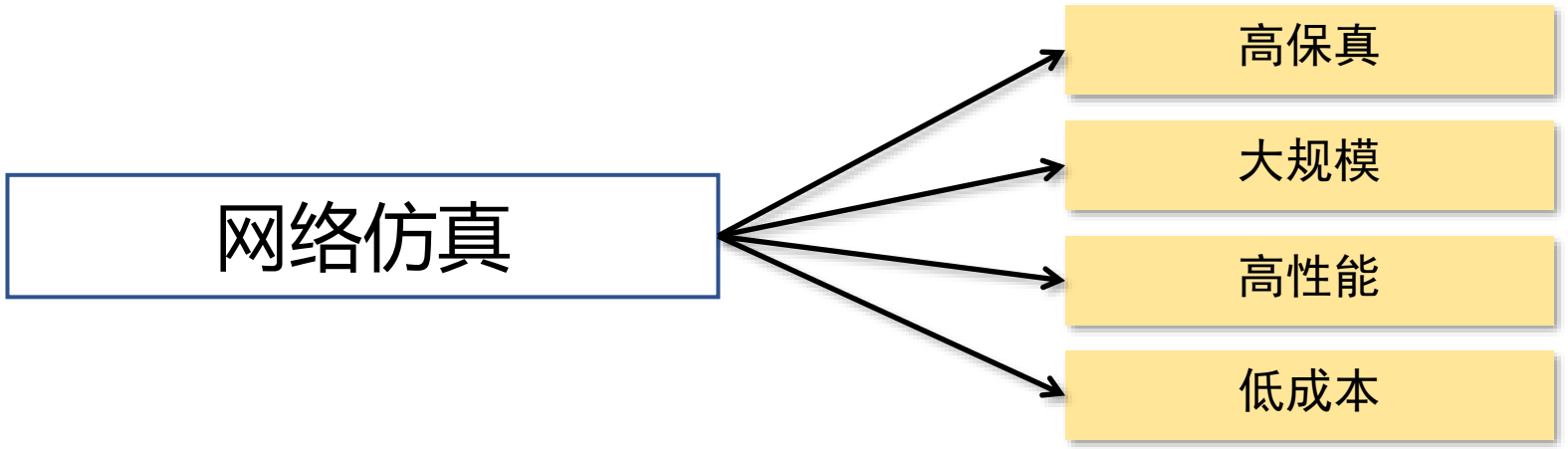


虚拟网络

- 部署



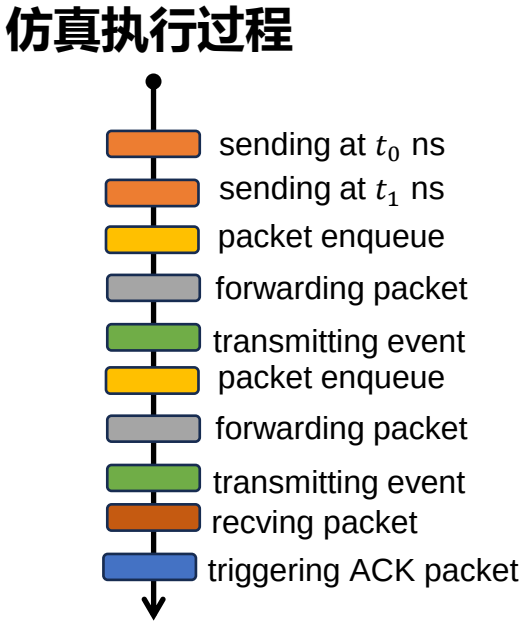
物理网络



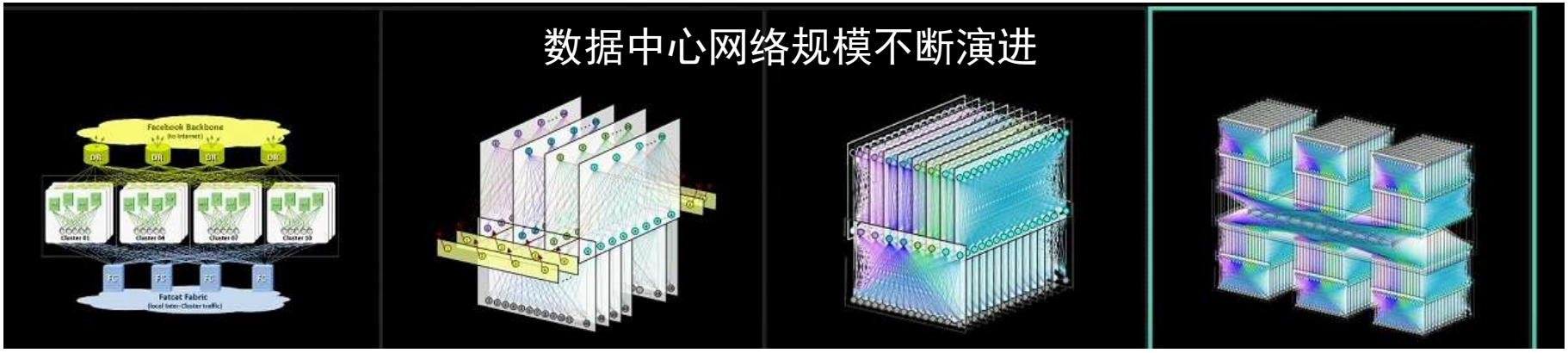
离散事件仿真技术

- 网络**离散事件仿真（Discrete Event Simulation, DES）**是网络仿真主流范式
- 如ns-2/3，OMNet++，OPNET等

  - **高保真**
  - **低成本**



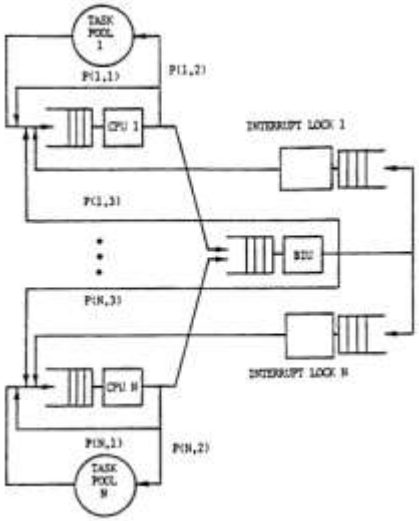
- **可扩展性较差**
  - OMNet++仿真常规DCN的一秒需要**九天**时间



相关工作

1. 连续时间仿真（Continuous-time simulation）

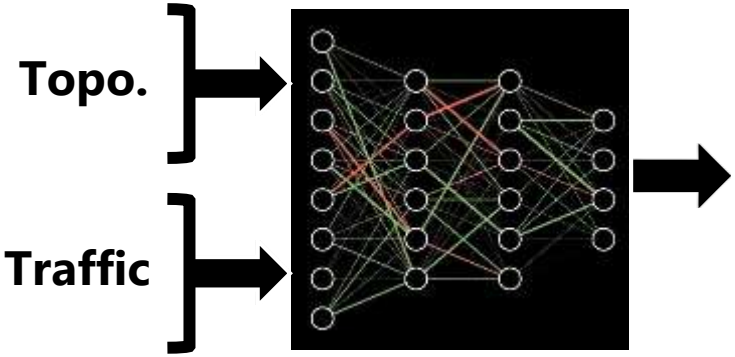
◆ 排队论，网络演算，控制论等技术



- ➢ 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

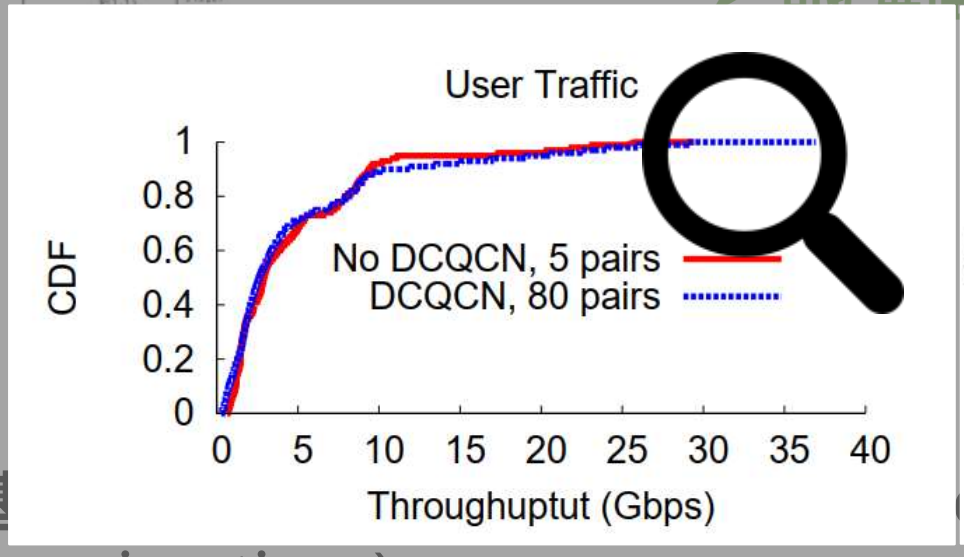


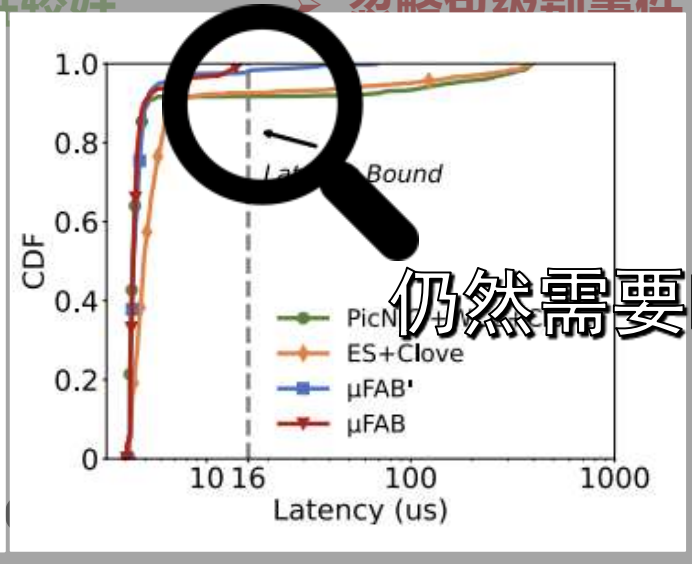
- ➢ 快速
- ➢ 需要GPU加速
- ➢ 固有的误差
- ➢ 扩展性仍然被DES限制

相关工作

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

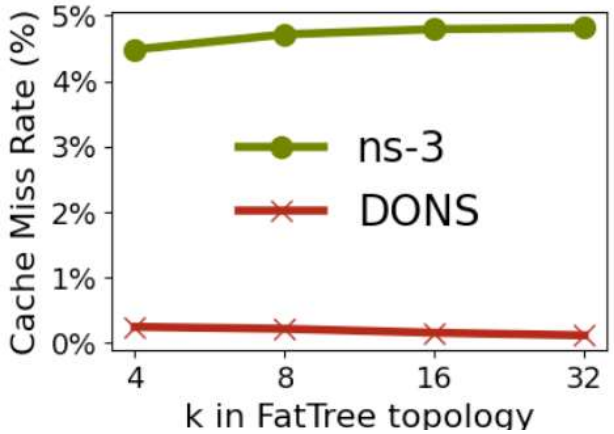
approximation）

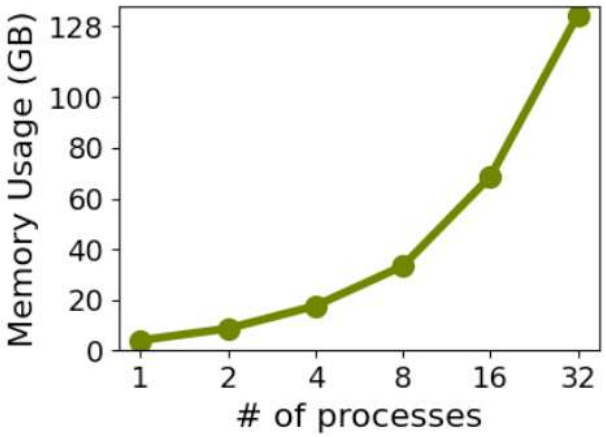
Insight: 问题不在于DES仿真的计算负载，
而是计算资源未被充分使用

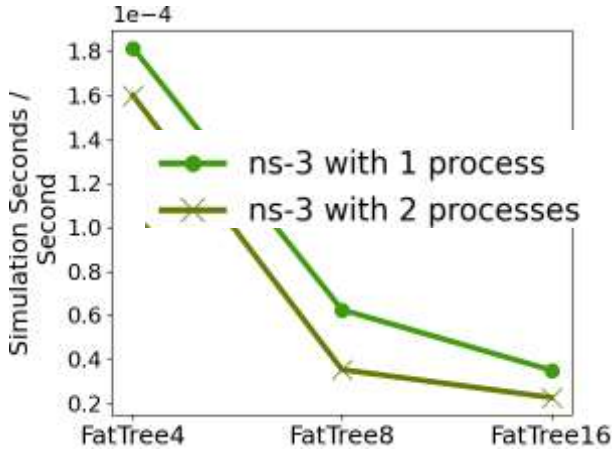
Performance Metrics (e.g., delay)

- 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

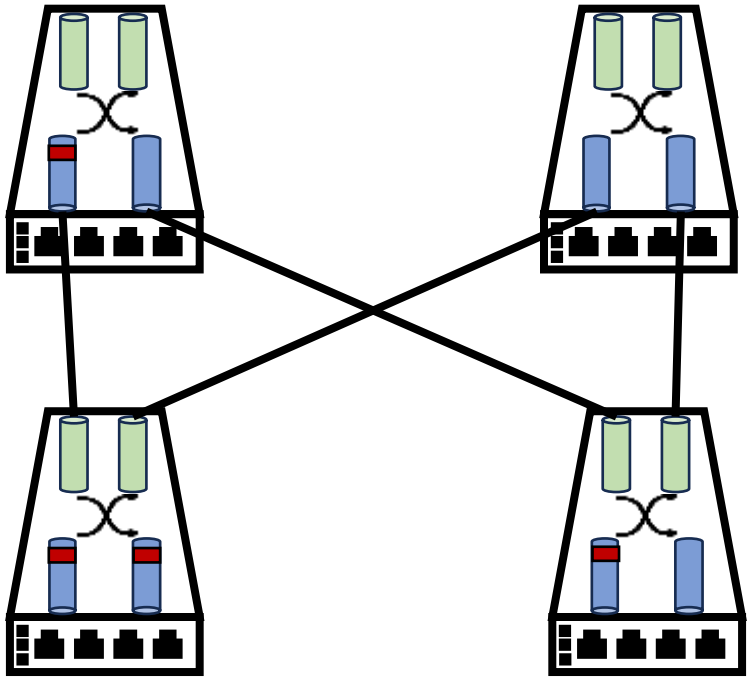
2. Poor memory efficiency:
使用32进程仿真Fattree (k=32)
需要**~5TB**

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

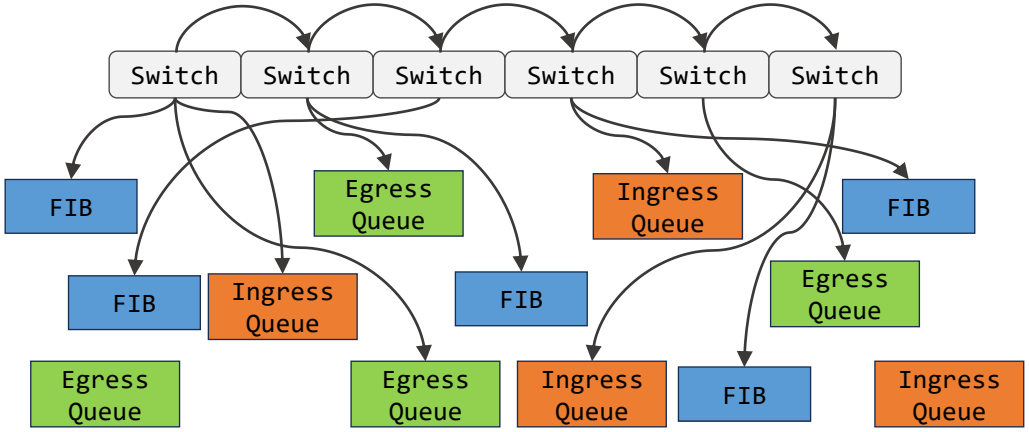
ns-3

OMNeT++

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

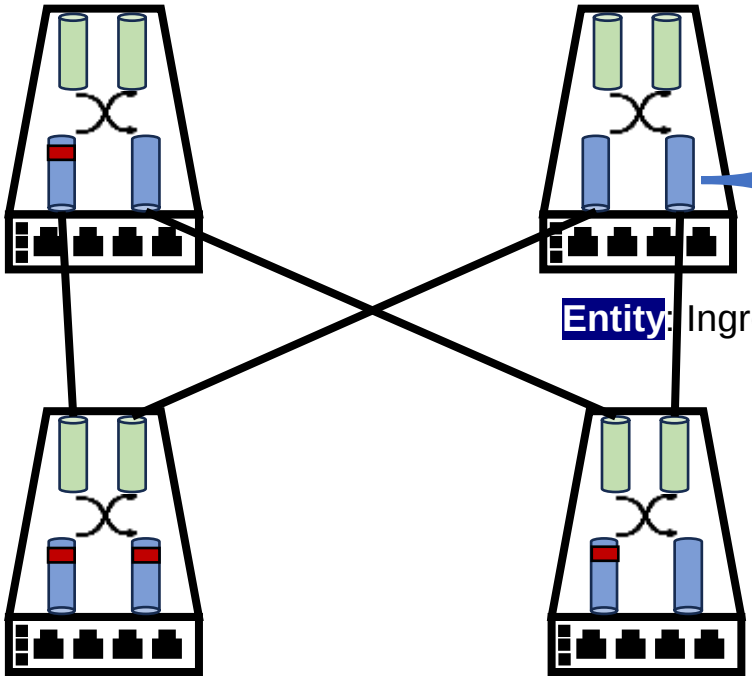
- 主流编程范式
- 数据与逻辑组合成对象
- 数据散列在内存中

- **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：**Data-oriented Design (DOD)**
- DOD思想的实现：Entity Component System (ECS)架构

Madrona

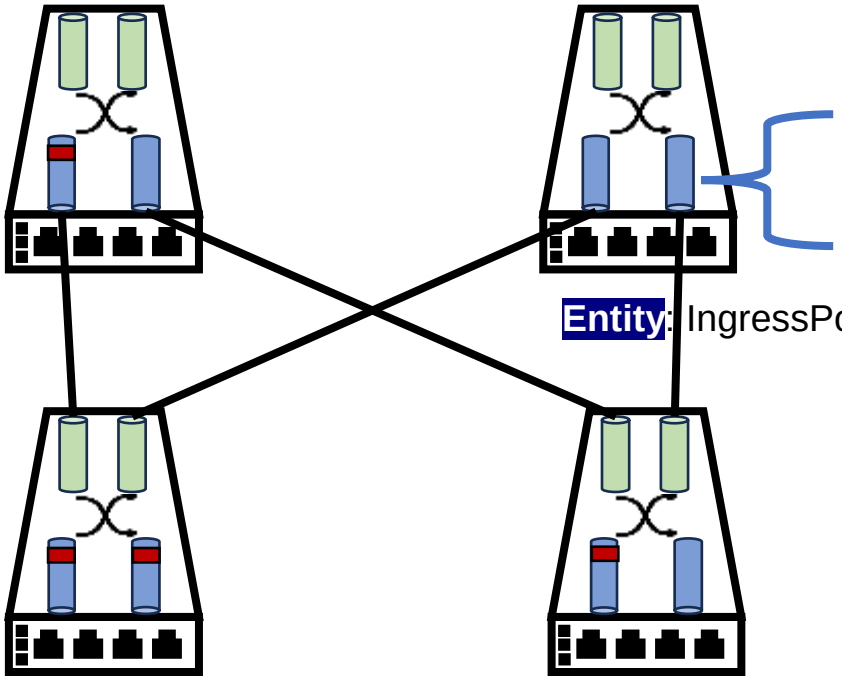



```
// Data-oriented Design
struct IngressQueue : Component { ... }
struct Ptr<EgressQueue> : Component { ... }
struct FIB : Component { ... }
```

提升OOD：DOD和ECS

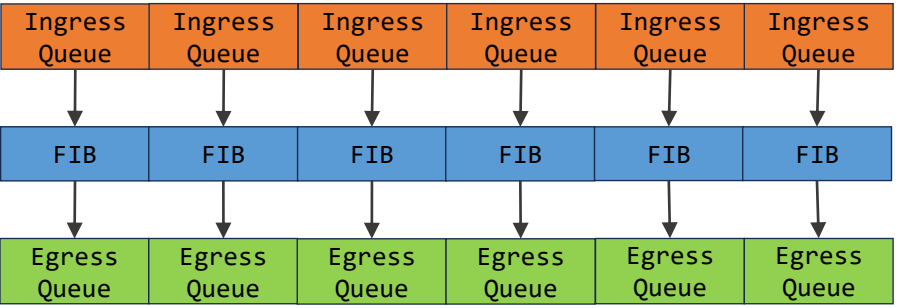
- 视频游戏领域中的解决方案：**Data-oriented Design (DOD)**
- DOD思想的实现：Entity Component System (ECS)

Madrona




```
// 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);
  }
}
```

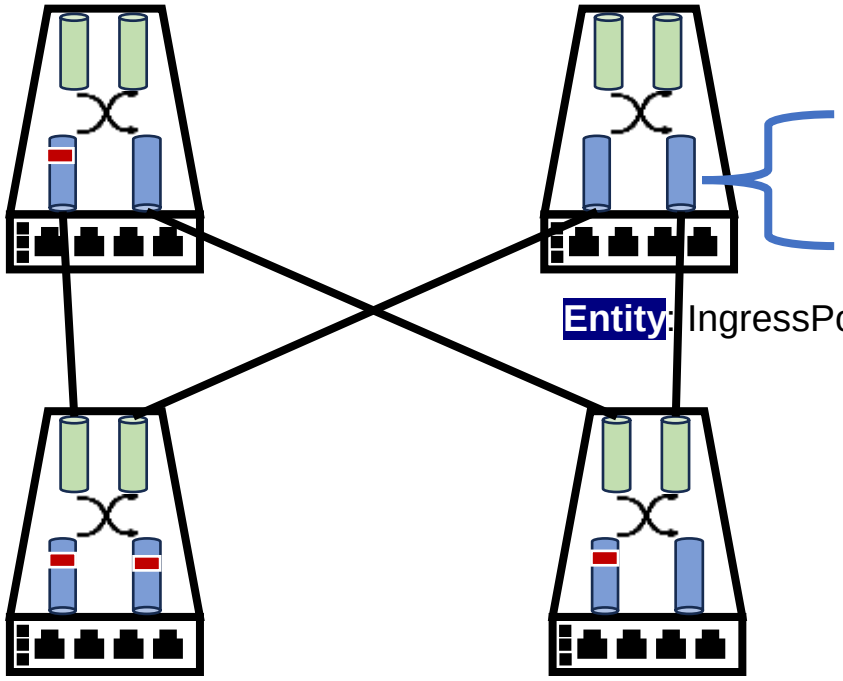
提升OOD：DOD和ECS



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**

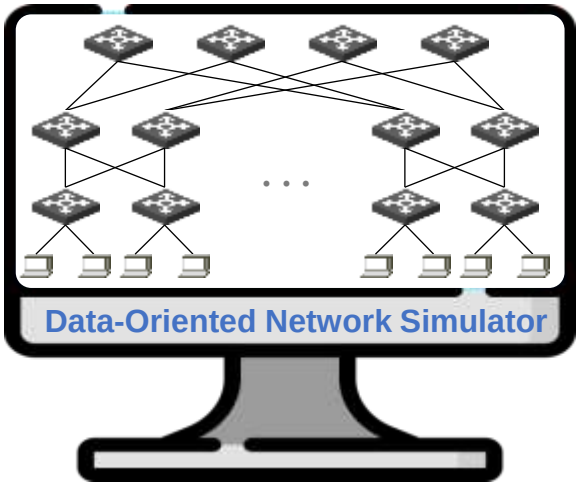



```
// Data-oriented Design
struct IngressQueue : Component { ... }
struct Ptr<EgressQueue> : Component { ... }
struct FIB : Component { ... }
void Forwarding_System(IngressQueue inq,
FIB fib_table, Ptr<EgressQueue> outqueues)
{
  foreach (var p in inq) {
    int index = fib_table.lookup(p.dst_ip);
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

研究挑战

- 将DOD思想应用于网络离散事件仿真，实现Data-Oriented Network Simulator (DONS)，面临着诸多挑战：

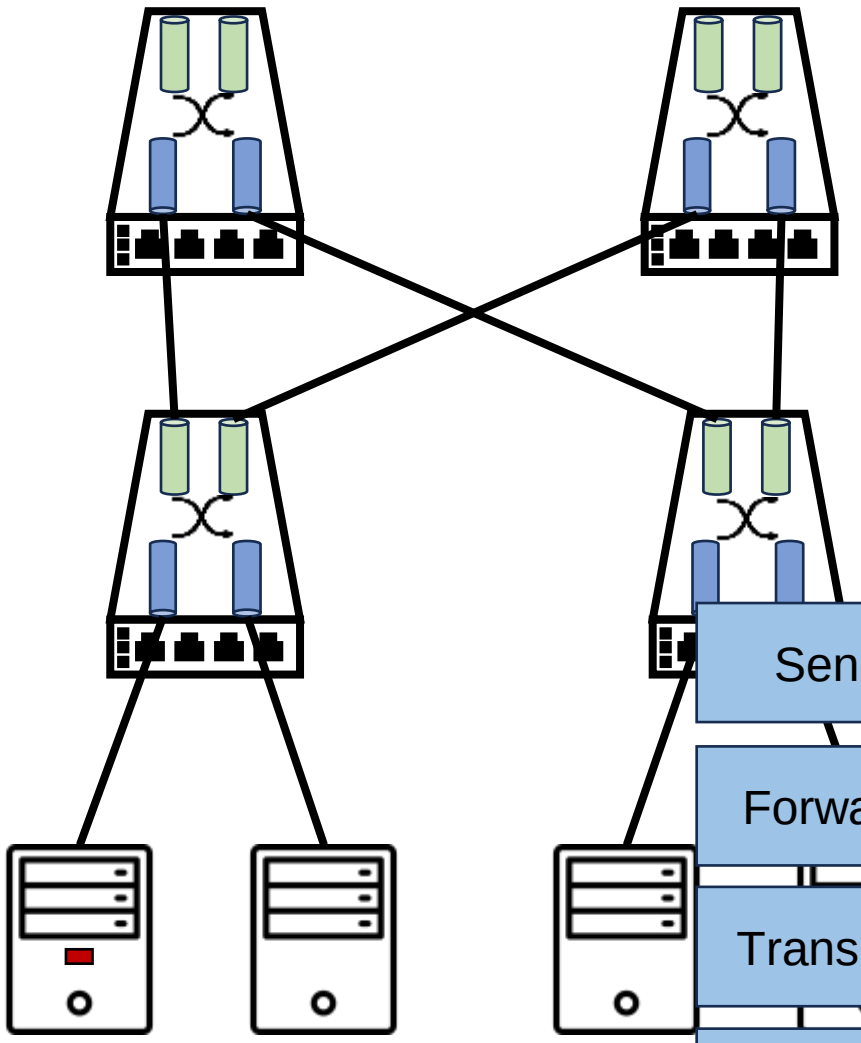
1. 如何基于ECS架构对复杂的网络设备、协议**建模**?



3. 如何划分仿真任务以实现**高效的分布式仿真**?

2. 如何在**保证仿真正确性**的同时**降低同步开销**?

DONS：网络建模

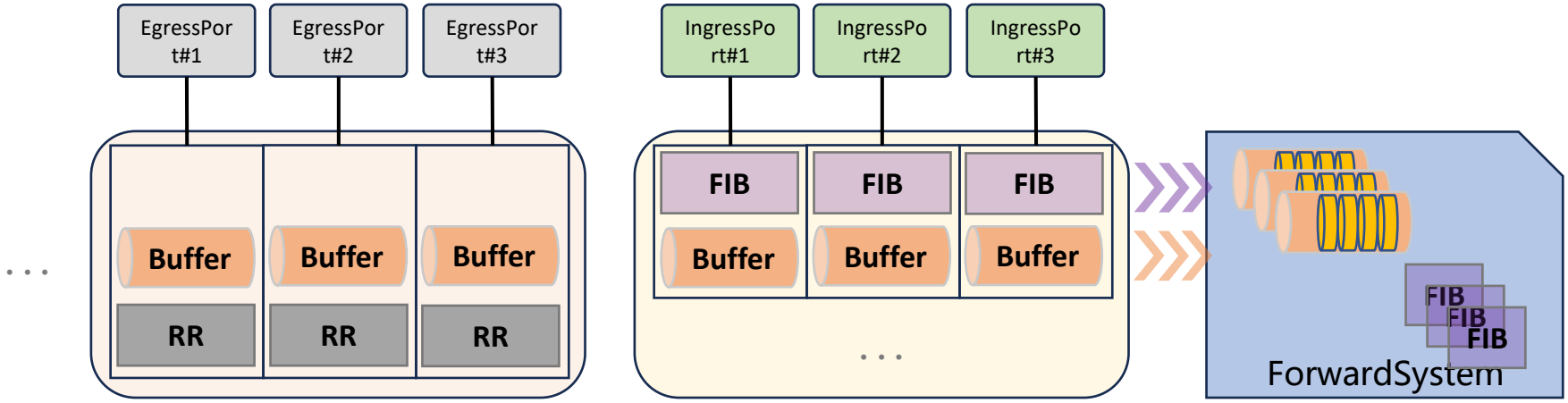


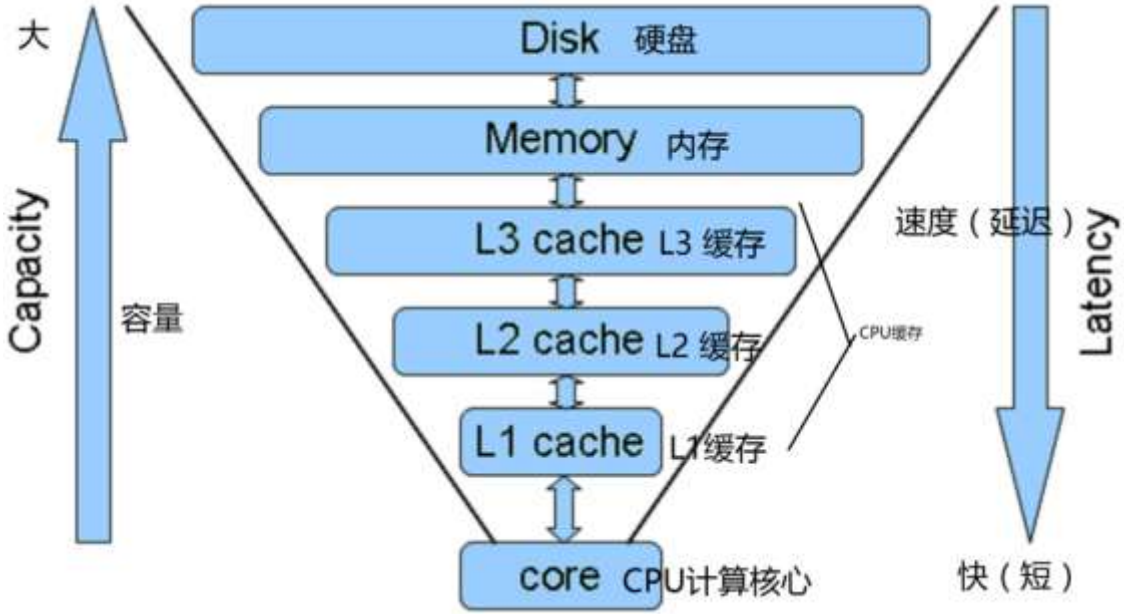
4 Entities	Components
Sender	IP, Flows, DCTCP, Stats
Receiver	IP, Buffer, RWND, Stats
IngressPort	MAC, Buffer, FIB, Stats
EgressPort	MAC, Buffer, RR, Stats

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

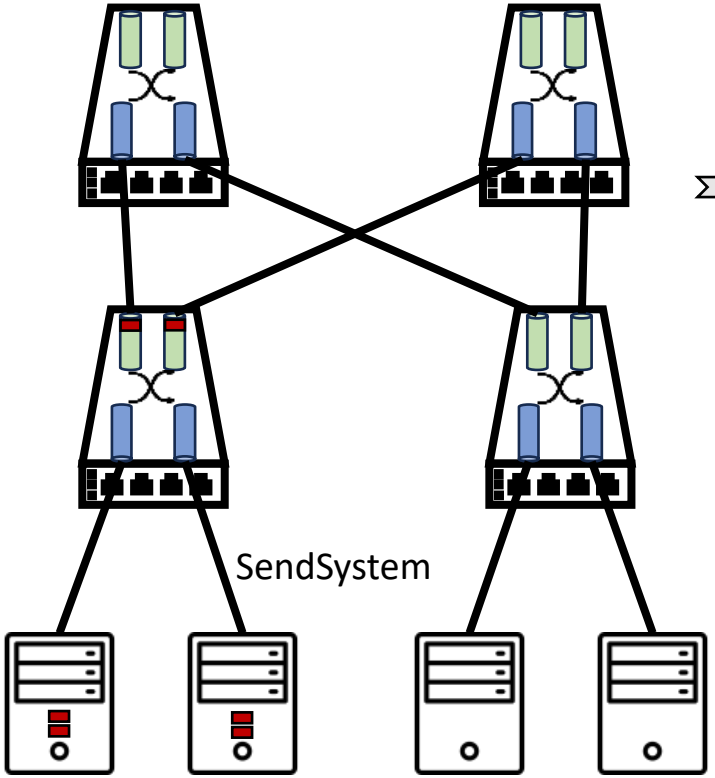
DONS：数据布局

- Packet作为component而不是entity

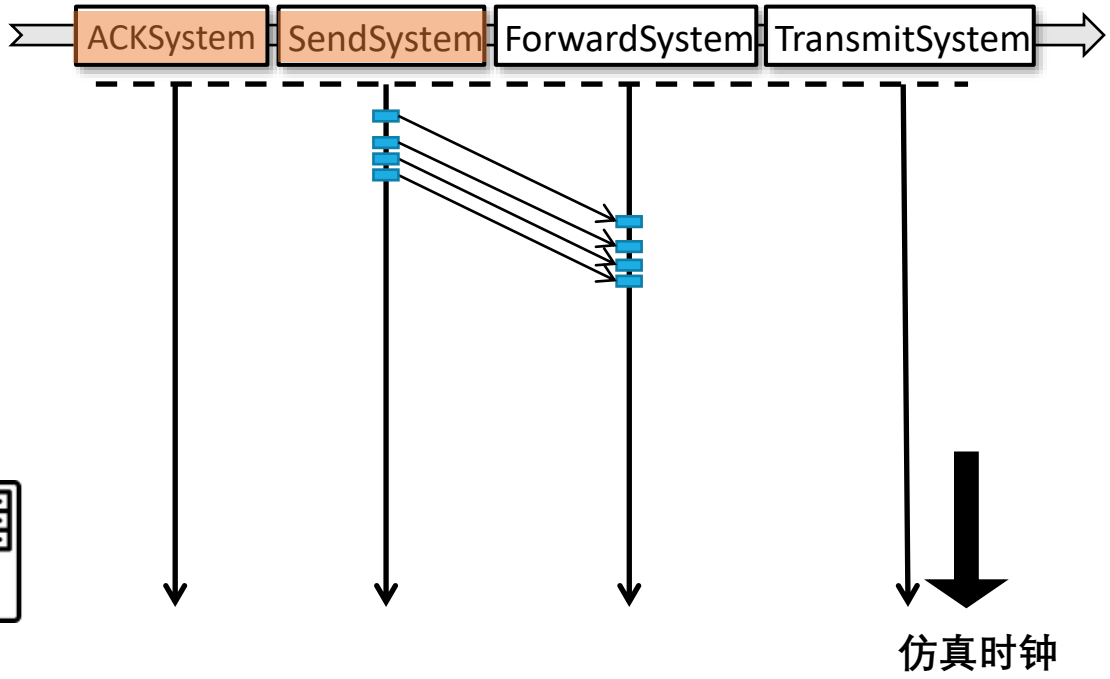




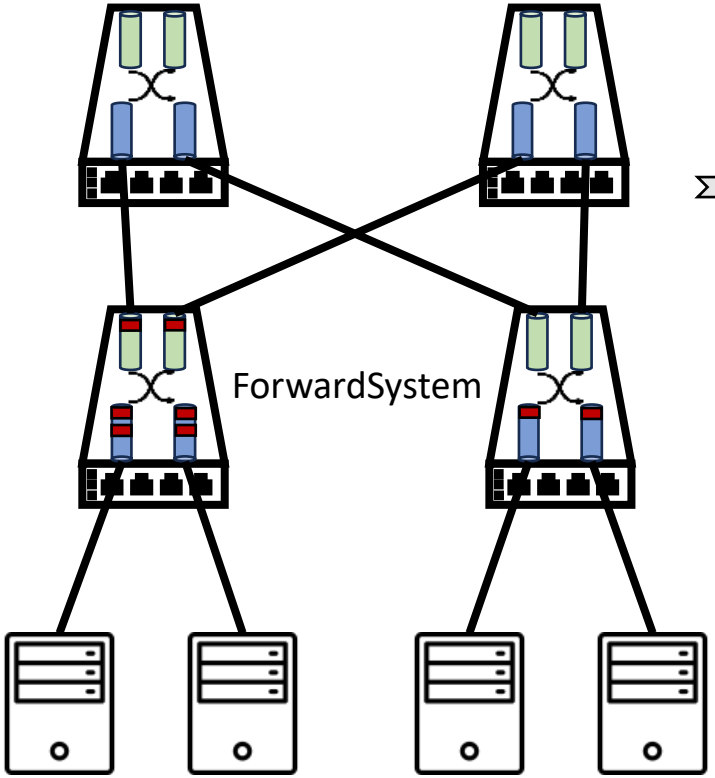
DONS：线程模型



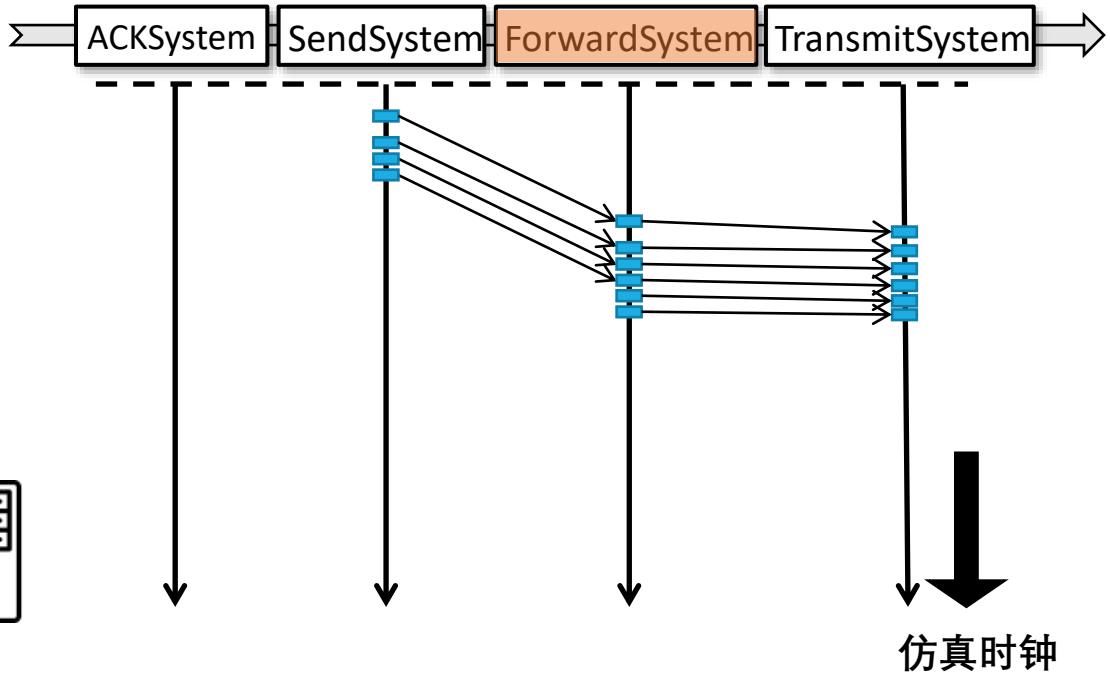
- 1. 顺序执行四个systems
- 2. 所有设备的同一system在多核CPU上并行执行



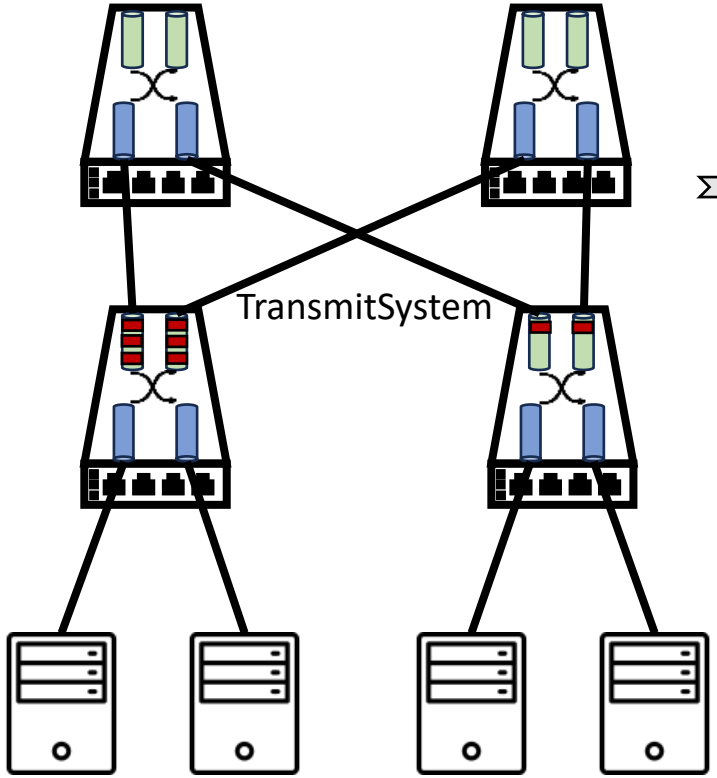
DONS：线程模型



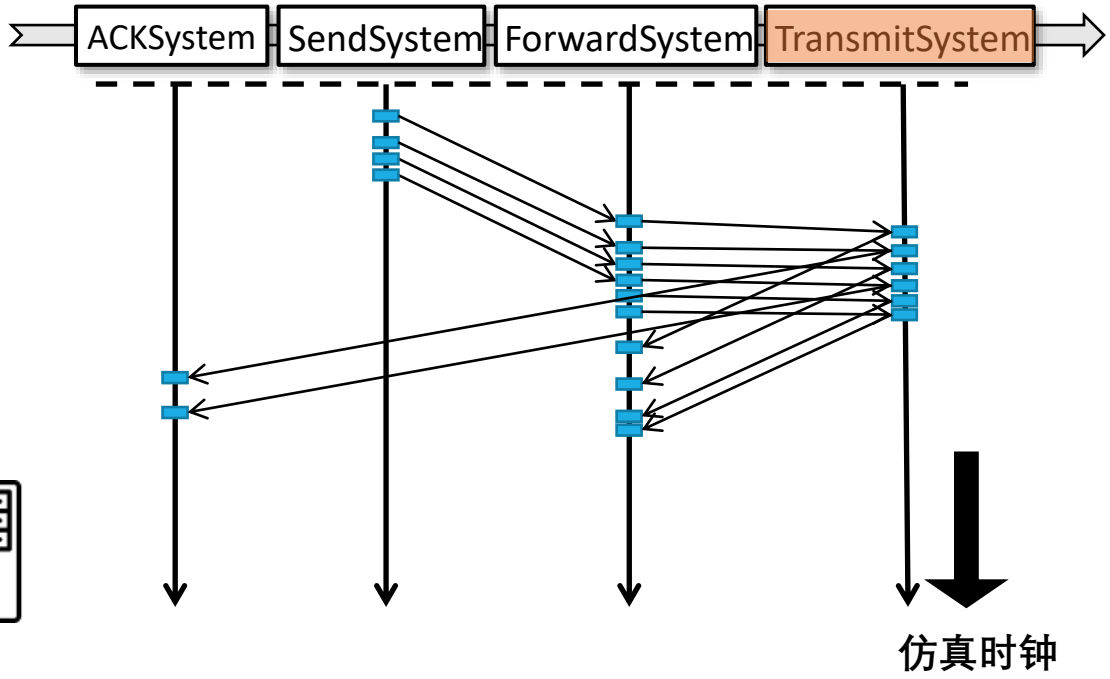
- 1. 顺序执行四个systems
- 2. 所有设备的同一system在多核CPU上并行执行



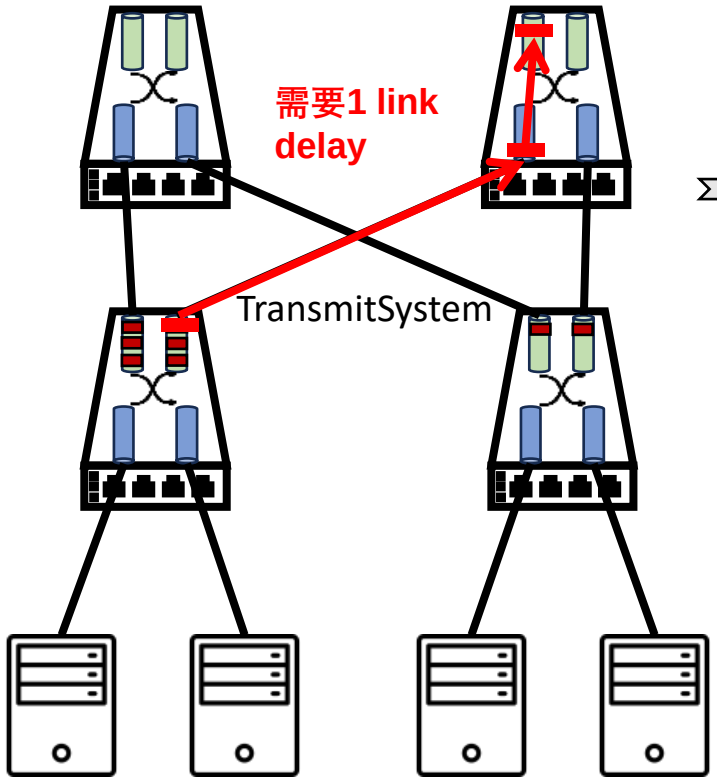
DONS：线程模型



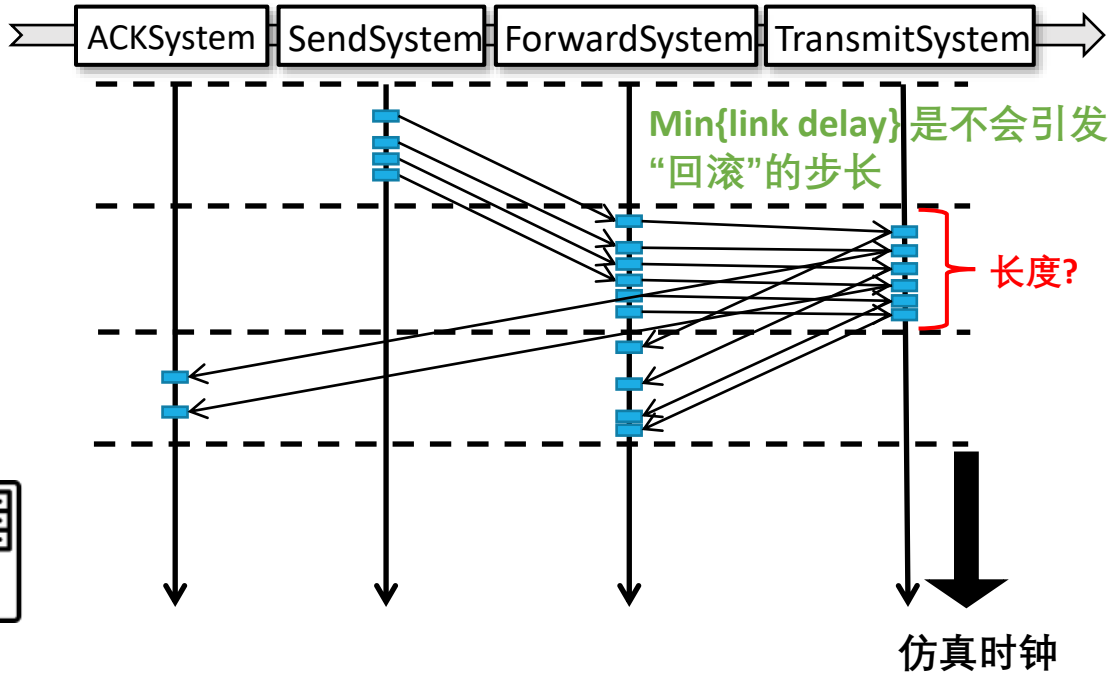
- 1. 顺序执行四个systems
- 2. 所有设备的同一system在多核CPU上并行执行



DONS：线程模型



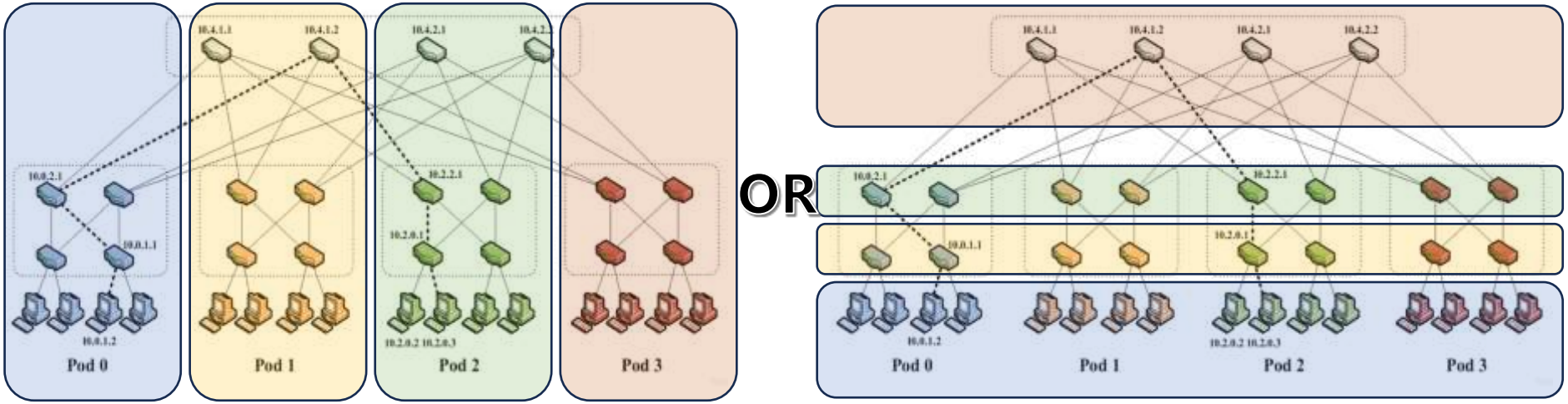
- 1. 顺序执行四个systems
- 2. 所有设备的同一system在多核CPU上并行执行



DONS：分布式仿真

- 无法直接将该线程模型应用到一个集群分布式仿真中

- 如何在多台机器之间划分仿真任务(网络拓扑)以**最小化仿真完成时间**，特别是对于不规则的网络拓扑?

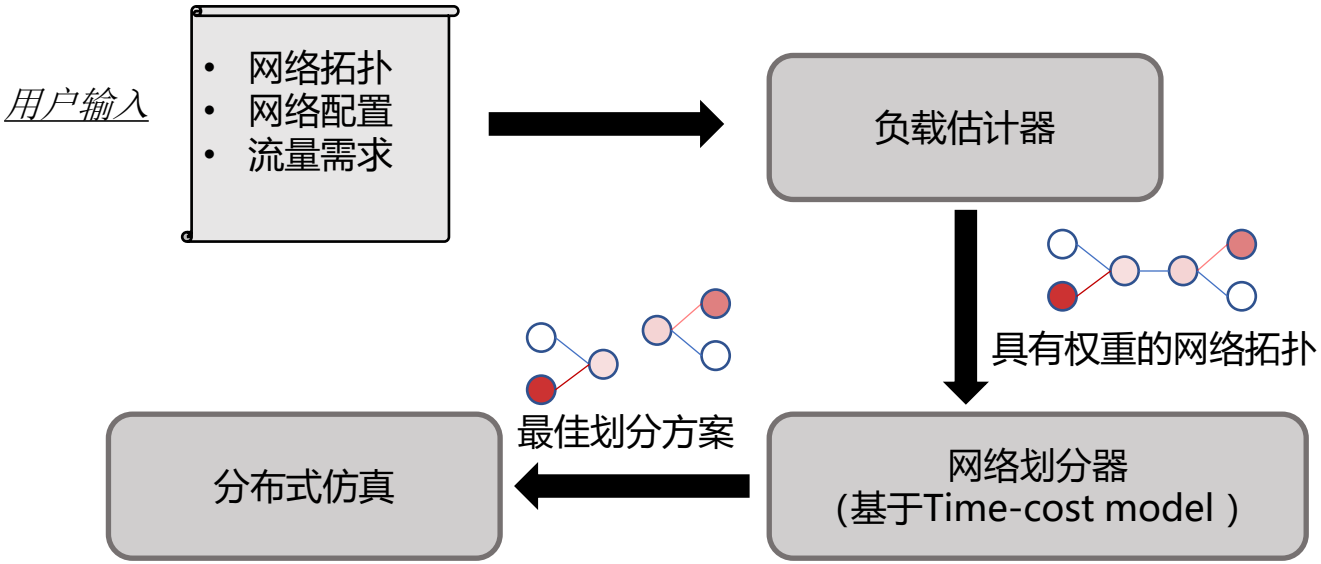


DONS：分布式仿真

- 无法直接将该线程模型应用到一个集群分布式仿真中

- 如何在多台机器之间划分仿真任务(网络拓扑)以**最小化仿真完成时间**，特别是对于不规则的网络拓扑?

>>>需要一个**仿真完成时间估计模型（Time-cost model）**来提前评估划分方案!



分布式并行模块工作流程

DONS：仿真完成时间估计模型

- 本研究发现以下因素会影响仿真完成时间：

1. 网络仿真任务的流量模式；
2. 集群的计算/通信能力。

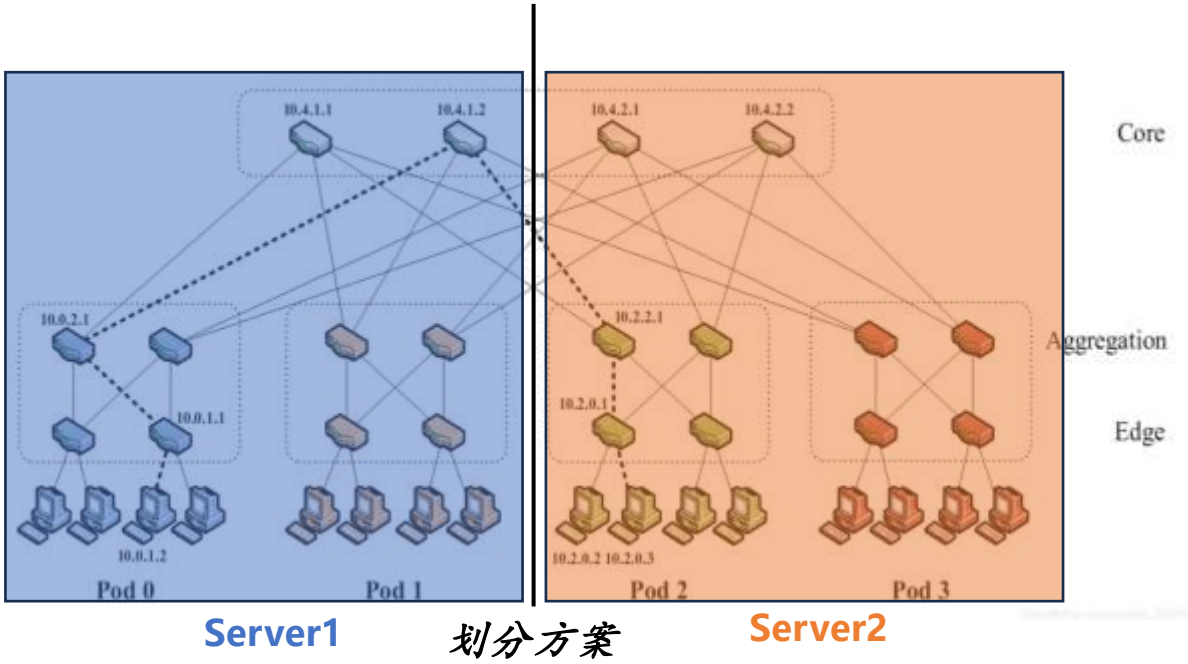
结合这些因素，我们提出了一个精确的Time-cost model来评估某个划分方案

计算开销　　通信延迟

机器1的仿真完成时间　$T_1 = \frac{E_1}{P} + \frac{Traffic_1}{B} + Delay_1$

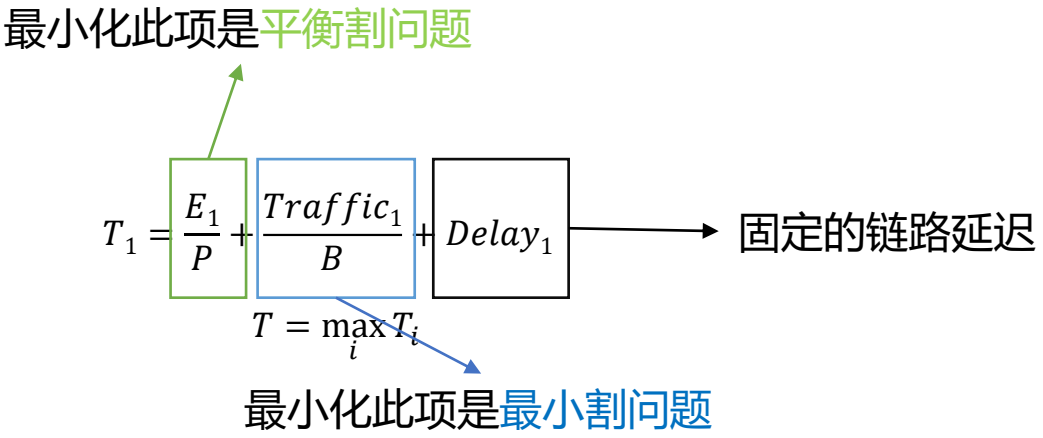
传输开销

整体仿真完成时间　$T = \max_i T_i$



寻找最优划分是一个最优化问题

- 约束条件（输入）：
  1. 网络拓扑；2. 路由方式；3. 流量需求矩阵；4.集群的计算/通信能力
- 输出:
  针对网络仿真任务的划分方案
- 目标函数:

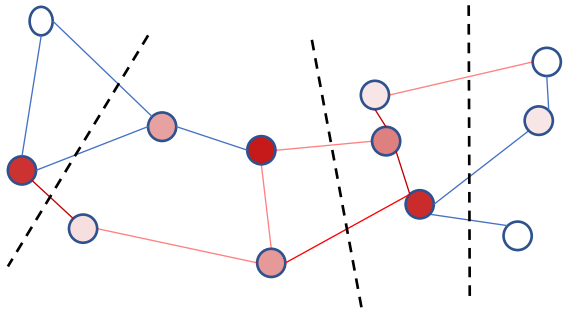


- 为了求解该问题，我们需要：

1. 运行一个简单的**流模拟**来获取E_i, $Traffic_i$, $Delay_i$, #Rounds等输入数据
2. 我们证明这是一个**NP-hard问题**，因此需要设计**启发式算法**

启发式划分算法

- 分层式划分算法：
  1. 使用最小平衡割算法将网络一分为二
  2. 迭代上述步骤，直到所有机器都被分配任务或进一步划分没有性能增益



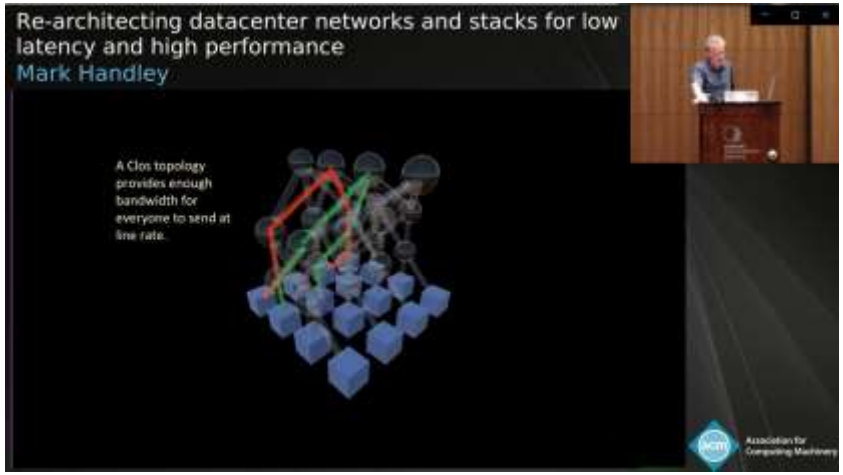
- 步骤一是一个经典的最小平衡割问题[1]，被证明是NP-hard问题，该问题有近似线性复杂度的求解算法[2]

[1] Chuzhoy J, Gao Y, Li J, et al. A deterministic algorithm for balanced cut with applications to dynamic connectivity, flows, and beyond[C]//2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS). IEEE, 2020: 1158-1167.

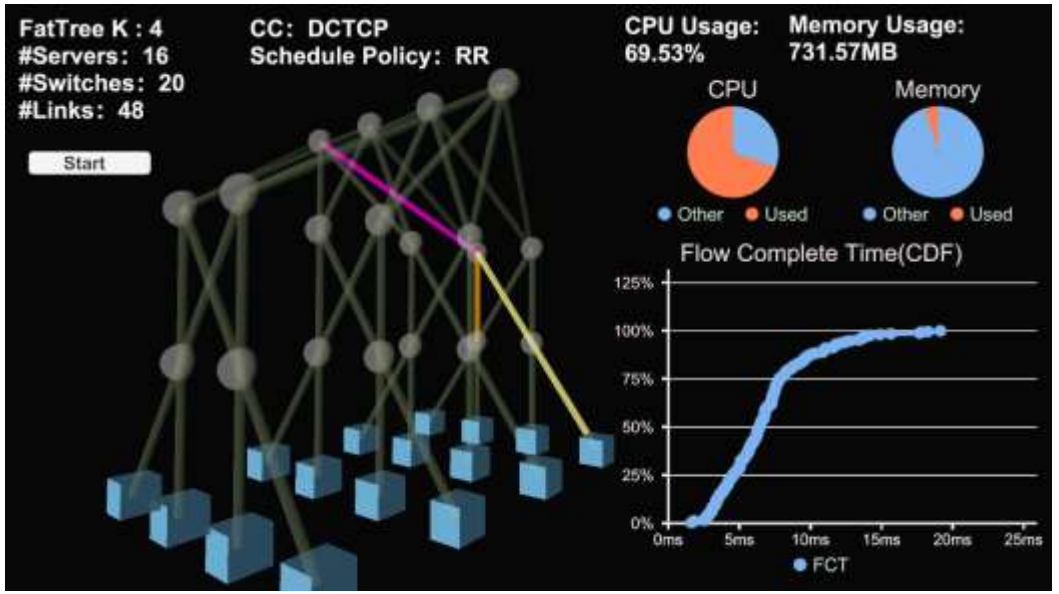
[2] https://arxiv.org/pdf/1910.08025.pdf

实现

- 受Mark Handley在SIGCOMM’ 17和SIGCOMM’ 19上的启发，我们也基于Unity实现了DONS
- ~3000代码，已开源[1]
- **发布DONS v0.1**，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

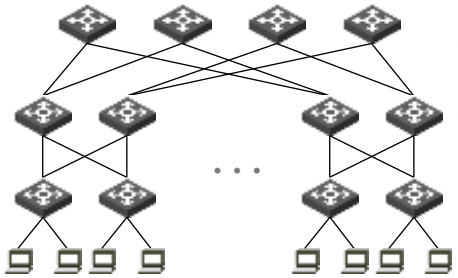


DONS前端

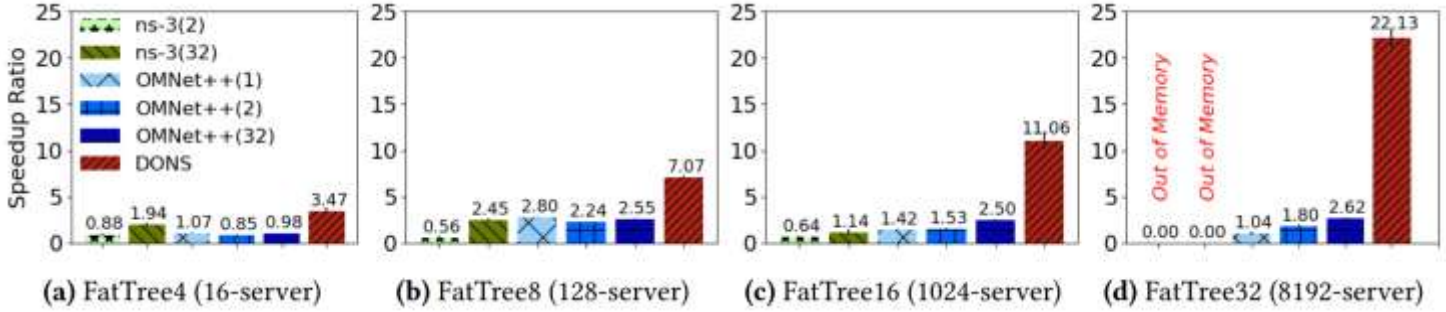
[1] https://github.com/dons2023/Data-Oriented-Network-Simulator

实验

仿真速度

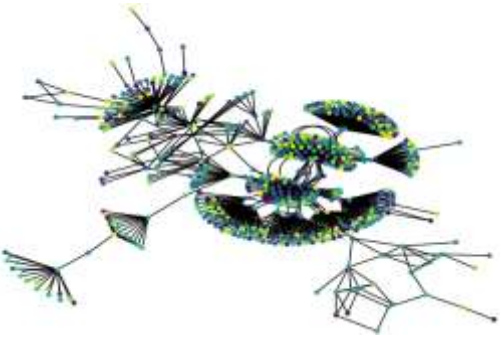


FatTree (K = 4, 8, 16, 32)



DONS相比于ns-3（单进程）加速了3倍 到 22倍

可扩展性



Large-scale WAN from a ISP

#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

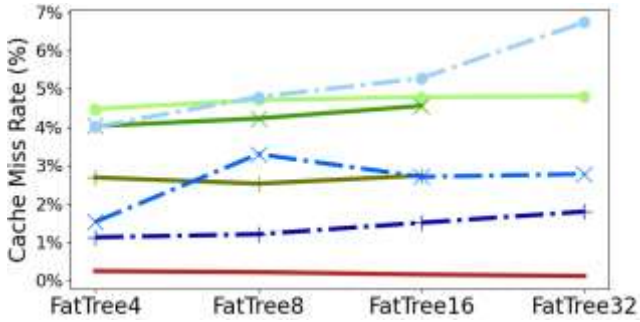
9.5 天 → 5 小时

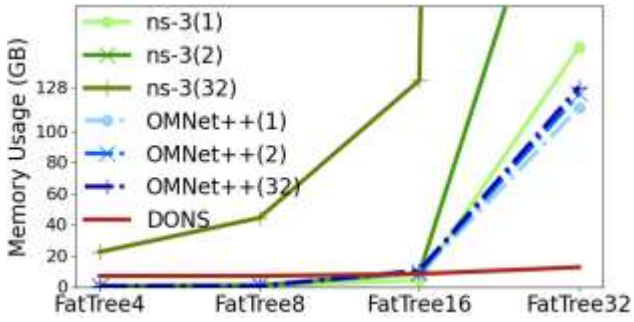
~8 天 → 3 小时

DONS相比于OMNeT++加速了42倍 到 65倍

实验

系统性能 & 开销



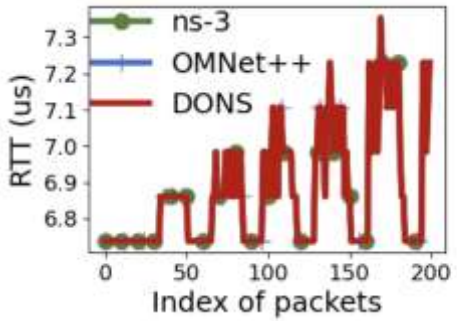


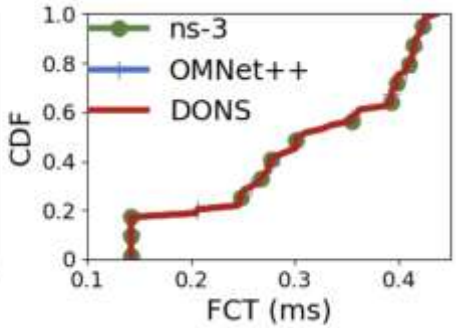
单机:
two Intel Xeon CPUs (totaling 32 cores),
Memory: 128 GB,
L3 cache: 20MB

DONS cache miss rate: 0.12%,
降低了4.5倍 to 56倍

ns-3 and OMNeT++ 单机最多支持FatTree32 (8k servers)
DONS 支持 FatTree48 (27k servers)

保真度

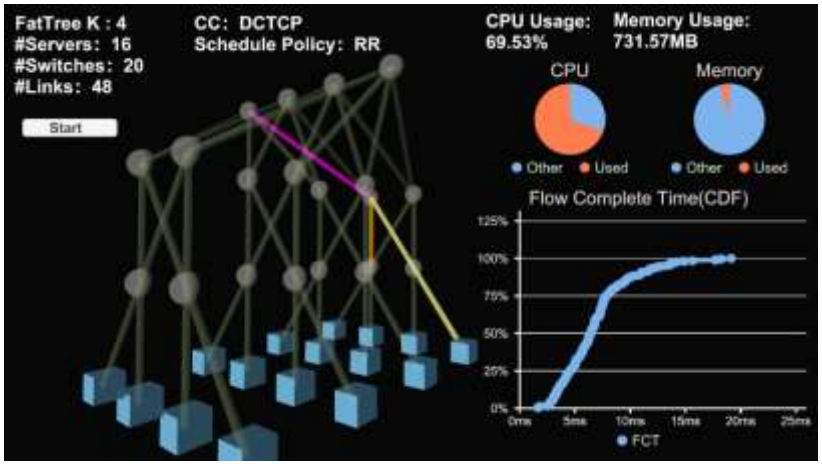




Link rate = 100Gbps
Link delay = 1us
Packet payload size=1,000B
Buffer size = 32MB
CCA = DCTCP

仿真结果与已有DES仿真器(ns-3 和 OMNeT++)的100%一致

总结



Data-Oriented Network Simulator (DONS)

高保真

- 离散事件仿真
- 正确性保证

大规模

- 支持分布式并行

高性能

- CPU缓存友好
- 内存友好
- 多核CPU并行效率高

低成本

- 仅用CPU就能快速完成仿真

- 已发布稳定版本v0.1：https://github.com/dons2023/Data-Oriented-Network-Simulator
- **欢迎大家使用！**

谢谢莅临！