中国科学技术大学
University of Science and Technology of China

Hedera: Dynamic Flow Scheduling for Data Center Networks

Mohammad Al-Fares, Sivasankar Radhakrishnan, et al.

NSDI 2010

授课教师：赵功名

中国科大计算机学院

2025年秋·高级计算机网络

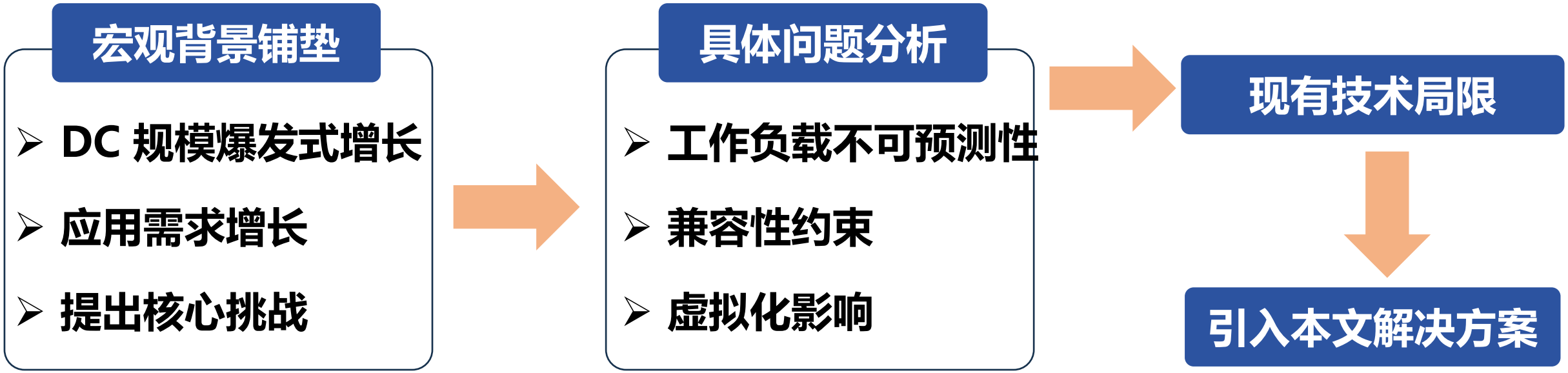
Outline

I. Introduction
II. Background
III. Architecture
IV. Estimation and Scheduling
V. Implementation
VI. Evaluation
VII. Review

Introduction

引言概述

本文的引言部分旨在阐述现代数据中心网络的背景、挑战以及现有方法的局限性，从而引出 Hedera 动态流调度系统的必要性。引言从数据中心的快速扩展和应用需求入手，逐步分析网络设计中的关键问题，并介绍Hedera的核心思想与贡献



Introduction：The core challenges of DC networks

数据中心的规模与带宽挑战

- 背景：近年来，大型组织正在构建支持数万台机器的庞大数据中心，同时许多计算、存储和操作正在迁移到云托管提供商。这导致应用（如科学计算、Web搜索和MapReduce）对**集群内（intra-cluster）**带宽的**需求急剧增加**
- 核心问题：随着数据中心和应用规模的扩大，网络结构必须支持潜在的 **All-to-All 通信**。数据中心拓扑通常采用多根树结构以克服交换机端口密度限制，但这种结构需要高效利用多条路径
- All-to-All 通信：是分布式计算中的一种核心通信模式。它要求每个计算节点将自己的数据分割后发送给所有其他节点，同时接收来自**所有**其他节点的数据片段。所有节点都在“互相交换数据”，最终每个节点获得来自所有节点的部分信息

Introduction：The core challenges of DC networks

云应用的三个特性使网络设计复杂化

- 工作负载不可预测：数据中心的工 作负载在时间和空间上可变，因此静态资源分配不足
- 兼容性要求：客户希望在商品操作系统上运行软件，网络必须在不修改软件或协议的情况下提供高带宽
- 虚拟化影响：虚拟化技术使应用难以保证运行在同一物理机架，导致传统拓扑中出现机架间网络瓶颈

这些特性强调网络需要**动态适应**，而非依赖静态设计

Introduction：The core challenges of DC networks

现有协议与拓扑的不匹配

- 传统协议局限：数据中心的路由和转发协议最初为企业环境设计，其中通信模式相对可预测，路径数量有限。而现代数据中心依赖路径多样性实现水平扩展，拓扑与典型企业网络截然不同
- 多根树（Multi-rooted Tree）拓扑的潜力与问题：由于商用交换机端口密度有限，数据中心拓扑常采用多根树结构，具有多条主机对路径。挑战是如何动态转发流以最小化链路过载，但现有协议（如ECMP）使用静态哈希，导致碰撞和带宽损失

Introduction：The core challenges of DC networks

ECMP的局限性

- 什么是 ECMP（等价多路径路由）：是一种网络路由技术，通过在开销值相同的多条路径上同时转发流量，实现负载均衡与带宽提升
- ECMP问题：ECMP通过**流哈希静态映射**路径，但忽略当前网络利用率和流大小，导致碰撞淹没交换机缓冲区，降低利用率。例如，多个大流可能哈希到同一端口，造成可避免的瓶颈
- Hedera的提出：本文提出Hedera，一个动态流调度系统，通过收集流信息、计算无冲突路径并指导交换机重路由，以最大化聚合网络利用率（**二分带宽**）

  Hedera采用全局视角，能识别交换机本地调度器无法看到的瓶颈

Outline

I. Introduction
II. **Background**
III. Architecture
IV. Estimation and Scheduling
V. Implementation
VI. Evaluation
VII. Review

Background

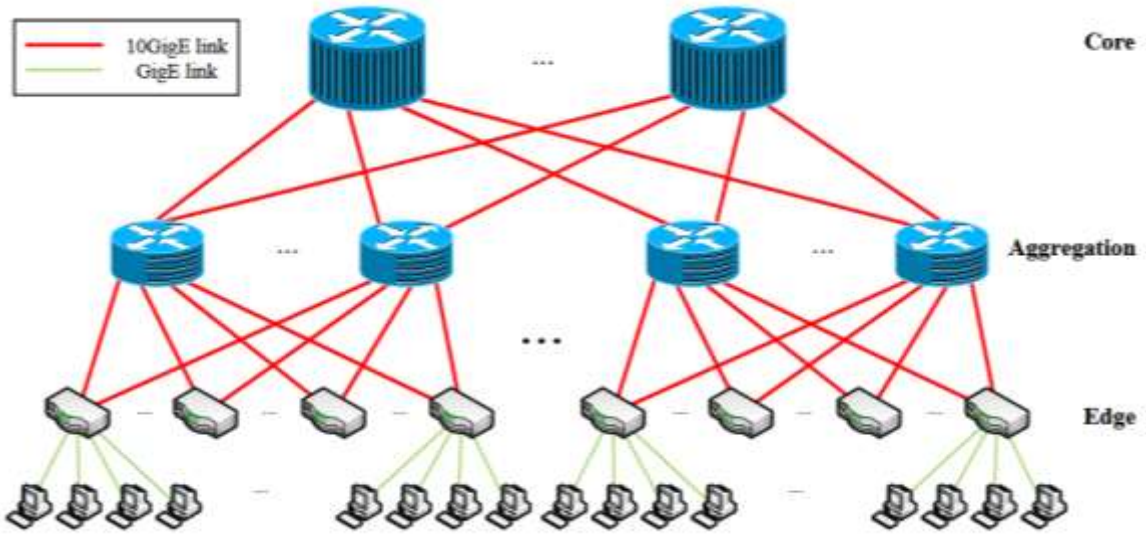
分布式计算的发展

- 分布式计算框架的兴起
  - 主要技术推动力：MapReduce、Hadoop、Dryad等分布式计算框架的出现
  - Web服务需求：搜索引擎、电子商务、社交网络等大型网络服务的发展
  - 基础设施变革：推动由商用PC组成的大规模计算集群的建设
- 数据规模的爆炸式增长
  - 数据量级：达到PB级别的大型数据集
  - 存储规模：分布在数万台机器上的复杂数据存储系统
  - 处理需求：对大规模数据处理能力提出新的挑战

Background

网络性能瓶颈

- 现实情况：网络成为新的性能瓶颈。集群应用性能**受限于网络**而非本地资源。改善应用性能的关键在于提升网络性能，这对传统数据中心网络架构提出了新挑战
- 拓扑结构演进：传统分层树状结构使用小型廉价边缘交换机，通过2-3层交换机互联克服端口密度限制。新一代数据中心采用水平扩展方案，利用**多根树拓扑**（右图）提供并行路径替代昂贵核心路由器



Background: 2.1 Data Center Traffic Patterns

数据可获得性问题

- 由于隐私和安全考虑，目前没有公开的数据中心流量追踪数据可供研究使用。
- 这一数据缺失给数据中心网络研究带来了显著挑战，使得研究人员难以基于真实流量模式进行算法验证和性能评估

替代研究方法

- 为克服数据可获得性限制，本文章采用了两种互补的方法：
  - 基于已发表工作中的流量分布模式生成典型数据中心工作负载
  - 创建能够充分测试网络极限的合成通信模式

Background: 2.2 Current Data Center Multipathing

ECMP（等价多路径）： DC 中利用多路径的主流技术

- 路径配置：交换机为每个子网配置多条等成本转发路径
- 哈希决策：基于包头字段哈希值对路径数取模，确定转发路径
- 流保持性：同一流的所有数据包走相同路径，维持报文顺序
- TCP友好性：避免报文乱序（TCP将乱序解读为拥塞导致丢包）

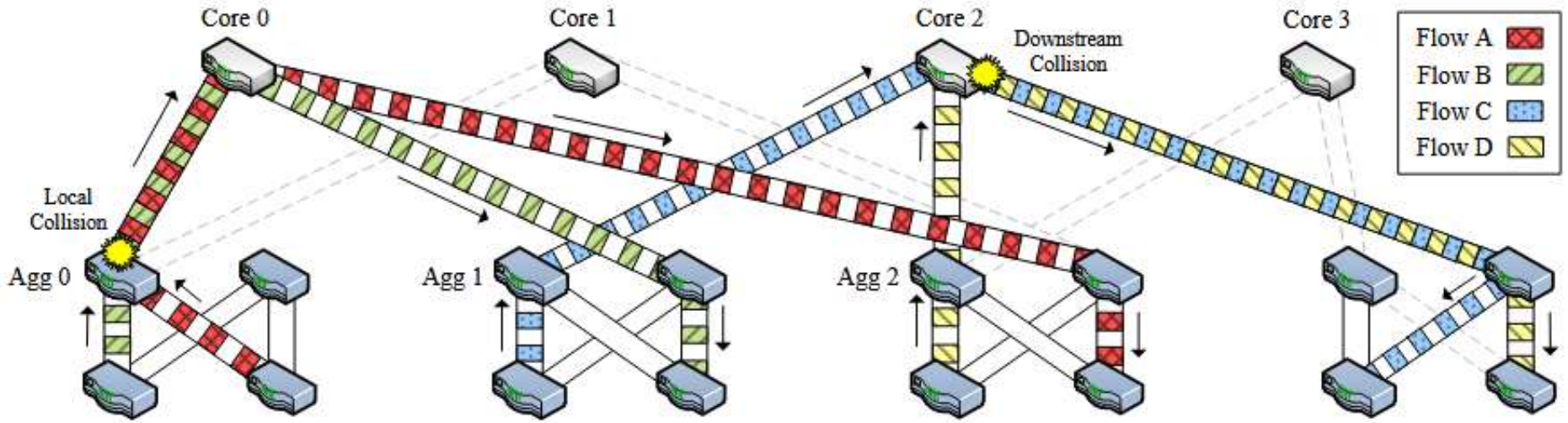
VLB（Valiant负载均衡）：ECMP的密切关联技术，核心特征包括

- 网格网络负载均衡：通过随机选择中间核心交换机实现负载均匀分布
- 每流随机化：现代VLB实现在每流而非每包基础上进行随机转发
- 等效性：每流VLB实质上与ECMP等效

Background: 2.2 Current Data Center Multipathing

ECMP的核心局限性：哈希碰撞问题

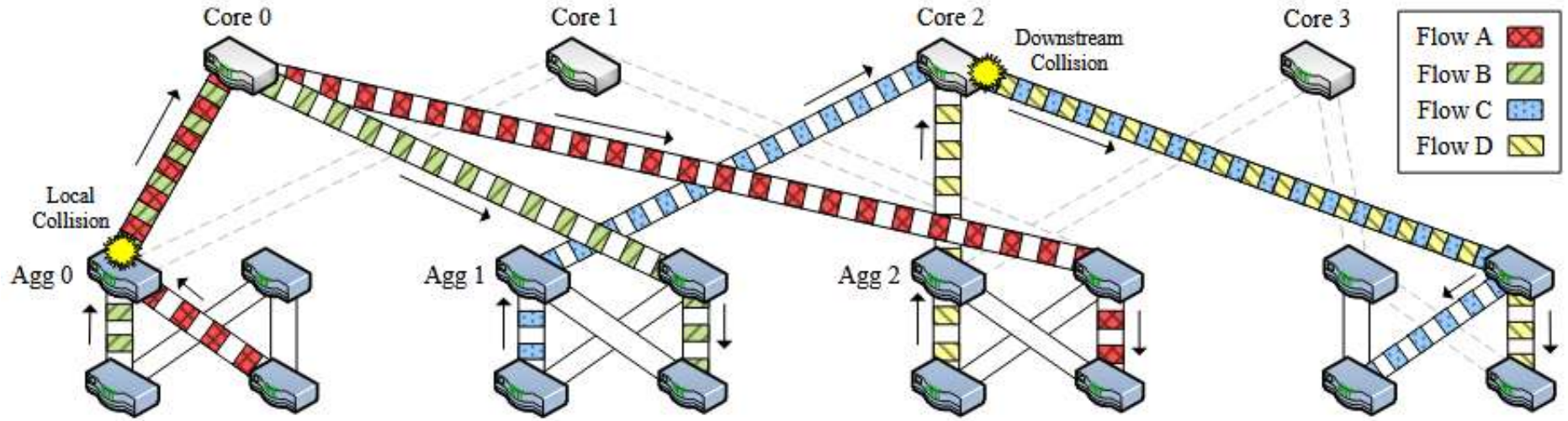
- ECMP的主要限制在于：两个或多个大流、长流可能发生哈希碰撞，最终映射到同一输出端口，造成可避免的瓶颈



Background: 2.2 Current Data Center Multipathing

ECMP的核心局限性：哈希碰撞问题

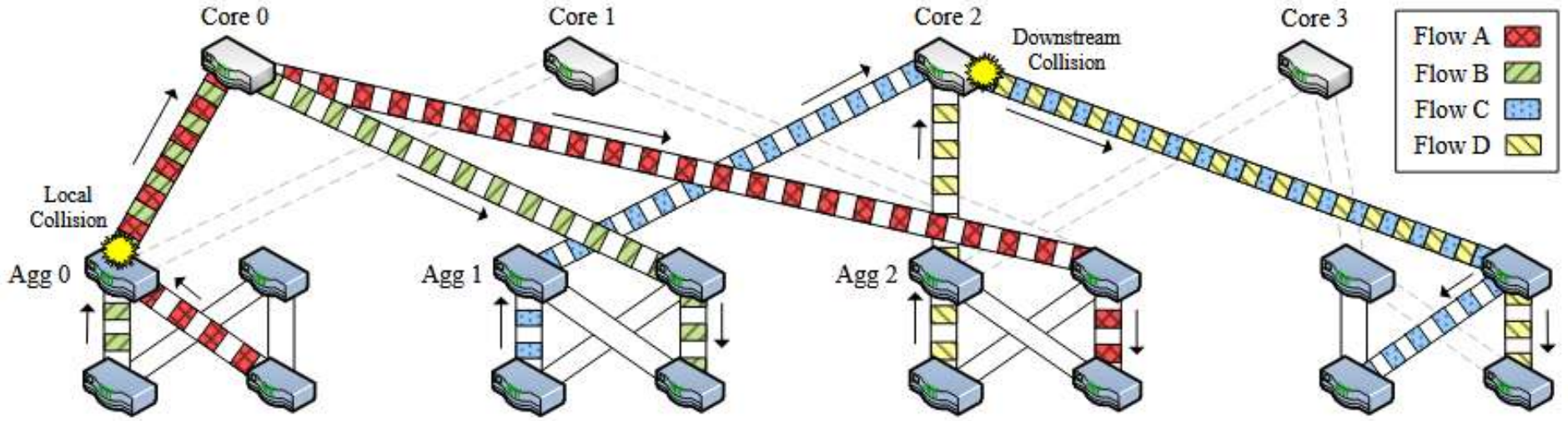
- 在 1Gbps 多根树拓扑中，哈希导致两处碰撞：
  - 本地干扰碰撞：TCP流A和B在Agg0交换机处发生哈希碰撞。结果两流共享带宽，各受限至500Mbps
  - 下游干扰碰撞：Agg1和Agg2独立转发报文，无法预见到Core2处流C和D的碰撞，同样导致50%的二分带宽损失



Background: 2.2 Current Data Center Multipathing

ECMP的核心局限性：哈希碰撞问题

- 在上述示例中，通过改进转发策略可实现性能优化
  - 流A可转发至Core1
  - 流D可转发至Core3
  - 所有四个 TCP 流**本可都达到** 1Gbps 容量，实际性能因碰撞而折半



Background: 2.2 Current Data Center Multipathing

关键影响因素

- ECMP 和基于流的 VLB 性能本质上取决于：
  - 流大小：大流对碰撞更敏感
  - 每主机流数量：流数量影响哈希分布均匀性
- 在以下情况下哈希转发表现良好：
  - All-to-All 通信：主机间同时进行全对全通信
  - 短流场景：仅持续几个 RTT 的个体流
  - 高并发流：每主机存在大量同时流时性能影响较小
- 涉及大块数据传输的非均匀通信模式需要更**精细的流调度**，以避免网络瓶颈
- 作者将完整的权衡评估推迟至第6章节

Background: 2.2 Current Data Center Multipathing

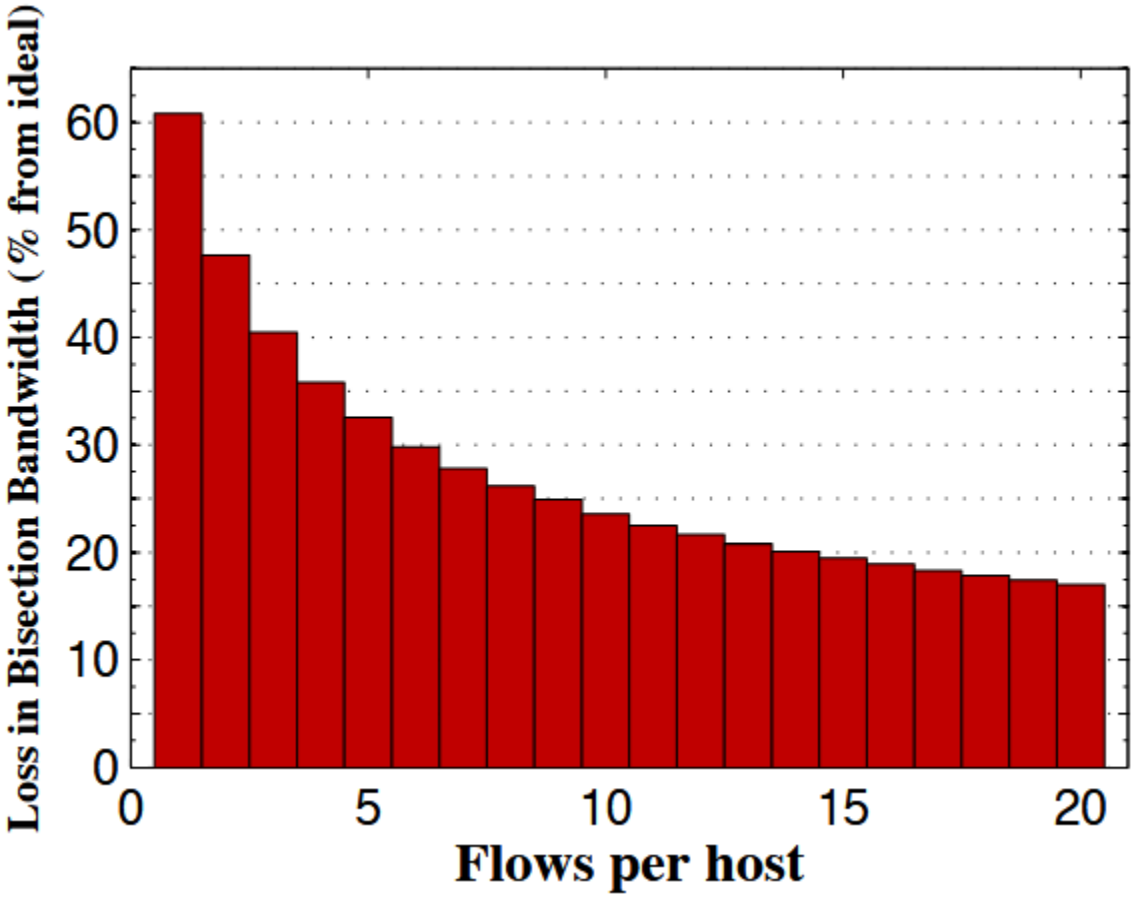
哈希机制导致的性能下降问题

- 通过**蒙特卡洛模拟**方法直观展示了哈希机制导致的性能下降问题
- 模拟环境配置
  - 采用三层胖树拓扑结构进行模拟实验：
  - 网络设备：1GigE 48 端口交换机
  - 主机规模：27,000台主机参与数据洗牌操作
    - 回忆：你能否计算出 k=48 时Fat-Tree 拓扑中主机的最大数量？
  - 链路限制：每条链路容量限制为1GigE
  - 路由机制：基于哈希的路径分配策略

Background: 2.2 Current Data Center Multipathing

哈希机制导致的性能下降问题

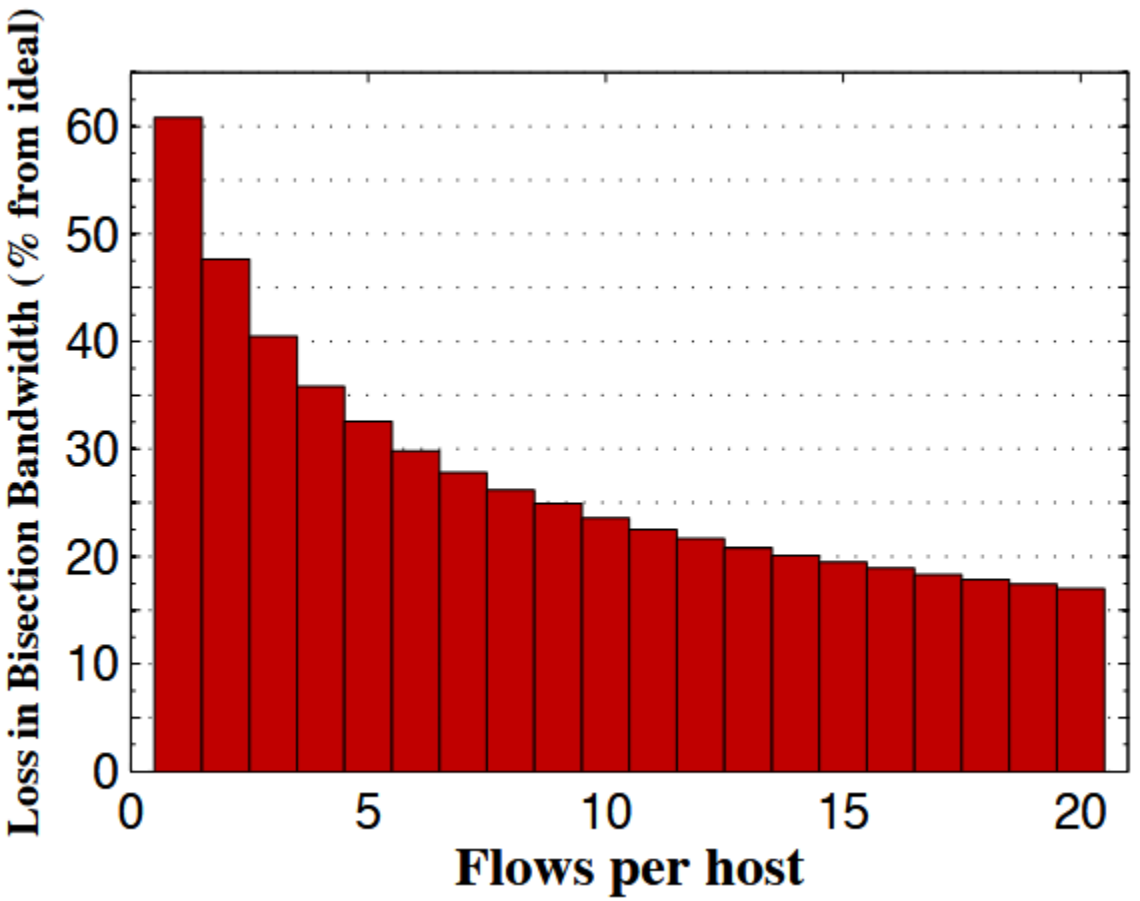
- 顺序传输场景
  - 当每台主机依次向所有远程主机传输等量数据时
  - 性能损失：哈希碰撞导致网络二分带宽平均降低60.8%（右图）
  - 根本原因：少量大流哈希碰撞造成严重带宽浪费



Background: 2.2 Current Data Center Multipathing

哈希机制导致的性能下降问题

- 并行传输场景
  - 当每台主机通过1,000个并行流与远程主机通信时：
  - 性能影响：哈希碰撞仅使总二分带宽降低2.5%
  - 优势原因：大量小流平均分配带宽需求



Background: 2.2 Current Data Center Multipathing

哈希机制导致的性能下降问题

- 性能差异的核心在于并发流数量对哈希碰撞成本的影响：
  - 高并发流优势机制
  - 个体流速率：大量并发流导致单个流速率较小
  - 碰撞成本：碰撞不会造成显著性能损失
  - 链路容量利用：每条链路有1,000个"槽位"需要填充
  - 性能阈值：仅当远超1,000个流哈希到同一链路时性能才会明显下降
- 流量模式与性能关系
  - 大流模式：顺序传输 → 高碰撞概率 → 严重性能损失
  - 小流模式：并行传输 → 低碰撞影响 → 轻微性能影响

Background: 2.2 Current Data Center Multipathing

Hedera的设计定位

- Hedera并非取代 ECMP，**而是作为补充方案**：
  - 默认行为保留：维持 ECMP 的正常转发机制
  - 问题场景优化：针对导致 ECMP 问题的通信模式提供增强解决方案
- 应用场景定位
  - 适用场景：处理引发 ECMP 问题的特定通信模式
  - 价值体现：在哈希碰撞敏感场景下提供性能保障
  - 设计理念：与现有 ECMP 基础设施保持兼容性和协同性
- 上述分析揭示了哈希转发机制的性能特征及其对流量模式的敏感性，同时明确了 Hedera 是一个在 DC 网络架构中的**动态流调度方案**

Background: 2.3 Dynamic Flow Demand Estimation

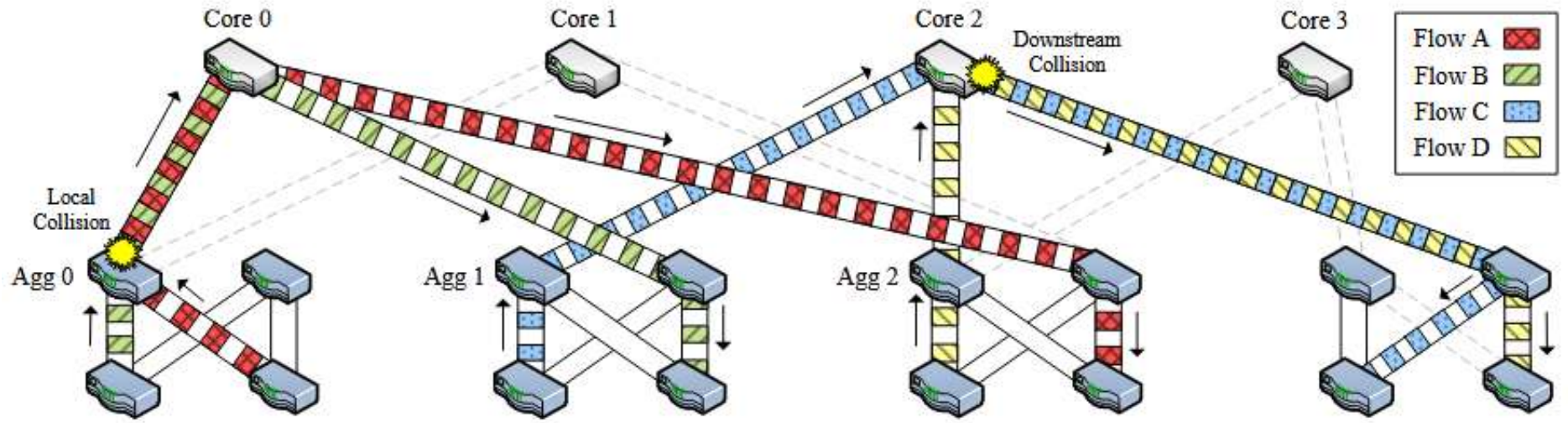
动态网络调度的核心挑战

- 动态网络调度机制的核心目标：寻找最优的网络全局调度方案
  - 最直接的解决方法是：
  - 监测链路利用率：实时测量网络中所有链路的利用情况
  - 流量重分配策略：将流量从高利用率链路迁移至低利用率链路
- 流量迁移关键决策难题
- 核心问题是选择哪些流量进行迁移。直观的解决方案是：
  - 测量受限链路上每个流消耗的带宽
  - 将流量迁移至具有足够容量的替代路径
- 不幸的是（**关键发现**）：流的当前带宽可能无法反映其真实需求。在非最优调度情况下，网络受限的流可能只能获得远低于其潜在需求的带宽

Background: 2.3 Dynamic Flow Demand Estimation

动态网络调度的核心挑战

- **关键发现**：流的当前带宽可能无法反映其真实需求。在非最优调度情况下，网络受限的流可能只能获得远低于其潜在需求的带宽
- 例如，在图2（下图）中，所有流都以500 Mbps的速度进行通信，尽管所有流都可以以1Gbps的速度进行通信并具有更好的转发



Background: 2.3 Dynamic Flow Demand Estimation

自然需求的概念定义

- 定义 TCP 流的**自然需求（natural demand）**
  - 在一个完全无阻塞的网络中，该流能够增长到的最大速率，最终仅受限于发送方或接收方网卡速度
- 文章定义的自然需求概念包含三个关键要素
  - 环境假设：基于完全无阻塞网络的理想条件
  - 增长评估：衡量流在最优条件下的速率增长能力
  - 限制因素：发送端或接收端的能力，而不是网络的能力
- 准确估计自然需求，使调度器能够：
  - 识别具有高潜在需求的流
  - 优先为这些流分配优质路径资源
  - 避免因当前带宽误导而做出次优调度决策
- 4.2 节将详细介绍如何高效估计流的自然需求

Outline

I. Introduction
II. Background
III. Architecture
IV. Estimation and Scheduling
V. Implementation
VI. Evaluation
VII. Review

Architecture: 3.0 Overview

Architecture Section Overview

- Hedera 采用三步骤动态调度机制：
  - 大流检测：在边缘交换机识别高带宽流
  - 需求估计与路径计算：评估自然需求并算法优化路径
  - 路径部署：将最佳路径安装到交换机
- 拓扑兼容性
  - 系统支持通用多根树拓扑，确保广泛适用性
- 实现验证
  - 通过胖树拓扑的物理实现验证实用性，平衡复杂度与性能

Architecture: 3.1 Switch Initialization

Hedera 路径多样性利用与流调度机制

- 路径多样性核心原则
  - 为了充分利用多根树拓扑的路径多样性，系统必须将任意主机的出入流量尽可能**均匀地分布**到所有核心交换机。文章通过上行路径的非确定性选择和下行路径的确定性路由实现
- 路径选择机制
  - 上行路径（**非确定性**）：数据包在到达核心交换机前，路径动态选择，以利用多路径优势
  - 下行路径（**确定性**）：从核心交换机到目的边缘交换机的路径固定，确保最小成本路由。每个核心交换机到任何目的主机只有一条活跃最小成本路径

Architecture: 3.1 Switch Initialization

Hedera 路径多样性利用与流调度机制

- 网络初始化配置：为强制下行路径的确定性，系统初始化时：
  - 核心交换机：配置目的pod的IP地址范围前缀（pod指从核心交换机向下的子分组，在胖树测试床中为聚合和边缘交换机的完全二分图）
  - 聚合交换机：配置该pod内边缘交换机的下行端口前缀
  - 边缘交换机：直接转发包到连接的主机
- 默认流处理与动态调度
  - 新流默认行为：基于流的10元组哈希选择等成本路径（类似ECMP）
  - 阈值触发动态调度：当流速率超过阈值时，Hedera动态计算最优路径
  - 流定义：流由相同10元组的包流组成，确保一致性

<src MAC, dst MAC, src IP, dst IP, EtherType, IP protocol,

TCP src port, dst port, VLAN tag, input port>

Architecture: 3.2 Scheduler Design

Hedera 调度器设计

- Hedera 采用中心化调度器，基于全局通信需求的定期更新，动态控制边缘和聚合交换机的转发表。调度器核心目标是为流分配无冲突路径，即避免将多个流分配到无法承载其合计自然带宽需求的链路上
- 触发条件：当流持续一段时间且带宽需求超过阈值时，调度器使用算法分配路径
- 路径安装：根据所选路，在源边缘和聚合交换机中插入流表项，将流重定向
- 状态管理：流表项在流结束后超时失效。调度器维护的软状态无需与副本同步，不影响连接正确性，仅用于性能优化
- 算法开放性：支持多种调度算法，本文对比全局首次适应（Global First Fit）和模拟退火（Simulated Annealing）与ECMP的性能
- 优化目标：两种算法均通过搜索流到核心交换机的映射，提升当前通信模式下的聚合二分带宽，作为ECMP对大流转发的补充

Outline

I. Introduction
II. Background
III. Architecture
IV. Estimation and Scheduling
V. Implementation
VI. Evaluation
VII. Review

多商品流问题与流分类理论

- 多商品流问题：在通用网络中寻找流路由且不超出任何链路容量的问题被称为**多商品流问题**（Multicommodity Flow）
  - 其核心目标是在一个由节点和边构成的共享网络中，同时优化多种商品（或物流）的传输路径与流量分配
  - 该问题在整数流情况下被证明是 **NP 完全问题**
- 由于Hedera不针对特定拓扑进行优化，本文提出适用于多种现实数据中心拓扑的实用启发式算法，在计算复杂性和性能之间取得平衡

Estimation and Scheduling: 4.1 Host- vs. Network-Limited Flows

多商品流问题与流分类理论

- 流量可分为两类基本类型：网络受限流与主机受限流
- 网络受限流
  - 特征：使用分配路径上的所有可用带宽（如RAM数据传输）
  - 限制因素：受网络拥塞限制，而非主机网卡限制
  - 调度影响：在非最优调度下可能无法达到底层拓扑提供的最大可能带宽
- 主机受限流
  - 特征：理论最大吞吐量受源主机和目的主机中"较慢者"限制（如磁盘访问、处理能力等限制）
  - 限制因素：主机内部瓶颈而非网络资源

自然带宽需求估计

- TCP流的当前发送速率无法反映其在理想无阻塞网络中的自然带宽需求。为了做出智能的流放置决策，需要了解流在仅受限于发送方或接收方网卡时的带宽分配。**因此**本文设计了自然带宽需求估计算法
- 迭代收敛过程算法通过重复迭代实现需求收敛：
  - 增加阶段：从源端增加流容量
  - 减少阶段：在接收端减少超出容量
  - 收敛判断：每次迭代中，一个或多个流收敛，直至所有流达到自然需求

Estimation and Scheduling: 4.2 Demand Estimation

```
ESTIMATE-DEMANDS()
1   for all i, j
2       M_{i,j} ← 0
3   do
4       foreach h ∈ H do EST-SRC(h)
5       foreach h ∈ H do EST-DST(h)
6   while some M_{i,j}.demand changed
7   return M

EST-SRC(src: host)
1   d_F ← 0
2   n_U ← 0
3   foreach f ∈ ⟨src → dst⟩ do
4       if f.converged then
5           d_F ← d_F + f.demand
6       else
7           n_U ← n_U + 1
8   e_S ← (1.0 − d_F) / n_U
9   foreach f ∈ ⟨src → dst⟩ and not f.converged do
10      M_{f.src,f.dst}.demand ← e_S
```

```
EST-DST(dst: host)
1   d_T, d_S, n_R ← 0
2   foreach f ∈ ⟨src → dst⟩
3       f.rl ← true
4       d_T ← d_T + f.demand
5       n_R ← n_R + 1
6   if d_T ≤ 1.0 then
7       return
8   e_S ← 1.0 / n_R
9   do
10      n_R ← 0
11      foreach f ∈ ⟨src → dst⟩ and f.rl do
12          if f.demand < e_S then
13              d_S ← d_S + f.demand
14              f.rl ← false
15          else
16              n_R ← n_R + 1
17      e_S ← (1.0 − d_S) / n_R
18  while some f.rl was set to false
19  foreach f ∈ ⟨src → dst⟩ and f.rl do
20      M_{f.src,f.dst}.demand ← e_S
21      M_{f.src,f.dst}.converged ← true
```

Estimation and Scheduling: 4.2 Demand Estimation

```
ESTIMATE-DEMANDS()
1    for all i, j
2        M_{i,j} ← 0
3    do
4        foreach h ∈ H do EST-SRC(h)
5        foreach h ∈ H do EST-DST(h)
6    while some M_{i,j}.demand changed
7    return M
```

这个算法的核心思想是进行一种迭代式的、模拟的带宽分配

- 迭代过程：重复执行两个核心步骤，直到所有流的需求不再变化（收敛）
  - 从**发送方**角度分配：每个发送主机尝试将它所有的出向流的需求总和增加到1（即其网卡的全部带宽）。它会将剩余的带宽平均分配给所有尚未收敛的流
  - 从**接收方**角度修正：每个接收主机检查所有入向流的需求总和。如果总和超过1（即其网卡带宽），它就需要“削减”，降低某些流的需求，直到总和不超过1
  - 这个过程会持续进行，直到所有流的需求都稳定下来，不再变化
- 接下来分别简单介绍如何从发方和收方操作

Estimation and Scheduling: 4.2 Demand Estimation

```
EST-SRC(src: host)
1    d_F ← 0
2    n_U ← 0
3    foreach f ∈ ⟨src → dst⟩ do
4        if f.converged then
5            d_F ← d_F + f.demand
6        else
7            n_U ← n_U + 1
8    e_S ← (1.0 − d_F) / n_U
9    foreach f ∈ ⟨src → dst⟩ and not f.converged do
10       M_{f.src,f.dst}.demand ← e_S
```

发送方“灌水”

- 目标：计算从这个 src 发出的所有流的需求
- 计算每个未收敛流可以分到的带宽： (1.0 - d_F) / n_U。这里的 1.0 代表源主机1个单位的出口带宽（例如1Gbps）
- （左图）行9-10：将计算出的平均带宽 e_S 赋给每一个未收敛的流
- 简单比喻：一个水管（源主机）有1升/秒的流量，要分给几个小管子（流）。它先看看哪些小管子已经装满了（收敛的流），把它们的流量减掉，然后把剩下的水量平均分给还没满的小管子

Estimation and Scheduling: 4.2 Demand Estimation

```
EST-DST(dst: host)
1   d_T, d_S, n_R ← 0
2   foreach f ∈ ⟨src → dst⟩
3       f.rl ← true
4       d_T ← d_T + f.demand
5       n_R ← n_R + 1
6   if d_T ≤ 1.0 then
7       return
8   e_S ← 1.0 / n_R
9   do
10      n_R ← 0
11      foreach f ∈ ⟨src → dst⟩ and f.rl do
12          if f.demand < e_S then
13              d_S ← d_S + f.demand
14              f.rl ← false
15          else
16              n_R ← n_R + 1
17      e_S ← (1.0 − d_S) / n_R
18  while some f.rl was set to false
19  foreach f ∈ ⟨src → dst⟩ and f.rl do
20      M_{f.src,f.dst}.demand ← e_S
21      M_{f.src,f.dst}.converged ← true
```

EST-DST(dst: host) - 接收方“校准”

- 目标：确保发送到这个 dst 的所有流的需求总和不超过其接收能力（≤1.0）
- 行6-7：如果总需求 d_T已经 ≤ 1，说明接收方完全能承受，无需调整，直接返回
- 行8-18：如果总需求 > 1.0，则进入一个循环进行削减
- 简单比喻：一个水桶（目的主机）只有1升的容量，但好几个水管（流）都要往里面灌水，总水量超标。水桶就开始调整：先保护那些本来就流量很小的水管（它们的需求被保留），然后把超出的水量让那些大流平均分摊削减，直到总入水量不超过水桶容量。这些被削减的大流就被标记为“已调整好”

Estimation and Scheduling: 4.3 Global First Fit

全局首次适应算法

- 需求估计提供“需求洞察”，而我们的最终目的是执行“路径分配”，一个简单的路径分配方法是——全局首次适应算法
- 算法基础原理：在多根树拓扑中，当检测到大流时，调度器启动路径搜索机制。算法**核心是线性遍历所有等价路径**，寻找能完全容纳该流量的可用路径

```
GLOBAL-FIRST-FIT(f: flow)
1    if f.assigned then
2        return old path assignment for f
3    foreach p ∈ P_src→dst do
4        if p.used + f.rate < p.capacity then
5            p.used ← p.used + f.rate
6            return p
7    else
8        h = HASH(f)
9        return p = P_src→dst(h)
```

- 核心工作流程（右图）
  - 路径搜索(4)：顺序检查各路径的链路容量
  - 资源分配(5)：找到合适路径后执行容量预留
  - 流表安装(6)：在对应交换机创建转发规则
  - 状态维护：持续跟踪链路预留状态，流结束时释放资源
- 算法特性分析
  - 策略类型：贪心算法。不保证全局最优，但实践表现良好
  - 负载适应性：轻负载时效率高，重负载时选择受限

Estimation and Scheduling: 4.4 Simulated Annealing

模拟退火算法

- 为什么已有全局首次适应算法，还要引入一个相同功能的模拟退火算法?
  - 前面讲的全局首次适应的优势在于其简单性和低开销（时间复杂度独立于流数量），但它无法处理高负载场景，因此本文引入了模拟退火算法
  - 独特优势：与全局首次适应算法相比，模拟退火在搜索策略上更加智能，能够避免局部最优解，从而在复杂网络环境下实现更高的二分带宽
- 算法定位：模拟退火调度器是 Hedera 系统中用于动态流路径计算的核心算法
- 思想概括：通过概率搜索方式高效计算流路径

Estimation and Scheduling: 4.4 Simulated Annealing

模拟退火算法

- 关键创新：算法的核心创新在于显著减少搜索空间
  - 传统方法：为每个流单独分配核心交换机，导致搜索空间过大
  - 本文采用：为每个目的主机分配一个核心交换机的策略，将所有发往该主机的流通过同一核心交换机转发
  - 这一优化将搜索空间的直径减少到流数量与主机数量的最小值，极大提升了算法效率
- 虽然模拟退火本身**是已知的通用优化算法**，但 Hedera 论文的创新性体现在针对数据中心网络调度的特定问题进行了关键性优化和重构（搜索空间重构创新 +领域特定优化 + 工程实效验证）

Estimation and Scheduling: 4.4 Simulated Annealing

模拟退火算法 Pseudocode

- 核心工作流程（右图）
  - 初始化：设置初始温度 T 和初始状态 s
  - 迭代搜索：在温度降为零前重复执行
    - 生成邻居状态 s_n
    - 计算能量差 ΔE = E - E_n
    - 根据接受概率决定是否转移状态
    - 降低温度
  - 终止条件：温度降为零，返回最优解算法特性分析

```
SIMULATED-ANNEALING(n : iteration count)
1    s ← INIT-STATE()
2    e ← E(s)
3    s_B ← s, e_B ← e
4    T_0 ← n
5    for T ← T_0 ... 0 do
6        s_N ← NEIGHBOR(s)
7        e_N ← E(s_N)
8        if e_N < e_B then
9            s_B ← s_N, e_B ← e_N
10       if P(e, e_N, T) > RAND() then
11           s ← s_N, e ← e_N
12   return s_B
```

Estimation and Scheduling: 4.5 Comparison of Placement Algorithms

模拟退火算法 vs. 全局首次适应

- 与全局首次适应对比
  - 即时性：全局首次适应**更快**，可立即重路由，模拟退火等待调度周期
  - 最优性：模拟退火通过概率搜索获得**更优**解
  - 计算开销：模拟退火需要更多计算，但收益显著
- 实际部署考虑
  - 算法简单性使**模拟退火算法**适合硬件实现（如FPGA）
  - 算术操作为主，便于优化
  - 在大规模数据中心中，额外计算**开销被**带宽收益**抵消**
- 总结：概念上比全局首次适应复杂，但其带来的带宽增益证明**这种复杂性是合理的**，特别是考虑到现代数据中心网络基础设施的巨大投资

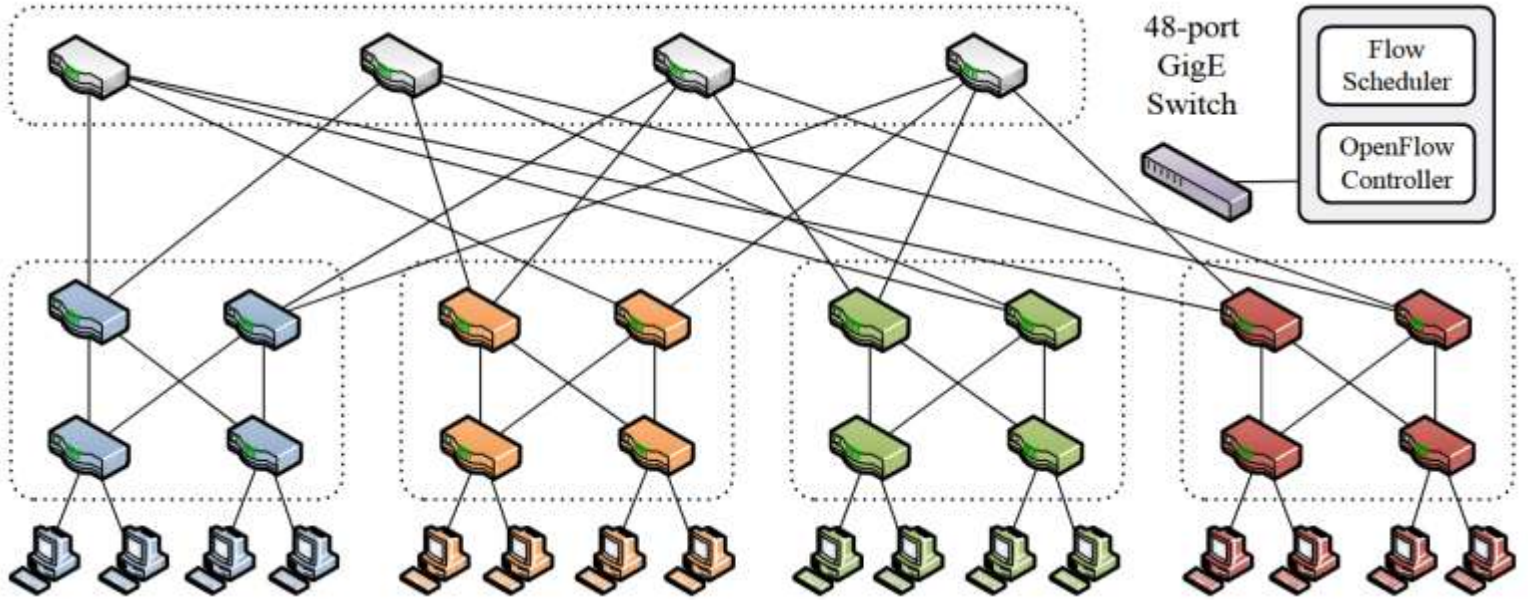
Outline

I. Introduction
II. Background
III. Architecture
IV. Estimation and Scheduling
V. **Implementation**
VI. Evaluation
VII. Review

Implementation: 5.1 Topology

实验部署与测试床设计

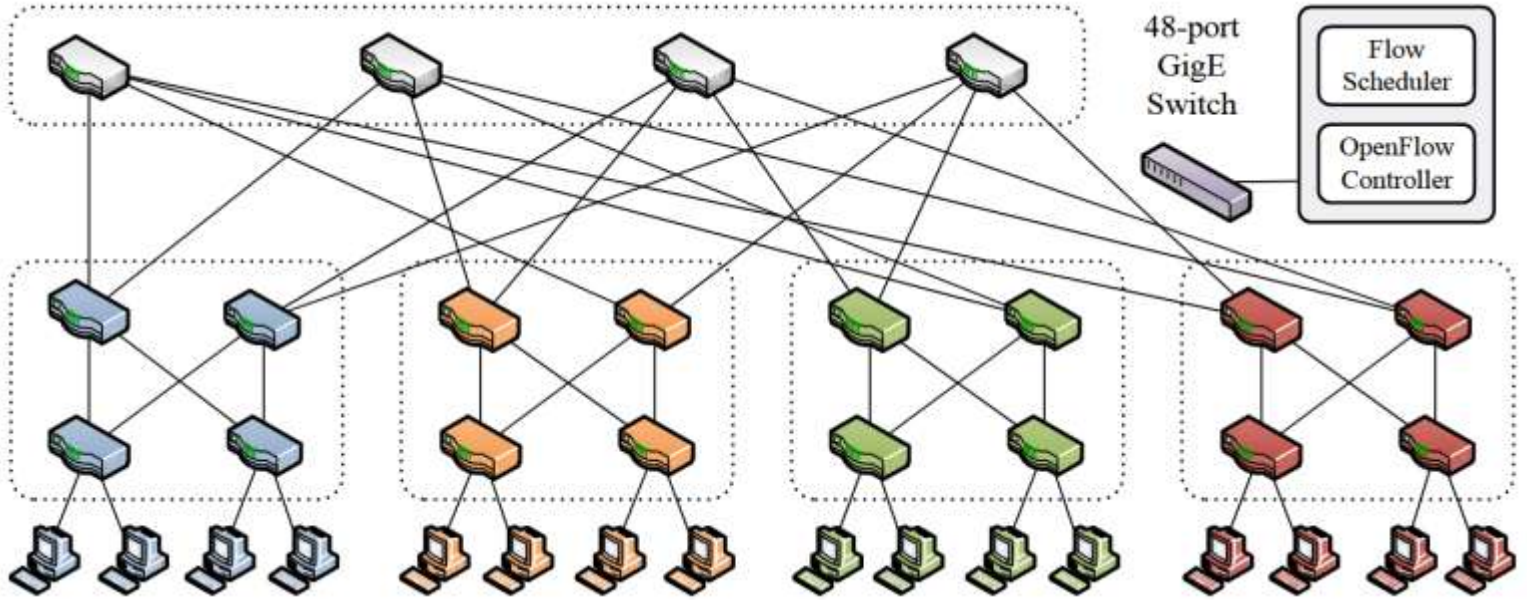
- Hedera采用**物理**测试床与**模拟器**相结合的方式验证系统性能：
  - 物理测试床：构建真实的多根树网络，验证算法在实际环境中的表现
  - 大规模模拟器：模拟超大规模网络环境，评估算法扩展性
- 拓扑：Fat-Tree
- Testbed
  - 6台主机
  - 网络设备：20台4端口交换机构建fat-tree拓扑
  - 控制网络：额外部署48端口非阻塞千兆以太网交换机作为控制平面



Implementation: 5.1 Topology

实验部署与测试床设计

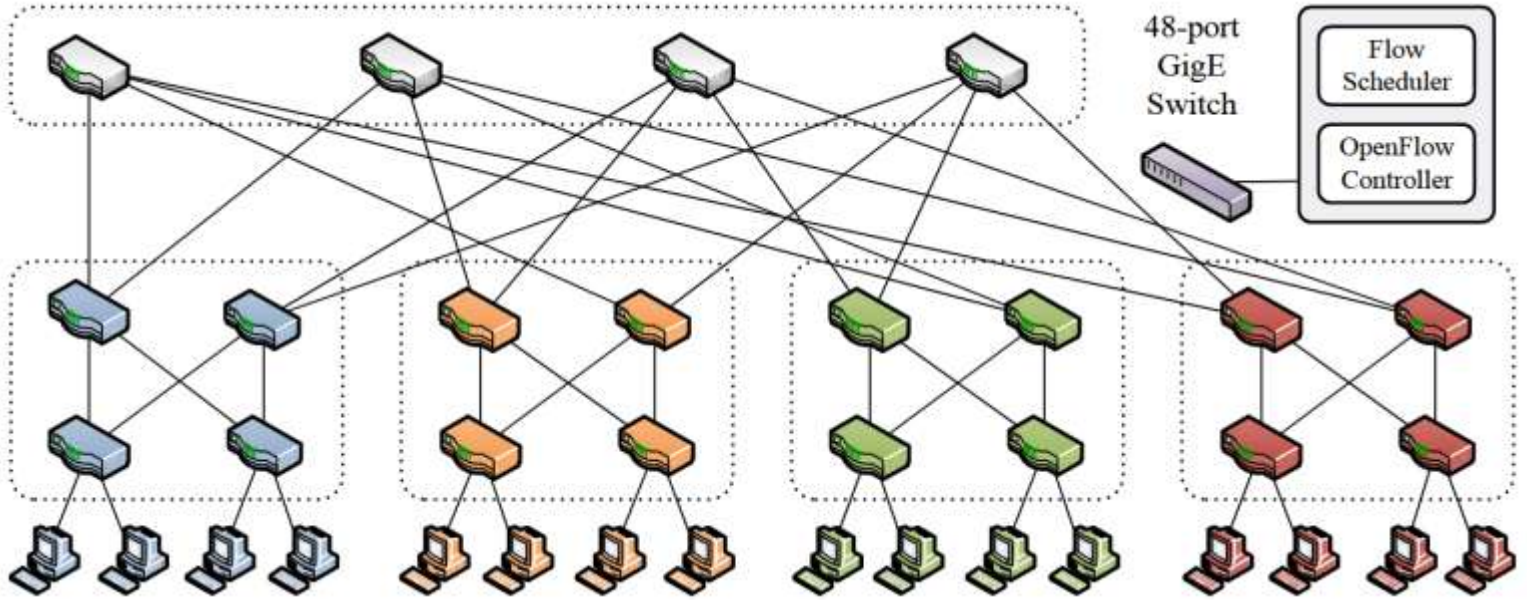
- Testbed 控制平面设计
  - 独立控制网络：连接所有交换机，用于传输监控和管理消息
  - 可选性强调：控制网络非Hedera架构必需，仅用于调试和对比
  - 替代方案：控制消息也可通过数据平面传输
  - 扩展性考虑：大规模网络中控制网络可采用传统树形结构
- 调度器位置：运行在独立机器上，连接至48端口控制交换机
- 流量分离：数据平面与控制平面物理分离
- 无修改要求：保持终端主机网络栈和操作系统无需修改



Implementation: 5.1 Topology

回顾：Fat-Tree拓扑及其好处

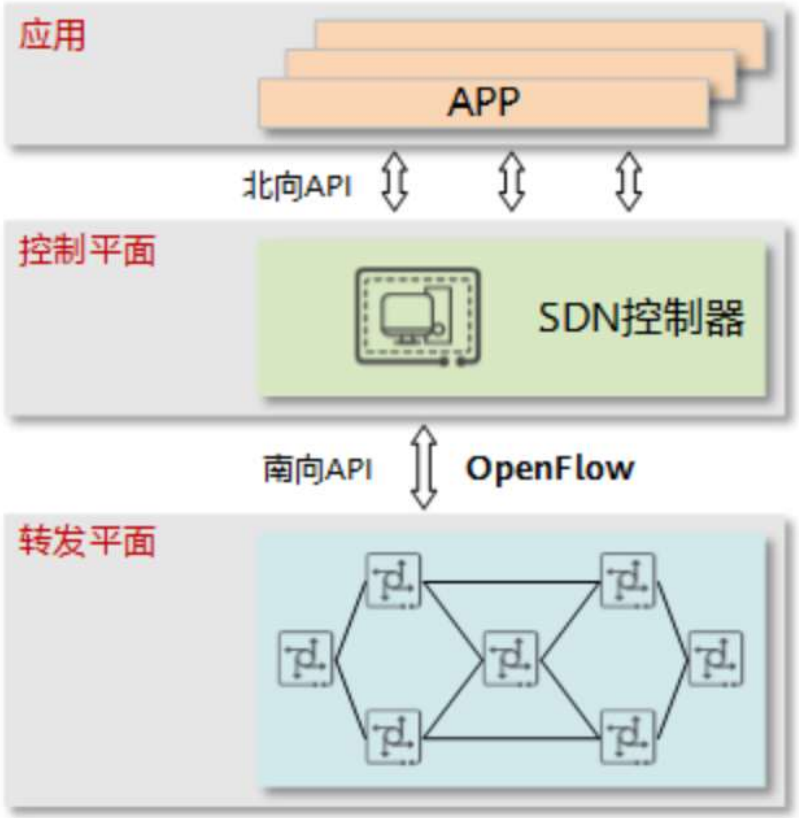
- Pod内部结构：k个pod，每个pod包含两层交换机
- 下层边缘交换机（edge switches）；上层聚合交换机（aggregation switches）
- 主机连接：每个边缘交换机连接(k/2)台主机
- 核心层：通过(k/2)^2台核心交换机互联所有pod
- 好处：
  - 路径数量：任意主机对之间存在(k/2)^2条等价路径
  - 路径特性：这些路径并非链路不相交（link-disjoint）
  - 调度挑战：需要为非冲突路径分配流以最大化二分带宽



Implementation: 5.3 OpenFlow Control

回顾：OpenFlow 控制框架

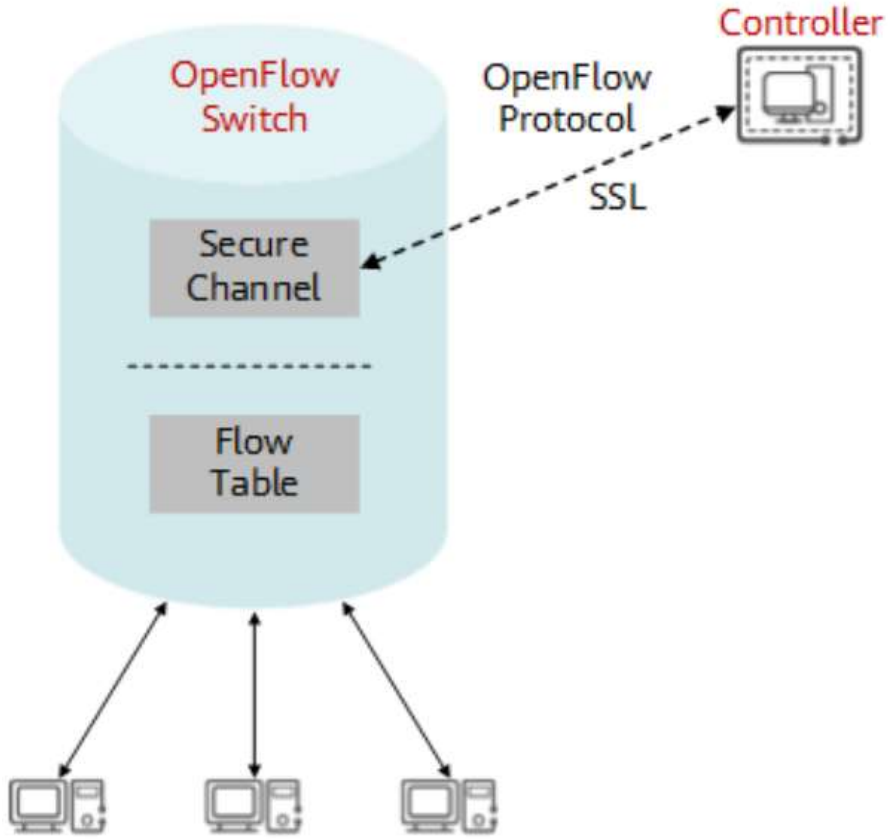
- Hedera 系统采用 OpenFlow 作为底层控制协议，该协议已成功移植到多种商用交换机（Juniper、HP、Cisco 等），提供了统一的转发表访问接口。这种标准化设计确保了系统在不同硬件平台上的兼容性和可移植性
- OpenFlow 交换机基于流表项对传入数据包进行匹配操作，支持多种动作类型：
  - 端口转发：指定数据包输出端口
  - 数据包丢弃：主动丢弃匹配的数据包
  - 广播转发：向多个端口同时转发
  - 数据包复制：实现多播功能



Implementation: 5.3 OpenFlow Control

回顾：OpenFlow 控制框架

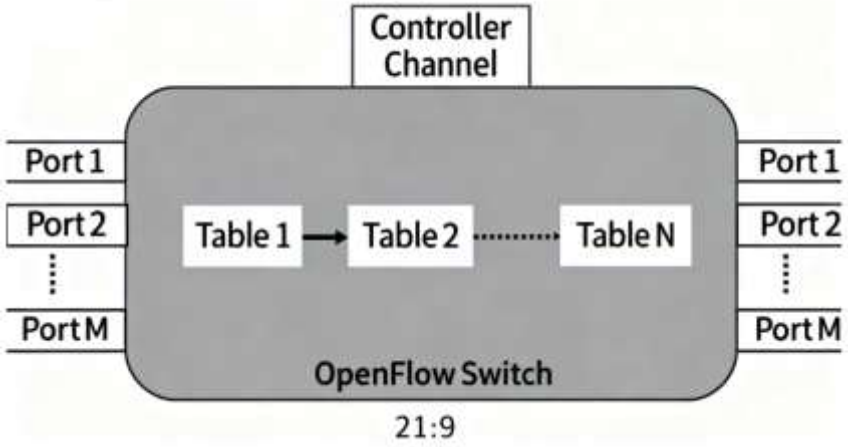
- 安全通信通道：交换机启动时自动建立与中央控制器的安全连接，形成集中式控制架构。控制器具备完整的流表管理能力
  - 流表查询：实时获取流状态信息
  - 流表操作：插入、修改、删除流表项
  - 高级功能：支持多种网络管理操作
- 统计信息维护
  - 交换机维护详细的流量统计数据：
    - 流级统计：每个流的字节数、持续时间等
    - 端口级统计：各端口的流量负载情况



Implementation: 5.3 OpenFlow Control

OpenFlow Switch 的动态调度工作流程

- 默认转发行为：当数据包不匹配任何现有流表项时，交换机自动创建新流表项
  - ECMP基础：基于等成本多路径算法选择输出端口
  - 硬件加速：后续数据包直接硬件转发

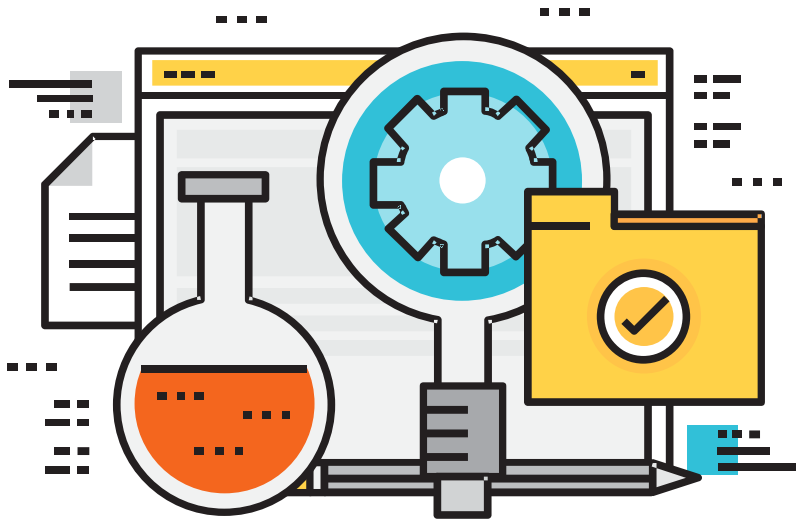


- 动态重路由机制：Hedera 调度器监控流状态，当流量超过阈值时：
  - 流识别：检测大流量流
  - 路径计算：计算最优转发路径
  - 表项更新：修改流表项实现重定向
  - 性能优化：确保网络资源高效利用
- 这种设计使得 Hedera 能够在保持硬件转发性能的同时，实现灵活的流量工程和动态负载均衡

Implementation: 5.5 Simulator

仿真器设计的必要性与方法

- 本文设计并实现了一个仿真器
- 仿真器的必要性：物理测试床仅支持16台主机的规模限制，需要模拟器来评估系统在更大规模网络环境（8,192台主机的大型数据中心网络场景）下的性能表现
- 现有分组级模拟器（如ns-2）无法满足大规模仿真需求：
  - 计算复杂度：模拟8,192主机各以1Gbps发送数据，60秒运行需处理2.5×10^{11}个分组
  - 时间成本：使用分组级模拟器完成单个测试案例需要71小时
  - 效率需求：迫切需要更高效的仿真方法



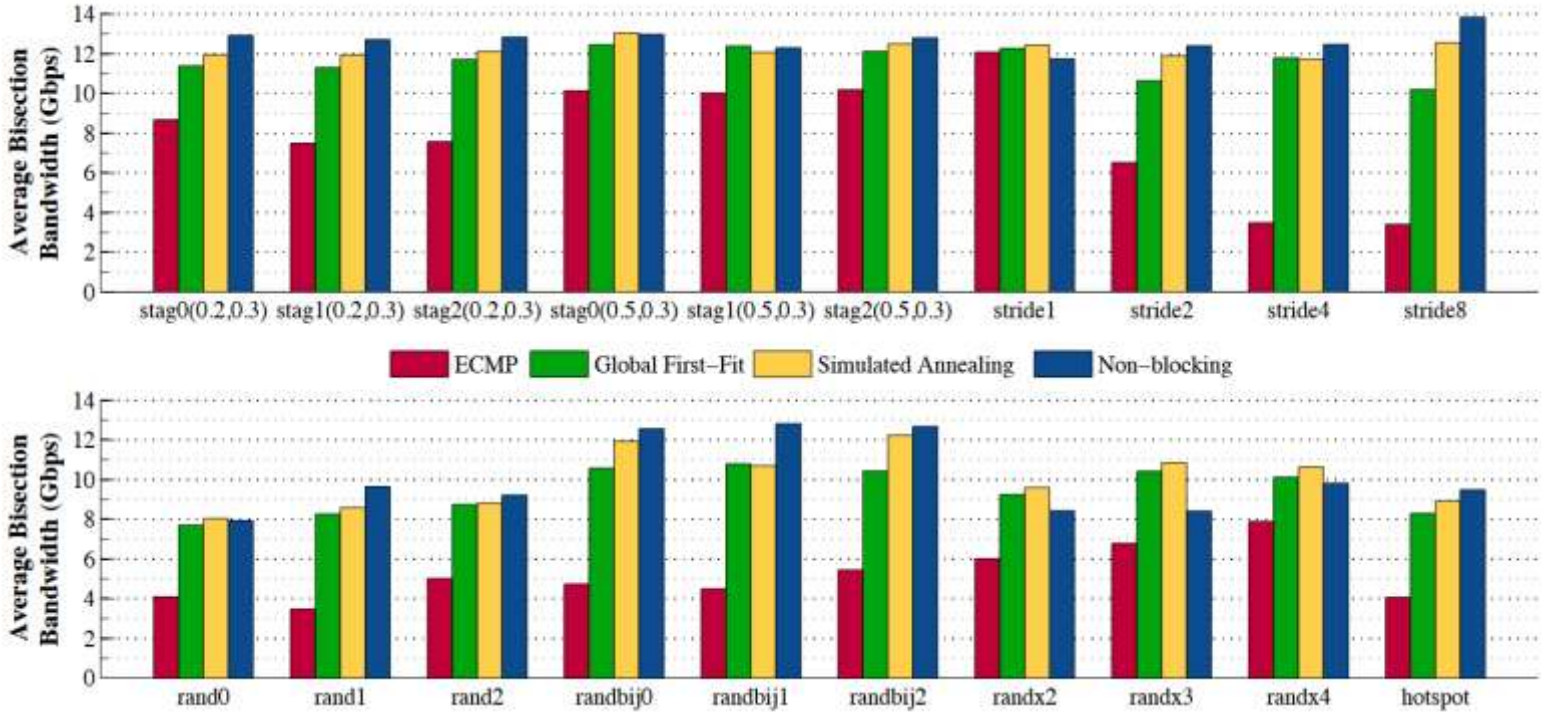
Outline

I. Introduction
II. Background
III. Architecture
IV. Estimation and Scheduling
V. Implementation
VI. Evaluation
VII. Review

Evaluation: 6.2 Testbed Benchmark Results

Testbed实验结果

- 算法性能对比：结果显示，在几乎所有测试的通信模式（随机、交错、跨步和热点模式）中，**全局首次适应和模拟退火算法都显著优于 ECMP**
- 通信局部性影响：算法性能随通信局部性程度提高而改善。对于 HPC 计算应用中常见的跨步模式，启发式算法能够持续计算正确的流到核心映射，有效利用胖树网络，而静态哈希性能随跨步增加恶化



- 左图：三种路由方法与 Non-blocking（理论最优）的物理测试台基准测试结果。图中显示了交错、跨步和随机通信模式所实现的网络二分带宽

Evaluation: 6.3 Data Shuffle

Testbed实验结果

- 数据洗牌是许多MapReduce/Hadoop操作中昂贵但必要的流程，其中每个参与主机都需要向所有其他参与主机传输大量数据。这种操作对网络带宽和调度效率提出了极高要求
- 实验结果显示，**集中式流调度相比静态ECMP**哈希路由表现出显著优势：
  - 高效场景：大流量、长持续时间流
  - 受限场景：小流量、短RPC类流
  - 理论依据：小流场景下哈希碰撞代价较低

	ECMP	GFF	SA	Control
Shuffle time (s)	438.44	335.50	335.96	306.37
Host completion (s)	358.14	258.70	261.96	226.56
Bisec. BW (Gbps)	2.811	3.891	3.843	4.443
Goodput (MB/s)	20.94	28.99	28.63	33.10

Evaluation: 6.4.1 Communication Patterns

模拟退火迭代次数对二分带宽的影响

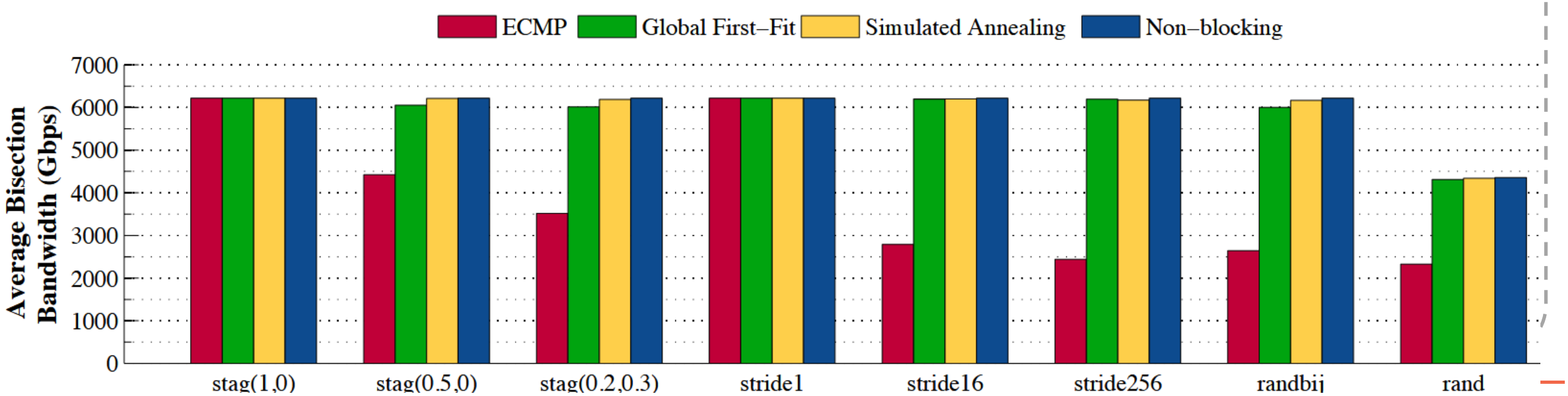
- 实验设计：通过调整模拟退火的迭代次数，测试不同规模网络（16、1024、8192台主机）的最终二分带宽百分比，并与非阻塞拓扑对比
- 核心结论：
  - 8,192主机
    - 迭代次数增加时，二分带宽提升并趋于稳定
    - 但仍显著低于非阻塞拓扑的理论值
  - 1,024主机
    - SA性能同样随迭代次数增加而改善
    - 但提升幅度有限，最高仍低于非阻塞拓扑
  - 16主机
    - SA性能接近非阻塞拓扑
    - **说明网络规模越小，优化算法越易接近最优**

	Number of Hosts		
SA Iterations	16	1,024	8,192
1000	78.73	74.69	72.83
50000	78.93	75.79	74.27
100000	78.62	75.77	75.00
500000	79.35	75.87	74.94
1000000	79.04	75.78	75.03
1500000	78.71	75.82	75.13
2000000	78.17	75.87	75.05
Non-blocking	81.24	78.34	77.63

Evaluation: 6.4.1 Communication Patterns

动态算法 vs. 静态ECMP与非阻塞拓扑

- 动态算法优势：在8,192主机的Fat-Tree中，动态流量调度算法（如SA）显著优于静态ECMP
- ECMP因路径冲突（尤其当局部通信概率降低时）导致带宽浪费，而动态算法通过全局调度减少重叠，提升聚合二分带宽
- 与非阻塞拓扑对比：动态算法虽优于ECMP，但与非阻塞拓扑（理论最优）仍存在性能差距，表明大规模网络中硬件限制难以完全克服



Evaluation: 6.4.4 Complexity of Simulated Annealing

模拟退火算法复杂度

- 复杂度独立于主机规模：运行时间仅与流数量相关，与主机数量无关，凸显算法对大规模数据中心的适应性
- 实际性能：在数千主机、合理流量的场景下，即使进行 10,000次迭代，模拟退火运行时间仅需数十毫秒
- 工程意义：该性能表明模拟退火算法**具备实际部署可行性**，可满足数据中心调度对实时性的要求（百毫秒级控制周期）

	1,024 hosts		8,192 hosts	
Iterations	f =3,215	f =6,250	f =25k	f =50k
1000	2.997	5.042	6.898	11.573
5000	12.209	20.848	19.091	32.079
10000	23.447	40.255	32.912	55.741

Evaluation: 6.4.5 Control Overhead

集中式调度控制开销

- 三阶段流程：交换机上报流信息 → 调度器计算需求与路由 → 下发新流表
- 计算时间主导：如表6所示，控制回路总时长主要取决于调度计算时间（100ms），而非通信延迟或带宽限制
- 可扩展性验证：在每主机大流量数量合理时（如10条/主机），集中式调度方案在超大规模数据中心（数万台主机）仍可稳定运行

	Flows per host		
Hosts	1	5	10
1,024	100.2	100.9	101.7
8,192	101.4	106.8	113.5
27,648	104.6	122.8	145.5

Table 6: Length of control loop (ms).

Outline

I. Introduction
II. Background
III. Architecture
IV. Estimation and Scheduling
V. Implementation
VI. Evaluation
VII. Review

7 Related Work

相关工作

- 1、负载均衡方案的局限性：VL2和Monsoon采用基于流的Valiant负载均衡（VLB），但容易因长期流量冲突导致带宽损失，且无法充分利用多路径优势
- 2、单层架构与递归拓扑的不足：
  - SEATTLE提出单层2域架构，但未解决多路径传输问题；
  - DCell和BCube通过递归拓扑支持大规模扩展，但依赖多网卡服务器，深层链路易过载且未优化多路径调度
- 3、广域网调度技术的适配问题：TexCP、MATE等广域网多路径流量调度方案依赖交换机显式拥塞通知（ECN），而现有数据中心交换机缺乏支持；FLARE的流粒调度可能因低延迟环境引发数据包乱序
- 4、集中式调度的探索：类似Hedera的集中式调度方案（如4D、Tesseract）通过中心控制器优化流分配，但需解决可扩展性与交换机兼容性问题
- 5、Clos网络理论的局限：Clos网络的理论研究（如非阻塞交换、分组调度算法）虽提升单级性能，但未解决基于商用组件的多级架构实际部署挑战

Review

- 本研究的核心结论是：**利用掌握全局信息的中心调度器来动态调度大流**，能显著提升网络性能
- 关键发现如下：
  - **中心化调度优势**：与传统的基于哈希的ECMP负载均衡相比，中心调度器能更高效地利用网络资源，尤其在网络负载较重时优势更明显
  - **高效调度策略**：通过专注于调度占主要带宽的大流来控制开销；其中，模拟退火算法表现最佳，能提供接近最优的二分带宽
  - **可行性与高性价比**：原型系统证明该方案可行，且以适中的额外部署成本（如增加少量服务器）即可为数据中心网络带来巨大的带宽收益