一、基本信息

个人情况

赵功名，1993年生，中国科大副教授，中国科学院青促会会员，CCF高级会员，CCF互联网/分布式计算与系统/网络与数据通信专委会执行委员，INFOCOM、IWQoS、ICPADS、MSN、NPC、ICA3PP 等会议TPC member，主要致力于云计算、数据中心网络、在网计算、智算网络、网络算法设计等方向研究

获奖情况

曾获2022年度安徽省自然科学一等奖（排名第3）、中国科学院优博、中国电子学会优博、ACM SIGCOMM CHINA“新星奖”，3次荣获华为难题揭榜“火花奖”

科研成果

近年来发表CCF A/B类论文60余篇，含NSDI、ToN、TPDS、TC、INFOCOM、WWW 等一作/通讯 CCF A 类论文 28 篇，ICNP、ICDCS、IWQoS、SECON、TCOM、ComNet 等一作/通讯 CCF B 类论文 20 篇，被 30 余位 ACM/AAAS/IEEE Fellow 等专家团队正面评价，授权国家发明专利12项

项目承担

作为负责人主持国家自然科学基金面上项目、青年项目、华为合作项目、华为智库项目、阿里云合作项目、移动合作项目、CCF-腾讯犀牛鸟基金、神州信息合作项目等项。作为技术负责人累计承担 8 项华为合作项目、2 项字节合作项目等，其中华为合作项目获得“优秀技术成果”（仅 10%项目获奖）

一、基本信息

助教信息

田佳林，18737516339，jltian@mail.ustc.edu.cn

邓立鑫，18173059857，denglx@mail.ustc.edu.cn

朱家成，13627378265，zhu_jc@mail.ustc.edu.cn

李紫惠，13255317680，lizihui2002@mail.ustc.edu.cn

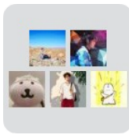
主页信息

团队主页：https://int-ustc.github.io/

个人主页：https://gmzhao-ustc.github.io/

课程主页：https://acn-2025.github.io/

课程群



群聊：2025年秋高级计算机网络课程群

该二维码7天内(9月21日前)有效，重新进入将更新

一、基本信息

考试占比

分数占比

期末开卷考试占60%，网络方向小论文撰写占40%

考试范围

课上讲解的论文为主，同学汇报的论文为辅

论文汇报

两到三人一组，从推荐的论文列表或近五年A类网络方向论文中选择汇报

汇报总时长约35分钟，问答交流10分钟，参加论文汇报的同学有5-8分的额外加分

有意向同学请组队并选好论文及汇报时间发送至lizihui2002@mail.ustc.edu.cn

先到先得，我们会及时在主页更新汇报安排，请勿重复选择论文及汇报时间点

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

A Scalable, Commodity Data Center Network Architecture

Mohammad Al-Fares, Alexander Loukissas, Amin Vahdat

SIGCOMM 2008

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. Implementation and evaluation

V. Review

Outline

I. Introduction
   1. Data Centers
   2. Traffic Patters in DCs
   3. DC Architectures
   4. Supplement: Fundamental Concepts
II. Background
III. Architecture
IV. Implementation and evaluation
V. Review

Introduction：Data Centers

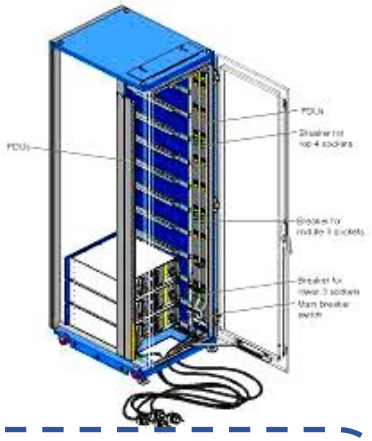
- 我们日常使用的各种在线服务，例如谷歌搜索、YouTube视频、淘宝购物、微信聊天等，都依赖于大量计算机处理信息与存储数据。
- 这些计算机、连接它们的网络设备，以及供电与冷却系统等支持设施，都集中部署在被称为数据中心的巨型设施中。
- 数据中心的三大核心功能为：存储数据、运行应用程序和处理请求。

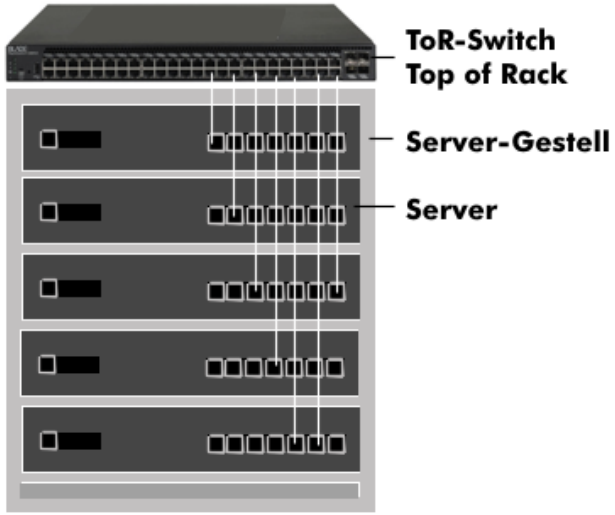
谷歌数据中心

微软水下数据中心

Introduction：Data Centers

数据中心设备





- 数据中心里核心硬件主要有三类：
  - 服务器：成千上万的主机，负责存储和处理数据
  - 交换机：连接所有设备的高速网络枢纽，负责数据流动
  - 供电与冷却系统：7x24小时不间断供电和空调，防止设备过热宕机
- 所有设备通过机柜整齐排列

8

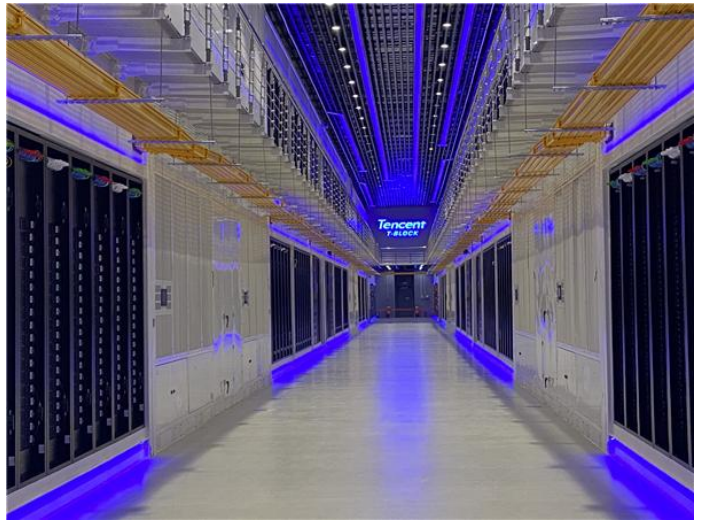
Introduction：Data Centers

数据中心正在快速扩张

- 随着生成式人工智能的蓬勃发展，全球数据总量与算力规模正保持高速增长态
  - 据IDC数据，2023年全球数据总产量为129.3ZB，过去五年平均增速超过25%，预计2028年将达到384.6ZB，较2023年翻近 3倍
- 数据成为新的生产资料，而算力则成为新的生产力
  - 中国信通院测算，2023年全球计算设备算力总规模为1397EFlops，同比+54%，其中智能算力规模（换算FP32)为875EFlops，同比+94%，占总算力的比重为63%
- 2017年我国机架总数不到500万，2022年为650万； 而2024年中国内地数据中心行业市场规模达到3048亿元、标准机架规模突破1000万架。

Introduction：Data Centers

腾讯贵安七星数据中心



- 腾讯把服务器藏进了贵州一处山洞，里面放置了30万台服务器，这里就是腾讯贵安七星数据中心，占地面积约47万平米，隧洞的面积超过3万平米，可以塞下4个标准足球场。
- 为什么把数据中心修在山洞里？从实际能耗来看，2023 年我国算力中心总用电量约为 1500 亿千瓦时，仅散热环节就消耗了约 400 亿度电，这一数字相当于三峡水电站半年的发电量。

Introduction：Data Centers

移动算力的"海上方舟"

- 日本三井商船株式会社（MOL）正在规划一个将船舶改造为 DC 的项目。该公司准备将一艘120米长的运输船改造为浮动数据中心。准备2027年正式投入运营。
- 这个数据中心的特色就是其冷却系统——直接引入海水或河水进行散热，使能效提升30%以上
- 相较于陆地数据中心，浮动方案具备三大战略优势：第一是地理灵活性：可根据电力供需变化在沿海、港口或公海间迁移；其次是建设周期短，从改装到投运仅需1年，较传统项目提速400%；第三是成本可控性，规避大都市区高昂地价，停泊费用仅为陆地成本的15%

Introduction：Data Centers

千奇百怪的选址

- 美国国家科学基金会数据中心将高性能计算集群安置在南极的冰川上；
- 微软开发的纳蒂克数据中心项目，直接将数据中心转移到海底世界中；
- SuperNAP将数据中心建在了美国拉斯维加斯的沙漠中；
- 阿里巴巴千岛湖数据中心是国内首个采用自然水制冷技术的数据中心，空调系统采用两路进水，湖水和冷冻水，可以实现同时或单独运行……
- 数据中心的选址看似奇怪，其实极其考究：
  - 大量且廉价的电力；设备易冷却；地价便宜；隐秘性和安全性；和其他数据中心的连接；税收优惠……

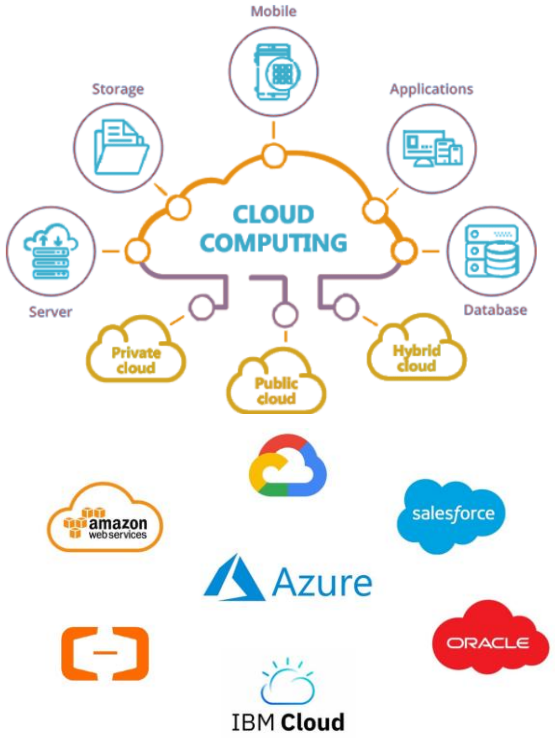
Outline

I. Introduction
   1. Data Centers
   2. Traffic Patters in DCs
   3. DC Architectures
   4. Supplement: Fundamental Concepts
II. Background
III. Architecture
IV. Implementation and evaluation
V. Review

Introduction：Traffic Patters in DCs

数据中心内的流量

- 大规模集群中存在大量机器间通信，其核心瓶颈往往在于节点间的通信带宽。
  - MapReduce：必须执行大量数据混洗操作，以传递映射阶段的输出结果，才能继续执行归约阶段。
  - 网络搜索引擎：通常需要与托管倒排索引的集群中所有节点进行并行通信，以返回最相关的结果。



对网络的要求

服务器需要持续相互通信，要求数据中心内部具备**高速、高效且可靠**的网络互联能力

Outline

Introduction：DC Architectures

方案一：专用硬件与协议

- 例如：InfiniBand、Myrinet
- 不采用商用硬件组件，**成本高昂**
- 与TCP/IP应用不兼容

方案二：采用商用以太网交换机与路由器互联集群机器

- 无需修改应用程序、操作系统及硬件
- 但存在明显缺陷：集群总带宽随规模扩展的**效果差**，且要达到最高带宽级别时，成本会随集群规模呈现非线性增长

方案二更受青睐

- 出于兼容性和成本考量，大多数集群通信系统采用第二种方案。
- 然而在大型集群中，通信带宽可能因通信模式的影响出现显著超额订阅（oversubscription）现象。
- 如何通过商用解决方案应对这些瓶颈？例如采用10Gbps交换机与路由器？

Introduction：DC Architectures

IB网络和以太网的区别在哪?

特性	InfiniBand (IB)	以太网 (Ethernet)
设计目标	为高性能计算（HPC）设计，注重超低延迟和高带宽 。	为通用数据通信设计，注重普适性、兼容性和成本效益 。
核心技术	基于通道（Channel-based）的点对点交换网络。	基于数据包（Packet-based）的共享介质或交换网络。
性能	**延迟极低**：通常在亚微秒（sub-microsecond）级别。	**延迟相对较高**：通常在几微秒到几十微秒级别
成本	硬件（交换机、网卡）通常比以太网更昂贵，因为它不采用商用部件	受益于巨大的市场规模和充分的竞争，是成本效益最高的网络技术
拓扑结构	常采用胖树（Fat-Tree）等Clos网络拓扑，以实现可扩展的高带宽 。	拓扑灵活，但传统数据中心多采用分层树状结构 。

Introduction：DC Architectures

数据中心网络架构的理想特性

- 可扩展的互联带宽（Scalable interconnection bandwidth）
  - 任意主机均能以本地网络接口的全带宽能力与其他主机进行通信
- 规模经济效益（Economies of scale）
  - 采用廉价的现成以太网交换机作为大规模数据中心网络的基础架构
- 向后兼容性（Backward compatibility）
  - 整个系统应向后兼容运行以太网和IP协议的主机

Outline

Introduction：本文涉及的基础概念

- **交换机**（Switch）：在数据中心语境下，“交换机”通常指集成第二层交换与第三层路由功能的设备。因其需同时处理局域网内服务器通信与跨子网服务器通信
- **聚合带宽**（Aggregate Bandwidth）：网络中所有连接带宽容量的总和
- **二分带宽**（Bisection Bandwidth）：假设将整个网络划分为两个对等部分，连接这两个部分的全部链路带宽之和即为二分带宽，对应图论中的割概念。该指标用于评估网络的整体通信能力
- **商用硬件**（Commodity Hardware）：指广泛可得、成本相对较低且功能标准化的通用硬件设备，区别于专用化、昂贵的高端定制设备

20

Introduction：本文涉及的基础概念

三层分层网络互连模型（Three-tier Hierarchical Model)

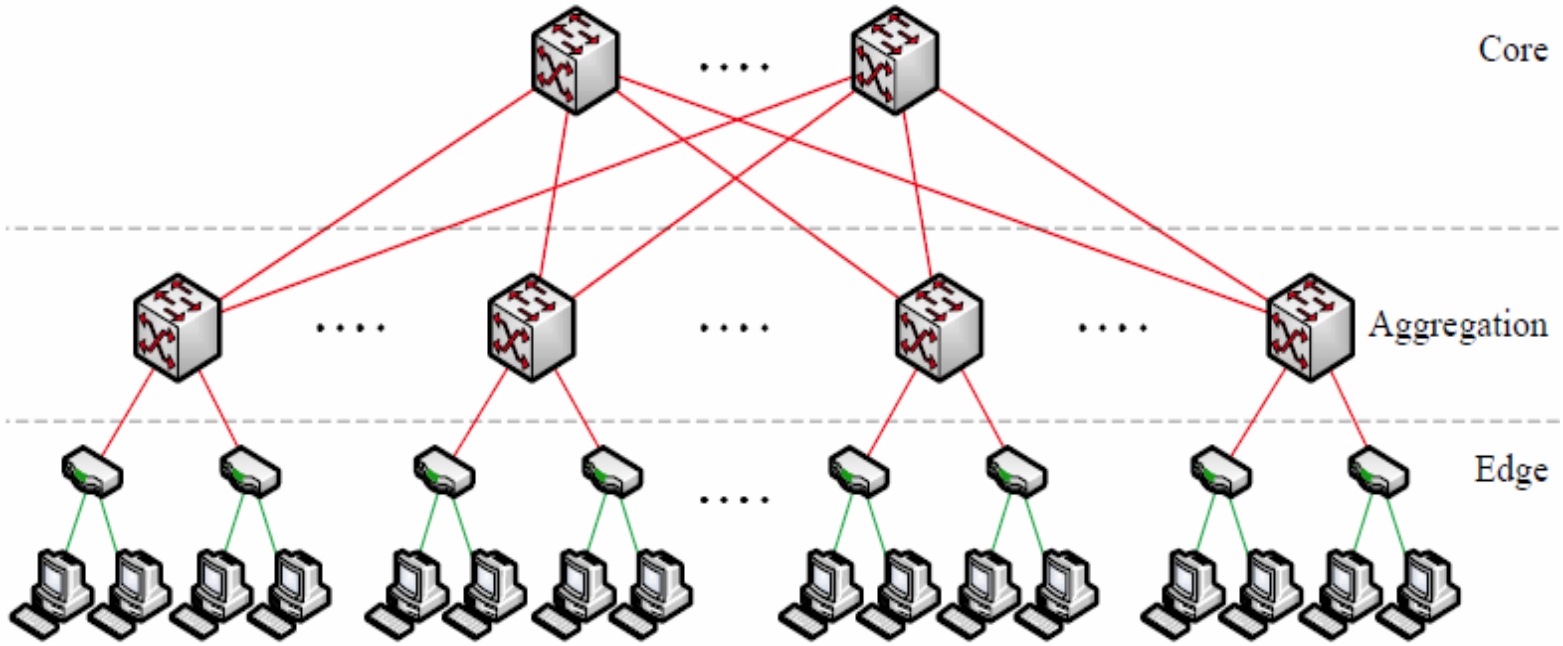
- 最早由 Cisco 提出，是一种被广泛采用的网络设计方法，尤其常见于企业环境中。该模型通过将网络划分为三个功能明确的层级来实现架构优化：
  - Edge (Access Layer)：网络最外层，直接连接终端设备（如服务器、计算机）的层级
  - Aggregation (Distribution Layer)：连接接入层与核心层的中间层级，负责流量聚合及实施网络策略（如路由策略、安全策略）
  - Core (Core Layer)：网络骨干层，专用于在不同汇聚层之间进行极高速数据转发的层级

Outline

I. Introduction
II. Background
    2.1 Current Data Center Network Topologies
    2.2 Clos Networks/Fat-Trees
III. Architecture
IV. Implementation and evaluation
V. Review

Background：2.1 Current Data Center Network Topologies

2.1.1 Topology



- 三层架构：核心层、汇聚层、边缘层（机柜顶部交换机）
- 两种交换机类型：
  - 48端口千兆以太网交换机：配备4个万兆上行链路，用于树形架构边缘层
  - 128端口万兆以太网交换机：用于通信层级中的更高层级
- 两类交换机均支持所有直连主机以网络接口全速相互通信

2.1.2 Oversubscription

- 超额订阅：指特定通信拓扑的总二分带宽与终端主机间最差情况下可达聚合带宽的比值
- 理想状态
  - 1:1（所有主机均可能以网络接口全速与任意其他主机通信）
  - 无带宽争用，性能最高，但成本也最高
- 典型设计
  - 通常存在2.5:1至8:1的超额订阅
  - 对于1 Gb/s网卡，5:1 意味着在某些情况下保证带宽仅为 200 Mbps
  - 大多数实际的数据中心设计都有意采用超额订阅，以在成本与性能之间取得平衡

Background：2.1 Current Data Center Network Topologies

2.1.2 Oversubscription

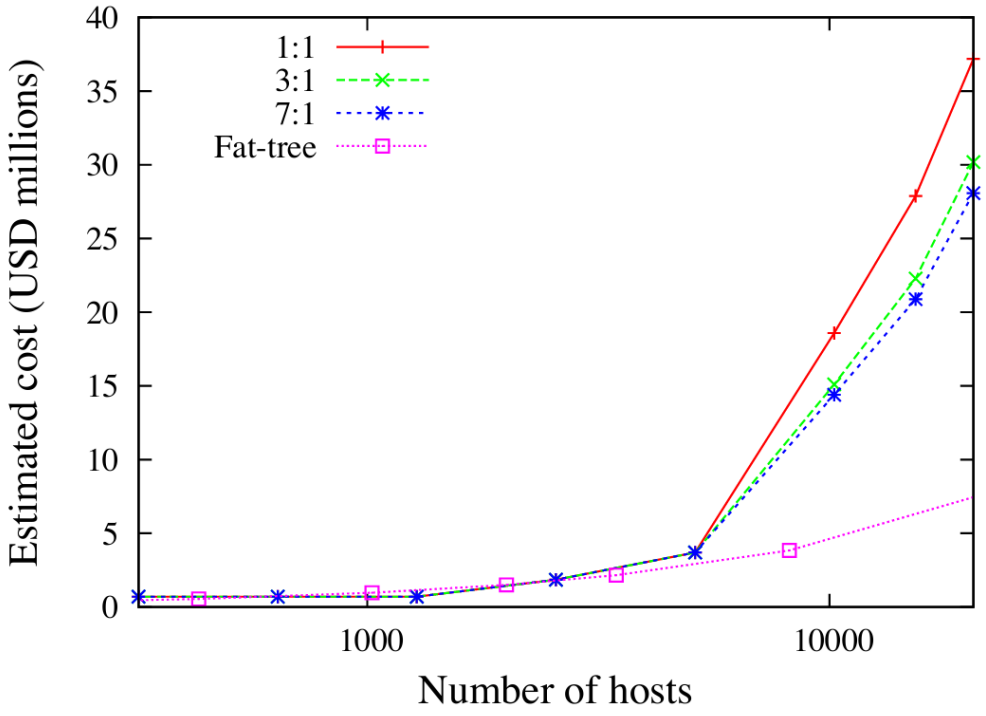


Figure 2: Current cost estimate vs. maximum possible number of hosts for different oversubscription ratios.

- **传统架构成本高昂：**在传统分层架构中，要为大规模集群（如超过10,000台主机）提供高带宽（即低收敛比），成本会急剧上升
- **收敛比是降低成本的手段：**引入较高的收敛比（如3:1或更高）是降低传统架构成本的常用方法，但这会牺牲网络性能
- **胖树架构的成本优势：**与传统架构相比，胖树架构在提供1:1全带宽的情况下，成本要低得多，且成本增长曲线更为平缓

2.1.3 Multi-path Routing

- 大规模集群需“多根树”结构
  - 要为大量主机提供1:1超售的全带宽，需采用具有多个核心交换机的 “多根树” 拓扑
  - 单一核心交换机的设计，其规模极限约为 1280个节点（基于一个128端口的10G交换机）
  - 无带宽争用，性能最高，但成本也最高
- 等价多路径 ECMP
  - 多根树需要 ECMP 等技术来利用多条路径
  - ECMP的工作原理： 在多个可用路径上对数据流进行静态的负载分割

2.1.3 Multi-path Routing

ECMP的局限性

- 局限一：静态分配，无视流量带宽
  - ECMP仅根据流进行静态分配，决策时不考虑每条流所需的实际带宽
  - 这可能导致即使是在简单的通信模式下，链路实际上仍被超额订阅
- 局限二：路径数量不足
  - 当前ECMP实现通常将路径数量限制在 8至16条
  - 对于超大规模数据中心，这个路径多样性通常不足以提供高的二分带宽
- 局限三：路由表膨胀
  - 路由表条目数会随着路径数量的增加而成倍增长
  - 这会导致：1) 硬件成本增加； 2) 可能增加查找延迟，影响性能

2.1.4 Cost

成本是首要驱动因素

- 构建大型集群网络互联的成本极大地影响最终的设计决策
- 引入超额订阅（Oversubscription） 是降低总成本的典型手段
  - 假设：边缘层：48端口千兆交换机，单价 $7,000；汇聚/核心层：128端口万兆交换机，单价 $700,000
  - 互联20,000台主机并实现 1:1 全带宽，硬件成本高达约 $3700万美元
  - 实现 3:1 的超额订阅（单主机保证带宽约330Mbps），成本则显著降低
- 结论： 现有技术为大型集群提供高带宽成本极高，而基于 FatTree 的架构能以更合理的成本提供可扩展带宽。规模集群成本仍可能极高（超$6.9亿），但至少技术上可行（一个拥有**27,648** 节点、所有节点之间都可实现 **10 Gbps** 带宽的集群）

2.1.4 Cost

传统架构 vs. Fat-Tree 架构的深层对比

- 传统分层架构的局限：
  - 最大集群规模长期受限于高端交换机的端口密度
  - 早期万兆交换机成本极高
  - 最关键的是：目前技术上根本无法用传统架构构建一个所有27,648个节点都具备10Gbps潜在带宽的集群，因为没有超过10G速率的商用交换机或以太网标准
- Fat-Tree 的优势：
  - 扩展性良好，总成本下降更快（更早遵循商品化定价趋势）
  - 无需高速上行链路要求
  - 虽构建同等规模集群成本仍可能极高（超$6.9亿），但至少在技术上是可行的

2.1.4 Cost

	Hierarchical design			Fat-tree		
Year	10 GigE	Hosts	Cost/ GigE	GigE	Hosts	Cost/ GigE
2002	28-port	4,480	$25.3K	28-port	5,488	$4.5K
2004	32-port	7,680	$4.4K	48-port	27,648	$1.6K
2006	64-port	10,240	$2.1K	48-port	27,648	$1.2K
2008	128-port	20,480	$1.8K	48-port	27,648	$0.3K

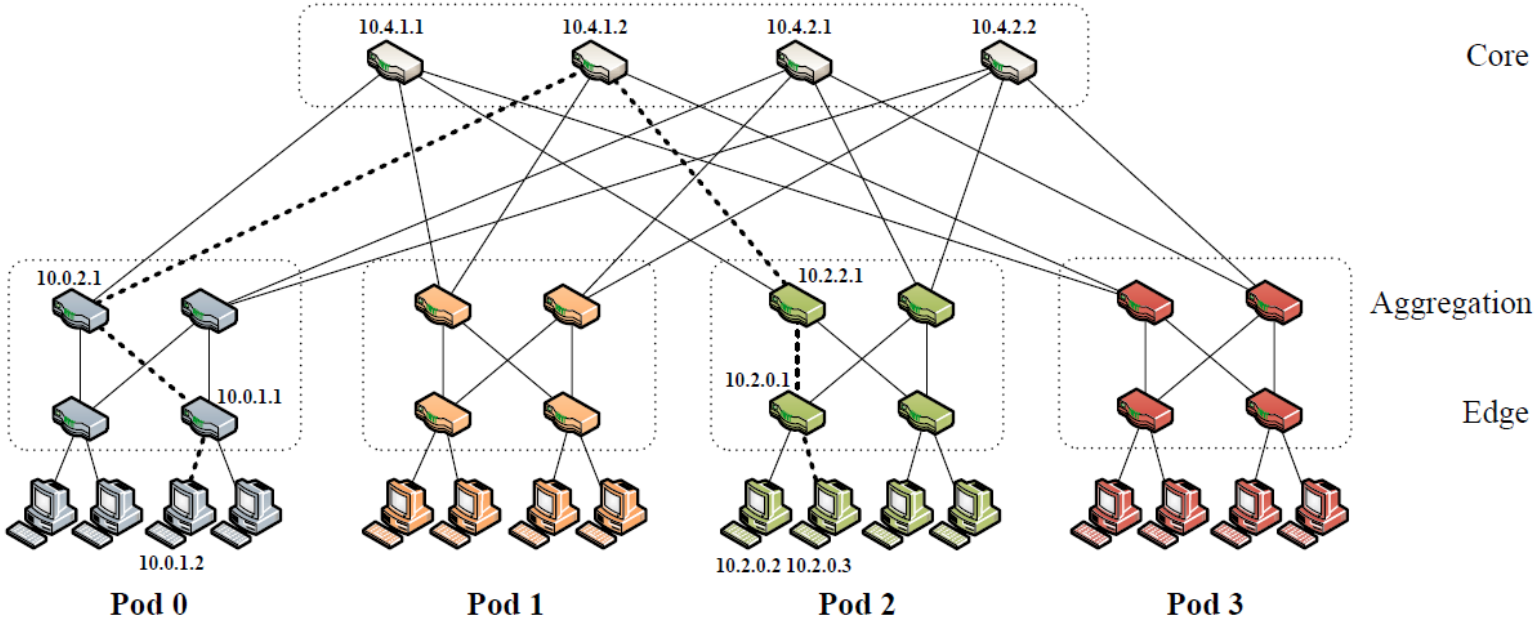
Table 1: The maximum possible cluster size with an oversubscription ratio of 1:1 for different years.

- 与依赖昂贵、非商用高端交换机的传统分层架构相比，采用商用交换机搭建的胖树拓扑在网络扩展性和成本效益上都表现出显著且持久的优势

Outline

I. Introduction
II. Background
    2.1 Current Data Center Network Topologies
    2.2 Clos Networks/Fat-Trees
III. Architecture
IV. Implementation and evaluation
V. Review

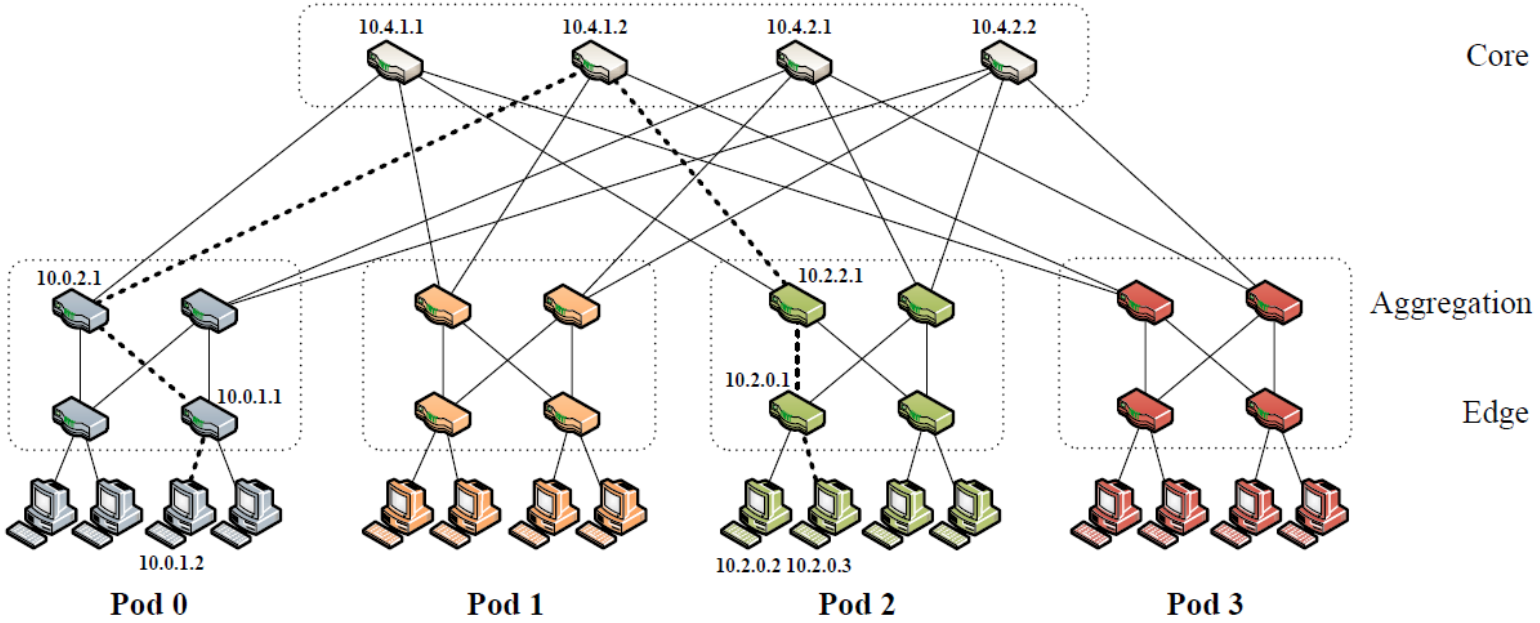
Background：2.2 Clos Networks/Fat-Trees



Fat-Tree 架构

- Fat-tree 是一种特殊 Clos 网络拓扑（即**分层拓扑架构**）
- 核心目标是使用商用以太网交换机构建高性能的网络互联
- 所有的交换机都是同一型号的

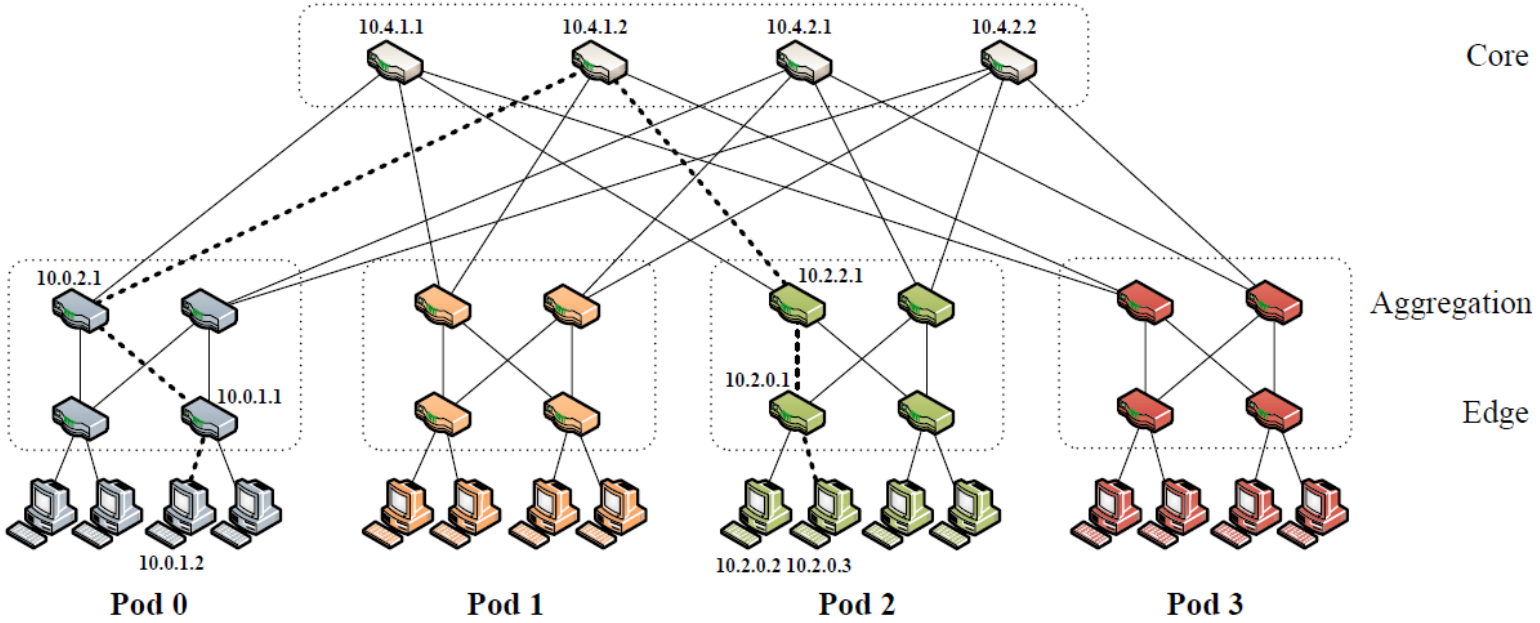
Background：2.2 Clos Networks/Fat-Trees



k-ary Fat-Tree 的结构组织

- 网络结构由一个**核心参数 k**（通常为交换机端口数）决定
- 网络整体呈对称的、分层级的树形结构
- 关键特征： 从树根（核心层）到树叶（边缘层），链路带宽逐层聚合，而非传统树结构的逐层收敛，从而消除了网络瓶颈

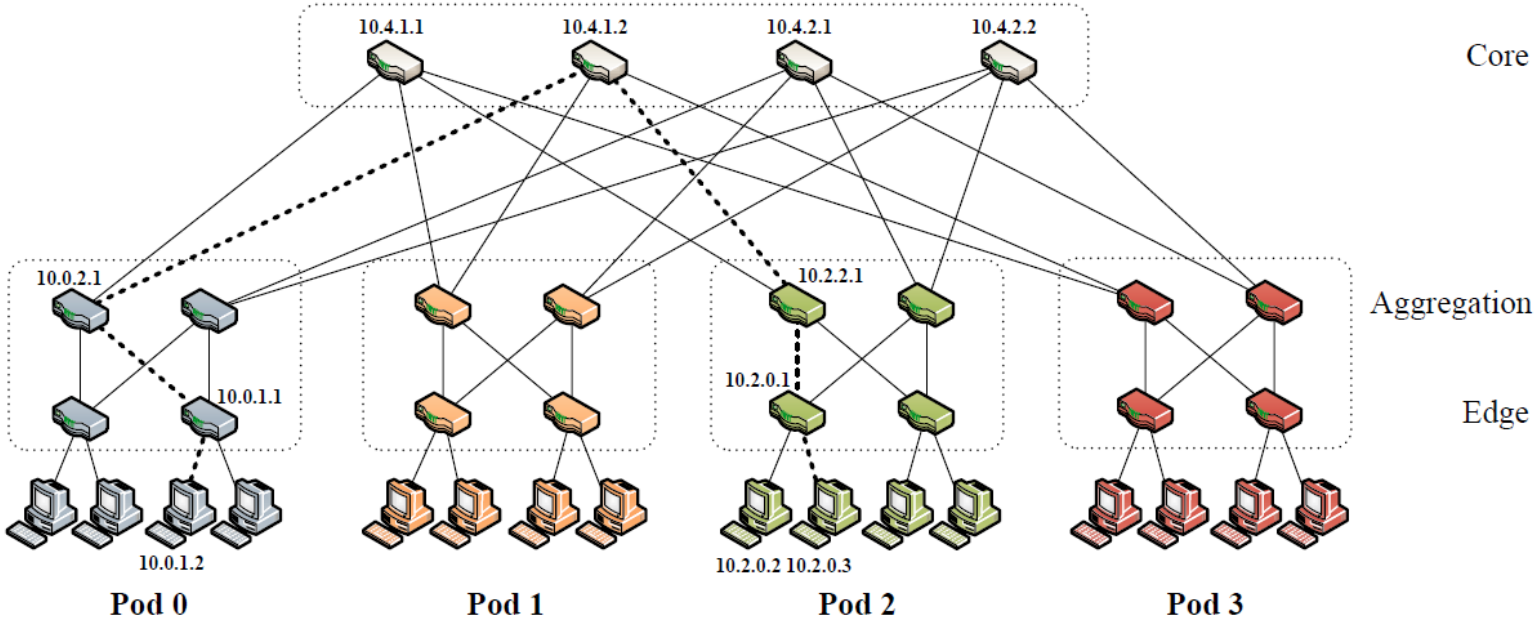
Background：2.2 Clos Networks/Fat-Trees



k-ary Fat-Tree 的结构组织

- 有 k 个pod，每 pod 内有两层，每层有 k/2 个交换机
- 对于 edge layer 中的每个 k-port 交换机
  - 有 k/2 个端口直接连到主机 -->可以算出，整个拓扑中主机数量为 $k^3/4$
  - 其余 k/2 个端口向上连接到位于 aggregation layer 的 k/2 个交换机

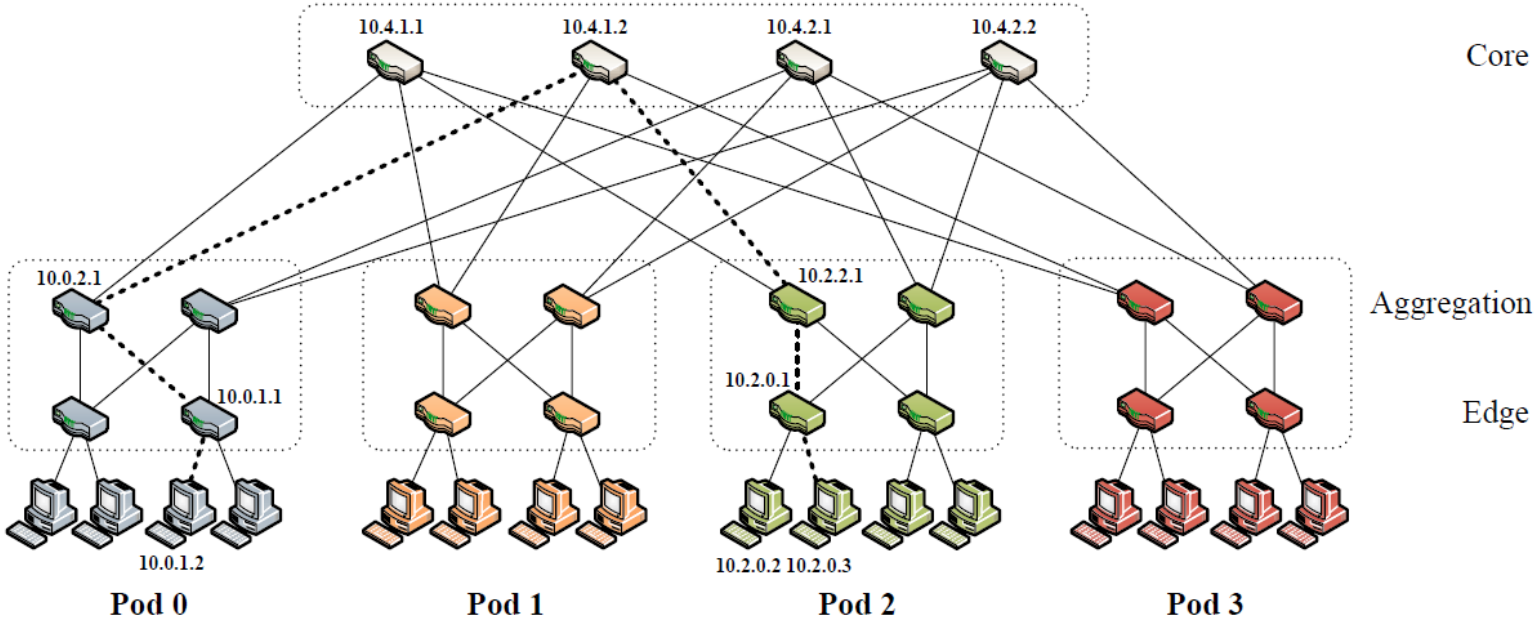
Background：2.2 Clos Networks/Fat-Trees



k-ary Fat-Tree 的结构组织

- 对于 aggregation layer 中的每个 k-port 交换机
  - 有 k/2 个端口向下连，其余 k/2 个端口向上连到 core
  - 由此可得 core 交换机数量为 $(k/2)^2$
  - 思考：如何计算?

Background：2.2 Clos Networks/Fat-Trees



k-ary Fat-Tree 的结构组织

- 对于 core 交换机
  - 每个端口向下连接到一个 pod 内的某台交换机，第 i 个端口连接到第 i 个 pod
- 跨 pod 主机间最短路的数量
  - $(k/2)^2$，和 core 交换机数量相同
  - 思考如何计算?

Background：2.2 Clos Networks/Fat-Trees

Fat-Tree 具体实例

- 以一个由 48端口 千兆以太网（GigE）交换机构建的胖树为例
  - 网络包含 48个Pod，每个Pod内含两层
    - 边缘层 (Edge Layer): 24台交换机
    - 聚合层 (Aggregation Layer): 24台交换机
    - 共1,152个子网，每个子网包含24台主机，共27,648台主机
- 卓越的性能与冗余
  - 位于不同 Pod 的任意两台主机之间，存在 576条 等价路径
  - 为负载均衡和链路故障冗余提供了极高的灵活性
- 显著的成本效益
  - 部署此胖树网络的硬件成本估算约为 864万美元
  - 与之前提到的传统技术实现相同规模所需的 3700万美元 成本形成鲜明对比。
- Fat-Tree 以传统方案1/4的成本，实现了大规模、高性能、高冗余的网络互联

Outline

Architecture：3.1 Motivation

Fat-Tree 全带宽路由难题

- 核心目标：均匀分发流量
  - 要实现最大的二分带宽，必须将从任一Pod发出的流量尽可能均匀地分布到所有核心交换机上
- 传统路由协议（如OSPF）的局限性
  - 问题一：路径选择单一
  - OSPF等协议以“跳数”为度量，在胖树中任意两主机间虽有(k/2)^2条等代价路径，但只会选择其中一条
  - 导致交换机去往某个子网的流量集中在一个端口，无法利用其他等价路径。
- 问题二：可能造成核心层流量集中
  - 根据OSPF消息到达的时序，可能只有极少数（甚至一个） 核心交换机被选为Pod间链路
- 后果： 核心层热点拥塞，完全浪费了Fat-Tree 固有的路径冗余优势

Architecture：3.1 Motivation

Fat-Tree 全带宽路由难题

- 扩展协议（如 OSPF-ECMP）的新问题
  - 首先，此类协议在商用交换机上不可用
  - 而且会导致所需前缀数量爆炸式增长（一个底层Pod交换机需要 k * (k/2)^2条前缀），无法实际部署
- Fat-Tree 创新方案：两级路由表
  - 需求： 需要一种简单、细粒度的方法来利用拓扑结构实现Pod间流量扩散
  - 核心思想：交换机必须能识别出需要被均匀分发的流量类别，并对其进行特殊处理
  - 实现方案：两级路由表，根据目的IP地址的低位比特来分散输出流量（之后介绍）
  - 优势： 在不引起前缀爆炸的前提下，智能地利用所有可用路径，实现流量负载均衡

Outline

Architecture：3.2 Addressing

Fat-Tree 的编址

- 地址池：所有IP地址均从私有地址块 10.0.0.0/8 中分配
- pod 交换机： 10.pod.switch.1
  - pod：pod编号，范围 [0, k-1]
  - switch：交换机在Pod内的位置编号，范围 [0, k-1](从左到右，从下到上)
- core 交换机：10.k.j.i
  - j 和 i：表示该核心交换机在 (k/2)^2 网格中的坐标，范围 [1, k/2]（从左上开始）
- 主机： 10.pod.switch.ID
  - pod 和 switch：与其所连接的边缘交换机一致
  - ID：主机在该子网内的位置，范围 [2, k/2+1]（从左到右）
  - 每个 edge 交换机负责一个 /24 的子网

Outline

Architecture：3.3 Two-level routing table

两级路由表

- 设计目标：实现流量均匀分布
  - 引入两级前缀查找机制，以解决此前提出的流量均匀分发问题

Prefix	Output port
10.2.0.0/24	0
10.2.1.0/24	1
0.0.0.0/0	

→

Suffix	Output port
0.0.0.2/8	2
0.0.0.3/8	3

- 主路由表 (第一级)
  - 存放传统的前缀条目（左匹配，掩码格式为 1^m 0^(32-m)）
  - 一个条目如果是终止前缀 (Terminating)，则直接指向输出端口
  - 一个条目如果是非终止前缀 (Non-terminating)，则指向一个小的二级后缀表
- 后缀表 (第二级)
  - 存放后缀条目（右匹配，掩码格式为 0^(32-m) 1^m）
  - 可被多个一级前缀条目共享
  - 查找时，先进行最长前缀匹配，若命中非终止前缀，则在其指向的二级表中进行最长后缀匹配，最终决定端口

Architecture：3.3 Two-level routing table

两级路由表

- 表规模极小
  - 任何Pod交换机中的路由表最多仅包含 k/2 个前缀 和 k/2 个后缀
  - 这种小规模设计是可行的关键
- 对性能的影响轻微
  - 两级查找结构会略微增加路由表查找延迟
  - 但由于硬件中前缀查找的并行性，以及表格体积很小，性能损失非常有限
- 优势： 在几乎可以忽略的性能开销下，实现了对流量的细粒度、均匀的扩散。

Prefix	Output port
10.2.0.0/24	0
10.2.1.0/24	1
0.0.0.0/0	

Suffix	Output port
0.0.0.2/8	2
0.0.0.3/8	3

- 两级转发表示例：这是位于交换机 10.2.2.1 的转发表。目标 IP 地址为 10.2.1.2 的数据包在端口 1 上转发，而目标 IP 地址为 10.3.0.3 的数据包在端口 3 上转发。

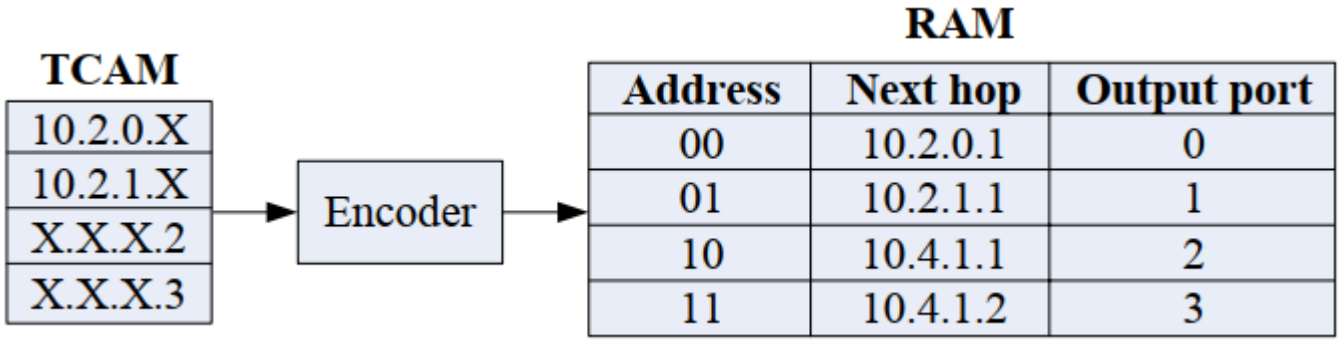
Outline

Architecture：3.4 Two-Level Lookup Implementation

两级查找的硬件实现： TCAM

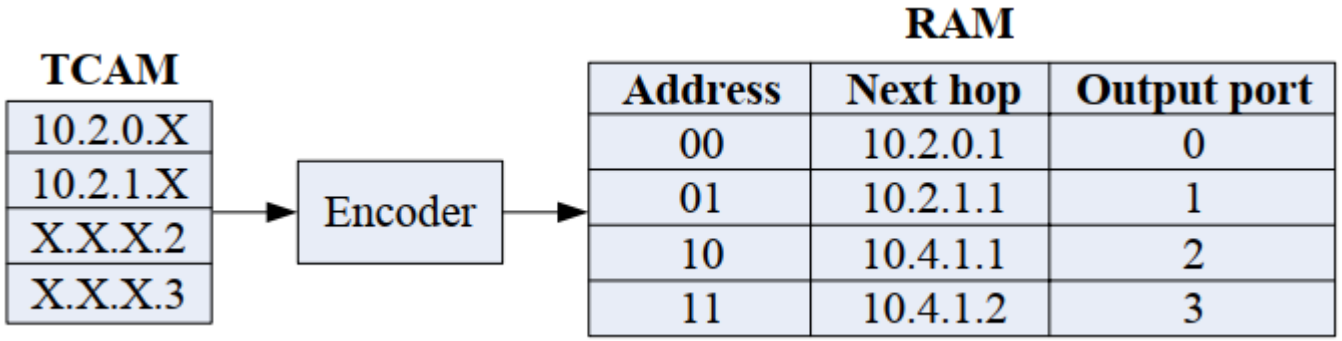
- 核心硬件：三态内容可寻址存储器 (TCAM)
  - 用于执行高速、并行的搜索密集型应用
  - 工作原理： 可在单个时钟周期内将其内部所有条目与目标位模式进行并行匹配
  - 关键特性： 除0和1外，还能存储 “不在乎” 位 (X)，因此非常适合存储路由表中可变长度的前缀/后缀
- TCAM的优缺点
  - 优点： 查找速度极快，远快于基于软件的算法方法
  - 缺点： 存储密度低、功耗大、单位比特成本昂贵
  - 价值： 得益于所设计的极小的路由表规模（仅需约 k个条目），可以采用成本可控的TCAM来实现

Architecture：3.4 Two-Level Lookup Implementation



- 硬件架构设计
  - TCAM：存储所有的前缀和后缀规则
  - RAM：被TCAM的匹配结果索引，存储下一跳IP地址和输出端口号
- 编码规则：
  - 将左匹配（前缀） 条目存放在数值较小的TCAM地址中
  - 将右匹配（后缀） 条目存放在数值较大的TCAM地址中
  - CAM的输出经过编码，确保总是选择数值最小的匹配地址对应的条目。这保证了前缀匹配优先于后缀匹配的语义

Architecture：3.4 Two-Level Lookup Implementation



- 参考上图举例
- 例一：目的 IP 10.2.0.3
  - 同时匹配前缀 10.2.0.X(Port 0) 和后缀 X.X.X.3
  - 根据“最小地址优先”规则，选择前缀表项，报文从端口 0 转发
- 例二：目的 IP 10.3.1.2
  - 仅匹配后缀 X.X.X.2(Port 2)
  - 报文从端口 2 转发

Outline

Architecture：3.5 Routing Algorithm

Fat-tree 路由机制

- 核心目标：实现流量均匀扩散
  - 利用两级路由表（**前缀+后缀**）在Pod出口将流量均匀分散到核心层
- 路径确定性
  - 相同源目的主机的流量走固定路径，避免报文乱序
- 集中式路由控制
  - 假设一个中央控制器掌握全拓扑，静态生成所有路由表
  - 简化设计，无需复杂分布式协议（故障处理除外）

Architecture：3.5 Routing Algorithm

Fat-tree 路由角色与策略

- Pod内交换机（过滤与扩散）
  - 下层交换机： 负责将流量均匀上传至Pod内上层交换机
  - 上层交换机： 拥有Pod内所有子网的终止前缀，负责Pod内路由
  - pod间流量： 匹配默认 /0前缀，并进入二级后缀表（根据目的IP主机ID字节）选择出口，实现均匀扩散
- 核心交换机（简单指引）
  - **仅包含**指向各 pod 的**终止 /16前缀**（如 10.pod.0.0/16）
  - 功能简单：将报文指引到正确的目标Pod

```
1 foreach j in [1, (k/2)] do
2     foreach i in [1, (k/2)] do
3         foreach destination pod x in [0, (k/2) − 1] do
4             addPrefix(10.k.j.i,10.x.0.0/16, x);
5         end
6     end
7 end
```

Algorithm 2: Generating core switch routing tables.

Architecture：3.5 Routing Algorithm

Fat-tree路由表生成 - Pod交换机

- 核心任务：
  - 为**本Pod内**的所有子网添加**终止前缀**（直接路由）
  - 为所有**Pod间**流量添加一条默认 /0前缀，并关联一个**二级后缀表**
- 二级表关键：
  - 后缀表匹配目的IP的最低位字节**（主机ID）**
  - 通过 (主机ID mod k/2)等计算映射到出口端口，确保来自不同源机的、前往相同主机ID的流量也能被分散
- 表规模极小： 最多 k/2个一级前缀和 k/2个二级后缀表项

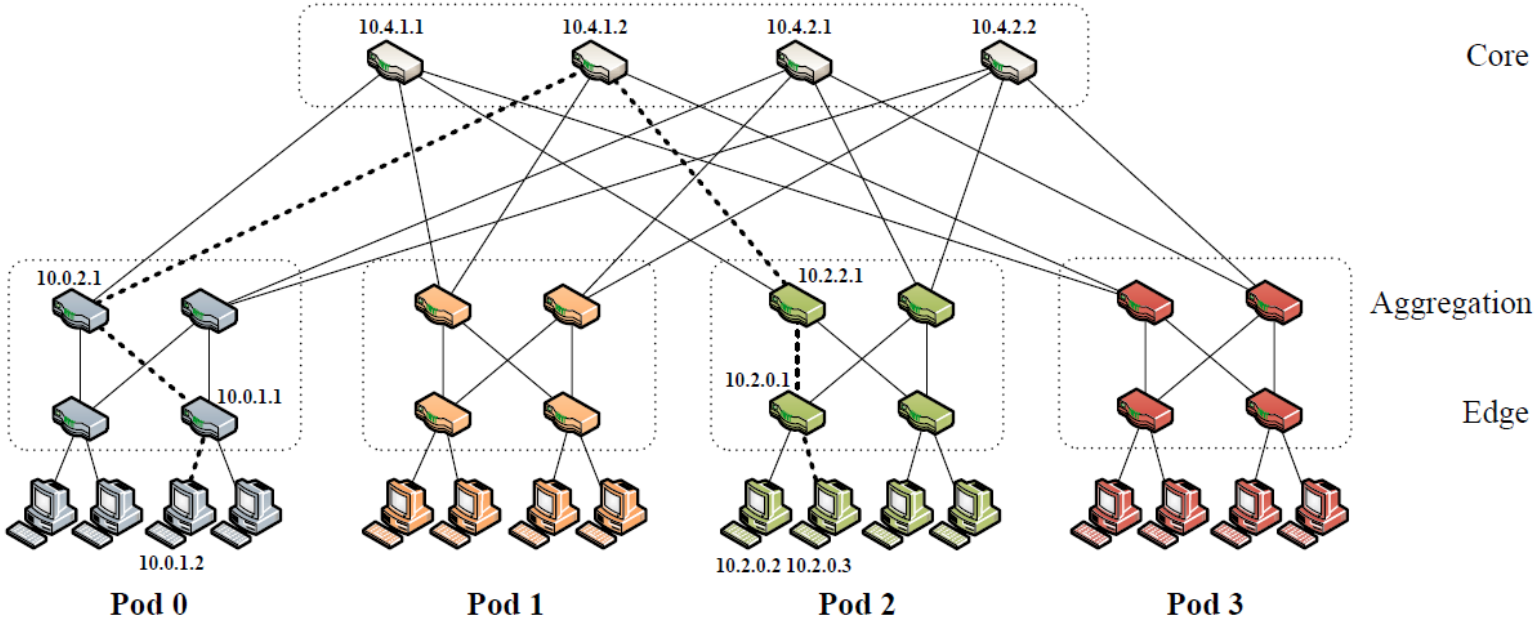
Architecture：3.5 Routing Algorithm

Fat-tree路由表生成 - Pod交换机

```
1  foreach pod x in [0, k − 1] do
2      foreach switch z in [(k/2), k − 1] do
3          foreach subnet i in [0, (k/2) − 1] do
4              addPrefix(10.x.z.1, 10.x.i.0/24, i);
5          end
6          addPrefix(10.x.z.1, 0.0.0.0/0, 0);
7          foreach host ID i in [2, (k/2) + 1] do
8              addSuffix(10.x.z.1, 0.0.0.i/8,
               (i − 2 + z)mod(k/2) + (k/2));
9          end
10     end
11 end
```

算法：生成 aggregation switch 的 routing table;
对于 edge 层交换机，z in [0, (*k*/2)-1], 然后忽略 line 3-5

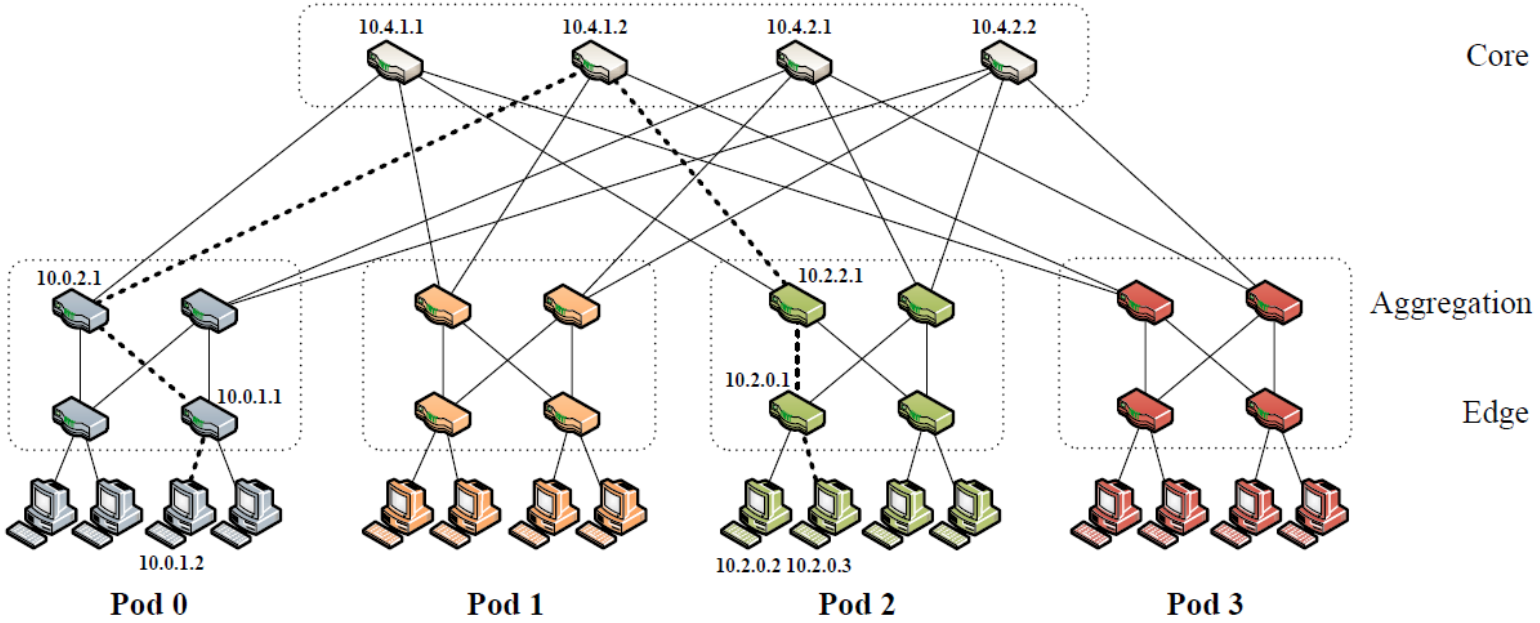
Architecture：3.5 Routing Algorithm



Fat-tree 路由举例 源： 10.0.1.2，目的： 10.2.0.3

- 1. 源网关交换机 (10.0.1.1)
  - 匹配 /0-> 查二级表。
  - 后缀 X.X.X.3匹配 -> 转发至端口2 (往上层交换机 10.0.2.1)
- 2. Pod内上层交换机 (10.0.2.1)
  - 同样操作，根据主机ID 3转发至端口3 (往核心交换机 10.4.1.1)

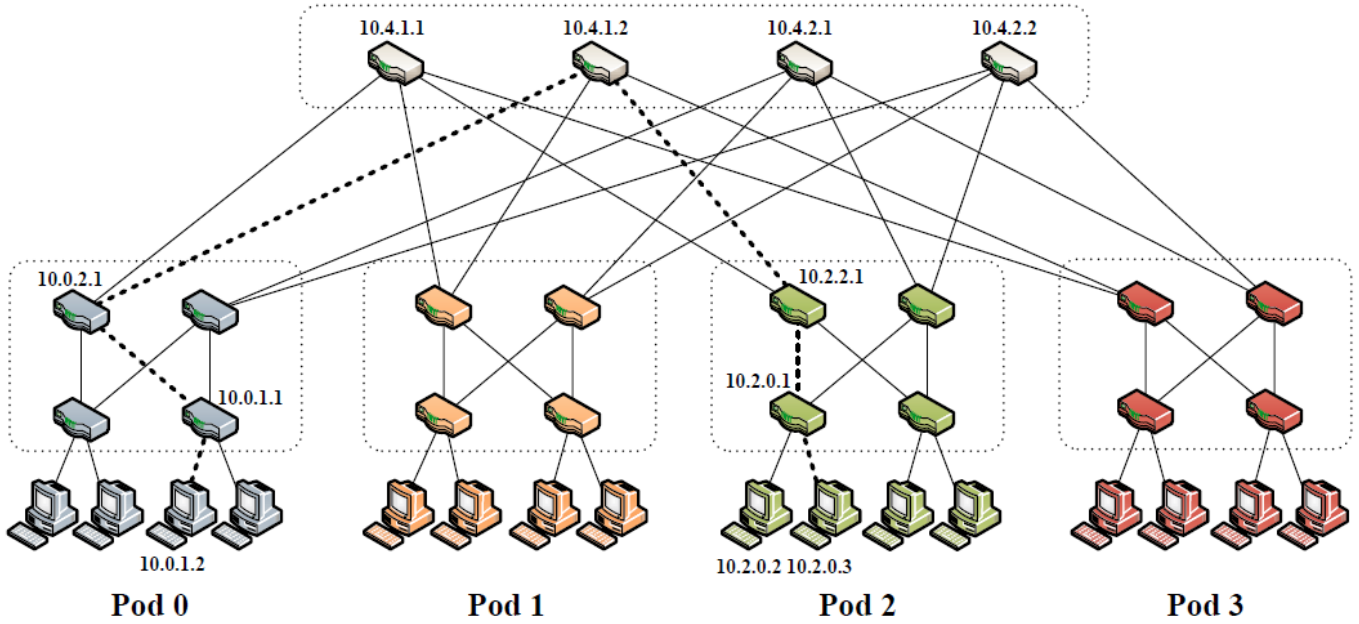
Architecture：3.5 Routing Algorithm



Fat-tree 路由举例 源： 10.0.1.2，目的： 10.2.0.3

- 3. 核心交换机 (10.4.1.1)
  - 匹配终止前缀 10.2.0.0/16-> 转发至端口2 (通往Pod 2)。
- 4. 目的Pod上层交换机 (10.2.2.1)
  - 匹配终止前缀 10.2.0.0/24-> 转发至端口0 (往目的子网交换机 10.2.0.1)。
- 5. 目的子网交换机 (10.2.0.1): 标准交换，送达主机

Architecture：3.5 Routing Algorithm



Ag

```
6    addPrefix(10.x.z.1, 0.0.0.0/0, 0);
7    foreach host ID i in [2, (k/2) + 1] do
8        addSuffix(10.x.z.1, 0.0.0.i/8,
             (i − 2 + z)mod(k/2) + (k/2));
```

Fat-tree 路由举例　思考：若目的IP变为 10.2.0.2 呢?

- 1. 在网关交换机(10.0.1.1)上，匹配 port 3，下一跳 (10.0.3.1).
  - port = (2-2+1) mod 2 +2 = 3
- 2. 在网关交换机(10.0.3.1)上，匹配 port 3，下一跳 (10.4.2.2).
  - port = (2-2+3) mod 2 +2 = 3
- 3. 接下来三跳上都会匹配中路由表中的终止前缀

Outline

I. Introduction
II. Background
III. Architecture
    3.1 Motivation
    3.2 Addressing
    3.3 Two-Level Routing Table
    3.4 Two-Level Lookup Implementation
    3.5 Routing Algorithm
    3.6 Flow Classification
    3.7 Flow Scheduling
    3.8 Fault Tolerance
IV. Implementation and evaluation
V. Review

Architecture：3.6 Flow Classification

动态流分类机制

- 设计定位：可选的高级功能
  - 这是在之前所述两级静态路由基础上，考虑增加的两种可选动态路由技术之一
  - 目标：为量化其潜在优势，但需承认它会带来额外的逐包处理开销
- 核心特性：软状态与回退
  - 此类方案维护的均是软状态（Soft State）
  - 若状态丢失，单个交换机可以无缝回退到基础的二级路由策略，保证网络连通性
- 流（Flow）的定义
  - 拥有相同包头字段子集（通常为：源IP、目的IP、目的传输端口）的一个报文序列

Architecture：3.6 Flow Classification

动态流分类机制： 工作机制（在 pod 交换机中实现）

- 流识别与保持
  - 识别出属于同一流的后续报文，并将其从相同的输出端口转发。
  - 目的： 避免同一流的报文出现乱序。
- 动态端口重分配
  - 周期性地重新分配最少数量的流的输出端口。
  - 目的： 最小化不同端口的聚合流容量之间的差异，解决可避免的局部拥塞（如多个流争抢同一端口，而等价路径却利用不足）。
- 核心价值
  - 在动态变化的流大小背景下，确保向上端口的流量能获得公平的分布，从而提升整体带宽利用效率。

Outline

Architecture：3.7 Flow Scheduling

流量调度

- 流量特征与设计动机
  - 网络流量具有长尾分布特征：少数大流量、长生命周期的流占据了大部分带宽，而大量小流则占小部分。
  - 核心观点： 对大流的路由决策至关重要，直接影响网络可实现的二分带宽，因此值得特殊处理。
- edge 交换机
  - 初始分配： 为新流分配至负载最轻的端口。
  - 大流检测与上报： 检测出口流量是否超过预定阈值，并周期性向中央调度器上报所有活跃大流的源和目的信息。
  - 关键限制： 与之前的流分类方案不同，边缘交换机无权自行重分配大流端口，必须等待中央调度器的指令。

Architecture：3.7 Flow Scheduling

流量调度

- 中央调度器的核心功能
  - 全局视角： （可冗余部署）追踪所有活跃大流，掌握全网所有链路的“是否可承载大流”bool 状态
  - 路径分配： 为目标主机对（Pod间流量）线性搜索可用的核心交换机路径，寻找一条无冲突（链路未被预留） 的路径
  - 资源管理： 找到路径后，标记相关链路为“已预留”，并通知源Pod的相关交换机调整该流的输出端口
  - 垃圾回收： 清理超时未更新的流状态，释放其链路预留
- 工作流程特点
  - 非阻塞操作： 边缘交换机在上报后不阻塞等待，而是先按普通流处理，直至收到调度器的重分配指令
  - intra-pod流量： 对Pod内大流，调度器 similarly 搜索 Pod 内上层交换机的无冲突路径

Outline

Architecture：3.8 Fault Tolerance

容错机制

- 设计基础：路径冗余与快速检测
  - Fat-tree 拓扑中任意两台主机间的多路径特性天然支持容错
  - 每个交换机通过与邻居建立 BFD会话，快速检测链路或相邻交换机故障
- Pod内链路故障（下层<->上层交换机）影响三类流量：
  - 1、发出的流量： 本地流分类器将故障链路成本设为“无穷大”，不再分配新流，并选择其他可用上层交换机
  - 2、Pod内中转流量： 受影响的上层交换机广播标签，通知Pod内所有其他下层交换机避开该故障链路
  - 3、传入的Pod间流量： 受影响的上层交换机向所有核心交换机广播标签，宣告其无法到达故障链路所连子网。核心交换机再将此标签镜像给其他Pod的上层交换机，使其在为该子网分配新流时避开受影响的核心交换机

Architecture：3.8 Fault Tolerance

容错机制

- 上行链路故障（上层交换机<->核心交换机）影响两类流量：
  - 1、发出的Pod间流量： 本地路由表将故障链路标记为不可用，并本地选择另一台核心交换机
  - 2、传入的Pod间流量： 受影响的核心交换机向所有直连的上层交换机广播标签，宣告其无法到达整个目标 pod。这些上层交换机在为该Pod分配流时避开此核心交换机
- 故障恢复与统一调度
  - 恢复： 当故障链路/交换机恢复并重建 BFD 会话后，上述过程被逆向操作以取消其影响
  - 与大流调度器的协同： 中央调度器可轻易将故障链路标记为“繁忙/不可用”，从而在为大流选择路径时自动绕开故障点

Outline

I. Introduction

II. Background

III. Architecture

IV. Implementation and Evaluation

V. Review

4 Implementation

原型实现与验证

- 双平台实现策略
- NetFPGA原型：基于其IPv4路由器和TCAM进行修改
  - 结果：仅增加不到100行代码，未引入任何可测量的额外查找延迟，证明方案可嵌入现有交换机
- Click软件路由器原型（评估重点）：
  - Click是一种模块化的软件路由器架构，通过连接称为“元素”的报文处理模块来实现复杂功能 https://github.com/kohler/click
- 核心元素一：TwoLevelTable
  - 功能： 实现了两级路由表查找逻辑（先最长前缀匹配，若非终止则进行最长后缀匹配）
  - 集成： 可直接替换Click标准IP路由器示例中的核心路由表元素

4 Implementation

原型实现与验证

- 核心元素二：FlowClassifier (动态流分类器)
  - 功能： 基于源/目的IP进行流分类，保证同一流报文从同一端口发出避免乱序。
  - 优化目标： 周期性（如每秒）尝试重分配最多3个流的端口，以最小化各输出端口总流容量的差异
  - 定位： 作为两级表扩散机制的替代方案，是一种软状态的性能优化
  - 特点： 具有偶尔的 disruptive（可能造成报文重排序），但能自适应动态变化的流大小
- 核心元素三：FlowScheduler (大流调度器)
  - FlowReporter (边缘交换机)： 检测大流并上报给中央调度器
  - FlowScheduler (中央)： 维护全网链路二进制状态（空闲/占用），为收到的大流请求线性搜索一条无冲突的路径，并通知相关交换机
  - 复杂度： 空间复杂度 O(k^3)，时间复杂度 O(k^2)，对于 k=48的典型值，是可管理的

Evaluation: 5.1 Experiment Description

实验平台

- 测试网络拓扑：4端口 fat-tree (k=4)
  - 规模： 共 16台主机、20台交换机
  - 在此规模下，交换机数量多于主机；但更大规模集群中，主机数将远超交换机
- 物理平台配置
  - 硬件： 将36个逻辑元素（交换机+主机）复用到10台物理服务器上，通过一台48端口千兆以太网交换机互联
- 关键测试参数
  - 链路带宽限制： 将所有虚拟链路带宽限制为 96 Mbit/s，确保实验瓶颈在于网络而非CPU。
  - 流量生成： 每台主机均以 96 Mbit/s 的恒定速率生成出口流量。
- 性能度量指标
  - 测量每台主机的入口流量速率。
  - 在所有双射通信映射下，所有主机的最小聚合入口流量即为网络的有效二分带宽。

Evaluation: 5.2 Benchmark Suite

基准测试流量模式

- 核心限制： 所有通信映射均为一对一（1-to-1），即每台主机只从一个源接收流量
- 测试策略：
  - 随机 (Random)：主机以均匀概率向网络中任何其他主机发送流量
  - 步长 (Stride(i))：索引为 x 的主机向索引为 (x + i) mod 16 的主机发送流量
  - 交错概率 (Staggered Prob)：主机以 SubnetP 概率向同子网主机发送，以 PodP 概率向同 Pod 主机发送，以 1-SubnetP-PodP 概率向任何主机发送

Evaluation: 5.2 Benchmark Suite

基准测试流量模式

- 关键 worst-case 场景：
  - Pod内涌入 (Inter-pod Incoming)：
  - 场景： 多个Pod向同一目标Pod的不同主机发送流量，且恰都选择同一核心交换机
  - 问题： 导致该核心交换机到目标Pod的链路超额订阅
  - 最差比例：(k - 1) : 1
- 同ID发出 (Same-ID Outgoing)：
  - 场景： 同一子网内的主机向网络中不同目标发送流量，但这些目标主机拥有相同的主机ID
  - 问题： 静态路由技术迫使这些流使用同一个上行端口，造成出口拥塞
  - 最差比例：(k/2) : 1
  - 注： 此场景是 FlowClassifier 最能发挥性能提升作用的用例

Evaluation: 5.3 Results

性能评估结果

- 核心结论：
  - 传统树形结构： 在所有 Pod 间通信模式中性能最差，受限于核心链路饱和，仅能达到~28%的理想二分带宽
  - 两级路由表： 性能显著提升，在随机通信模式下达到约75%的理想带宽
    - 局限性： 静态性导致其在 同ID发出 和 Pod内涌入 场景下性能下降（如后者降至50%）
  - 流分类器 (FlowClassifier)： 凭借动态分配，在所有情况下均优于前两者，最差情况下也能保持约75%的带宽
    - 局限性： 因仅有本地视图，无法避免上游决策引发的核心层拥塞
- 全局最优方案：流调度器 (FlowScheduler)
  - 凭借全局知识，为大数据流分配无冲突的路径
  - 在随机通信映射下实现了 93% 的理想二分带宽，全面 outperforms 所有其他方案

Evaluation: 5.3 Results

关键可行性证明：中央调度器开销极低 (数据来源：表3)

- 测试平台： 配备适中的商用PC (2.33GHz)
- 规模： 支持 27,648台主机 (k=48) 的大型网络
- 内存： 总内存需求仅为 5.6 MB
- 速度： 处理单个流安置请求的平均时间 小于 0.8 毫秒
- 结论： Fat-tree 拓扑的规律性极大简化了路径搜索问题，使得中央调度方案在资源开销极低的前提下即可实现，具备极高的实际可行性

Test	Tree	Two-Level Table	Flow Classification	Flow Scheduling
Random	53.4%	75.0%	76.3%	93.5%
Stride (1)	100.0%	100.0%	100.0%	100.0%
Stride (2)	78.1%	100.0%	100.0%	99.5%
Stride (4)	27.9%	100.0%	100.0%	100.0%
Stride (8)	28.0%	100.0%	100.0%	99.9%
Staggered Prob (1.0, 0.0)	100.0%	100.0%	100.0%	100.0%
Staggered Prob (0.5, 0.3)	83.6%	82.0%	86.2%	93.4%
Staggered Prob (0.2, 0.3)	64.9%	75.6%	80.2%	88.5%
Worst cases:				
Inter-pod Incoming	28.0%	50.6%	75.1%	99.9%
Same-ID Outgoing	27.8%	38.5%	75.4%	87.4%

Outline

I. Introduction

II. Background

III. Architecture

IV. Implementation and Evaluation

V. Review

Review

- 什么是数据中心网络？数据中心网络需要具备哪些理想特性？
- 什么是传统的三层数据中心网络拓扑，其局限性是什么？
- 胖树（Fat-tree）与传统设计有何不同？具体体现在：
  - 拓扑结构（Topology）
  - 寻址方案（Addressing）
  - 路由表设计（Routing Table）
  - 路由算法（Routing）

Review

Fat-tree拓扑优势总结：

- **可扩展的全网聚合带宽**：数据中心内的任意一台主机都能以其本地网卡的全速带宽与网络中的任何其他主机进行通信
- **卓越的成本效益**：允许整个网络架构都使用廉价的、现成的商用以太网交换机来构建
- **高效的多路径路由与负载均衡**：拓扑天生提供了大量的等价路径，同时可以将数据流在多个核心交换机之间均匀地分散开

Review

Fat-tree拓扑优势总结：

- **良好的向后兼容性**：该架构不需要对终端主机的网络接口、操作系统或应用程序做任何修改，完全向后兼容以太网、IP协议和TCP协议
- **优越的功耗与散热表现**：在一个支持约27,000台主机的网络中，胖树架构的功耗比传统设计降低了56.6%，散热量降低了56.5%
- **固有的容错能力**：任意两个主机之间存在的大量冗余路径使得胖树拓扑在面对链路或交换机故障时具有天然的优势和吸引力