云计算导引

张胜
南京大学
2019.11.18

云计算@Wiki

- 一种计算能力，提供了计算资源与底层结构之间的抽象，使用户可以通过网络方便的，按需使用的来对一个共享的资源池进行迅速的配置，部署与使用，并且只需要很少的管理以及与服务商的交互。

云计算@IBM

- 云计算可以用来描述平台与应用。一个云计算平台可以动态按需来供应和配置服务器，云中的服务器可以使用物理机或者虚拟机，云中也可以包括其他的计算资源如存储域网络（SANs）、网络设施、防火墙，以及其他的安全设备。

狭义云计算与广义云计算

- 狭义云计算——**提供资源的网络**称为“云”
  - “云”中的资源在使用者看来是可以无限扩展的
  - 随时获取，按需使用，随时扩展，按使用付费
    - 像水电一样使用IT基础设施

- 广义云计算——**任意服务构成的资源池**称为“云”
  - “云”是一些可以自我维护和管理的虚拟计算资源
    - 大型服务器集群，包括计算服务器、存储服务器、宽带资源
  - 云计算将所有的计算资源集中起来
  - 应用提供者无需关注细节，更专注于业务

相关技术

- 并行计算(Parallel Computing)
- 分布式计算(Distributed Computing)
- 网格计算(Grid Computing)

网格计算

- 利用互联网把地理上广泛分布的各种资源（包括计算资源、存储资源、带宽资源、软件资源、数据资源、信息资源、知识资源等）连成一个逻辑整体，就像一台超级计算机一样，为用户提供一体化信息和应用服务（计算、存储、访问等）。
- 网格计算是分布式计算的一种，是分布式计算封装。
- 云计算可以认为是网格计算的商业演化模式。

云计算概念模型

用户

服务
提供商

云端

云计算与云平台

- 云计算是一种计算模式
  - 不是一种产品……
- “按需服务”:Pay as you go
  - 云计算的核心理念
  - 类似于水、电等基础设施
- 云平台是实现云计算模式的产品
  - 云计算解决方案

云服务分类

软件即服务 SaaS（Software as a Service）	Salesfoce online CRM服务
平台即服务 PaaS（Platform as a Service）	Google App Engine Sina App Engine (SAE)
基础设施即服务 IaaS（Infrastructure as a Service）	Amazon EC2、S3 阿里云

基础设施即服务（IaaS）

- **IaaS —— Infrastructure as a Service：** 为IT行业创造虚拟的计算和数据中心，使得其能够把计算单元、存储器、I/O设备、带宽等计算机基础设施，集中起来成为一个虚拟的资源池来为整个网络提供服务。

- 按使用量付费
- Amazon WebServices，简作AWS
  - 弹性计算云EC2（Elastic Compute Cloud）——计算
  - 简单存储服务S3（Simple Storage Service）——存储

平台即服务（PaaS）

- **PaaS —— Platform as a Service：**把服务器平台或开发环境作为一种服务提供的商业模式。

- 从系统定制到PaaS

- Google App Engine、SAE
- Hadoop、Spark等大数据处理平台

软件即服务（SaaS）

- **SaaS —— Software as a Service：** 一种基于互联网提供软件服务的应用模式。

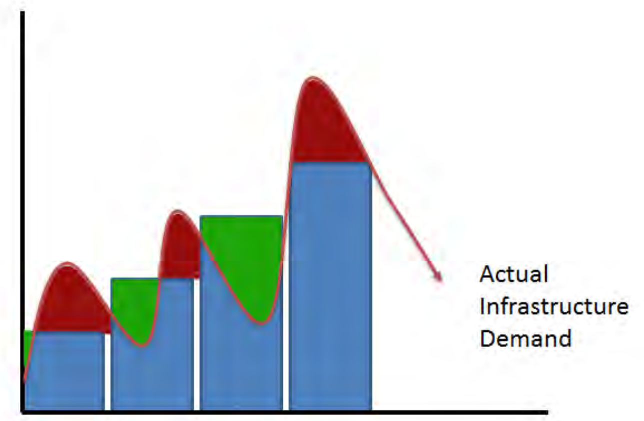
- 软件租赁：用户按使用时间和使用规模付费
- 绿色部署：用户不需安装，打开浏览器即可运行
- 不需要额外的服务器硬件
- 软件（应用服务）按需定制

云计算特点

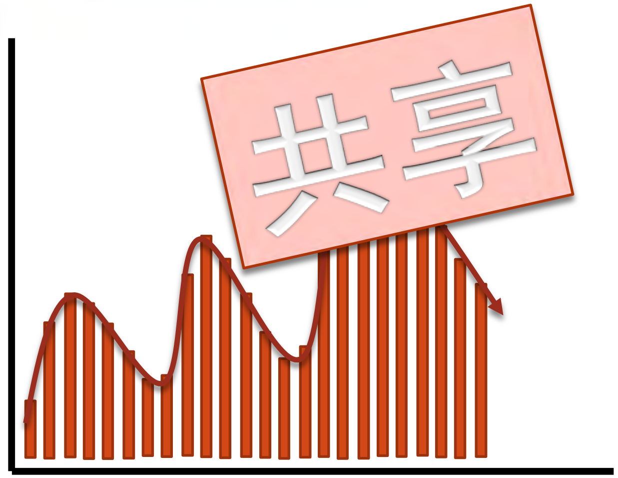
- 高可靠性：冗余副本、负载均衡
- 通用性：　　　　　　支撑千变万化的实际应用
- 按需服务：按需购买
- 安全：　　　　　　　摆脱数据丢失、病毒入侵
- 方便：　　　　　　　支持多终端、数据共享

“按需服务”

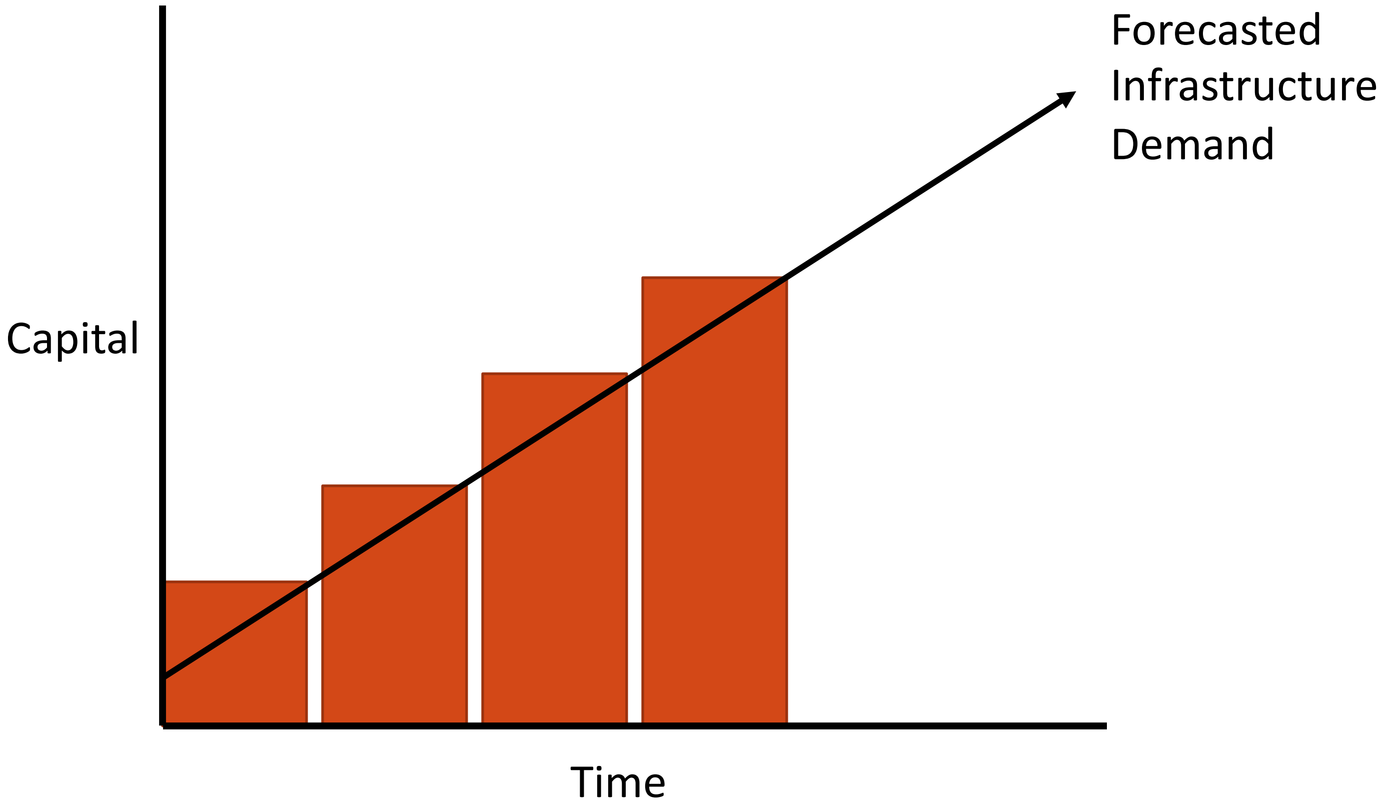
- 需求动态性
  - 资源数量
  - 资源类型
    - 软件/硬件
  - 工作负载



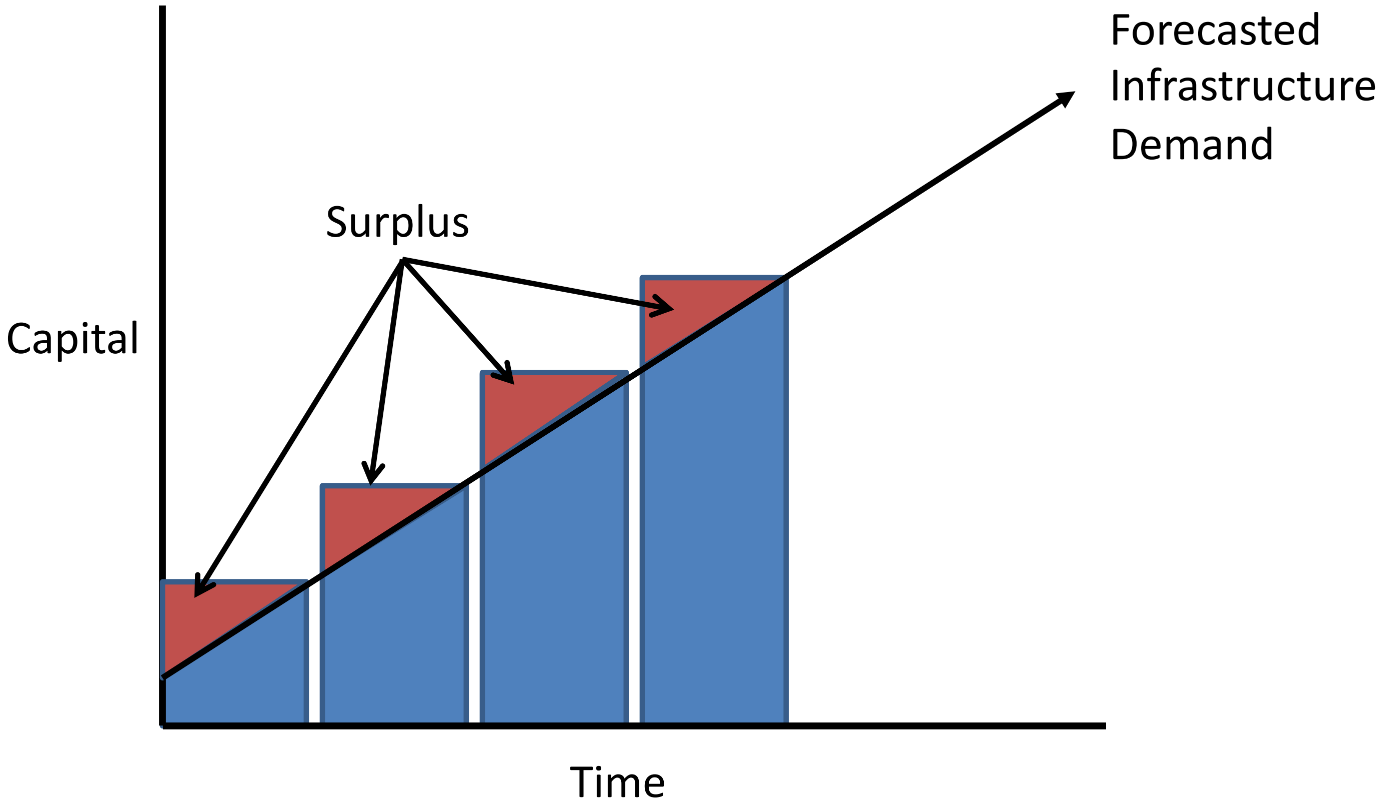
- 高效获取
  - 便捷
  - 低价



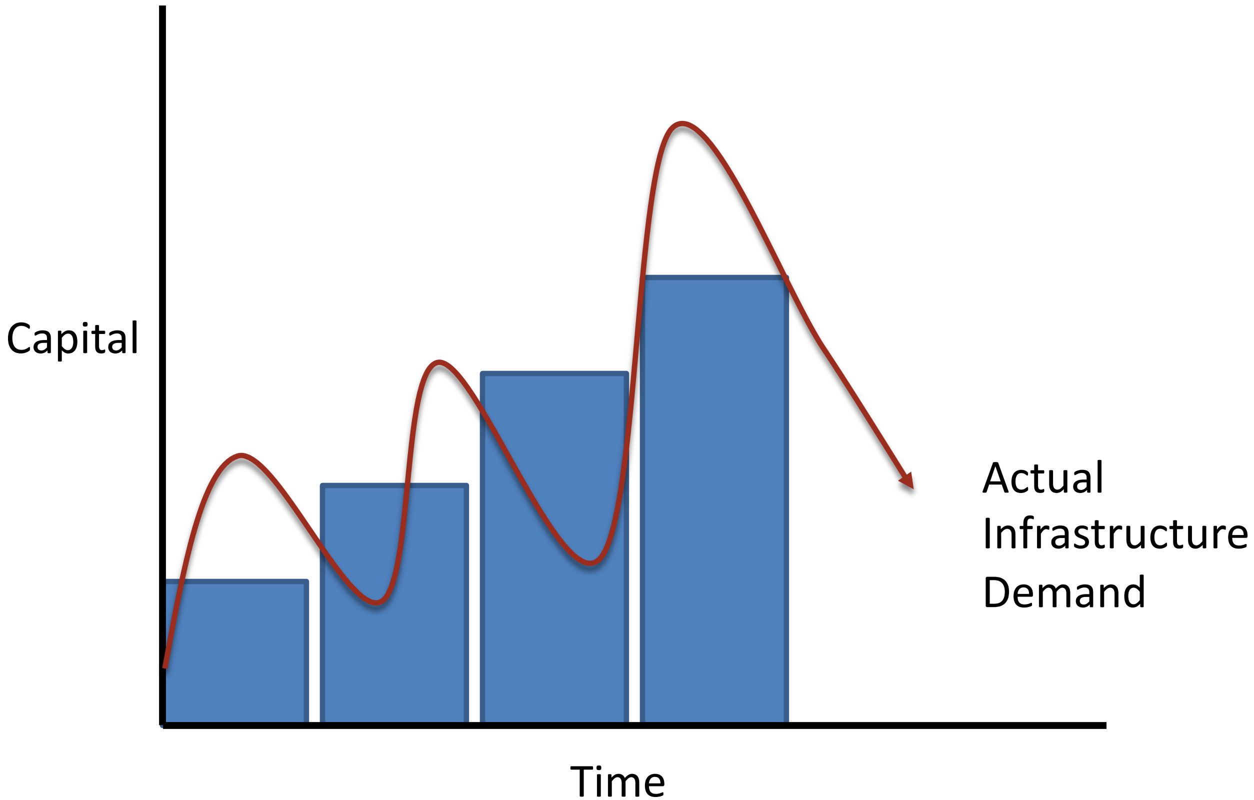
Traditional Infrastructure Model



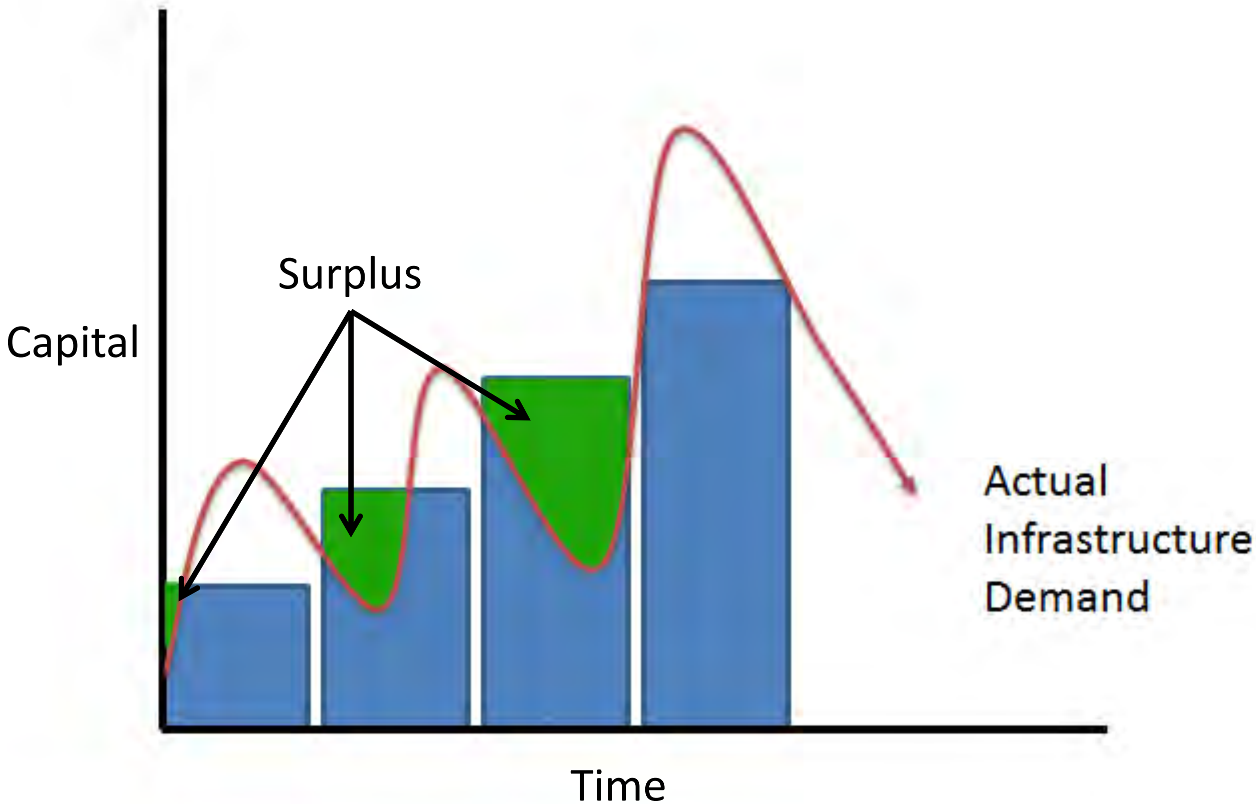
Acceptable Surplus



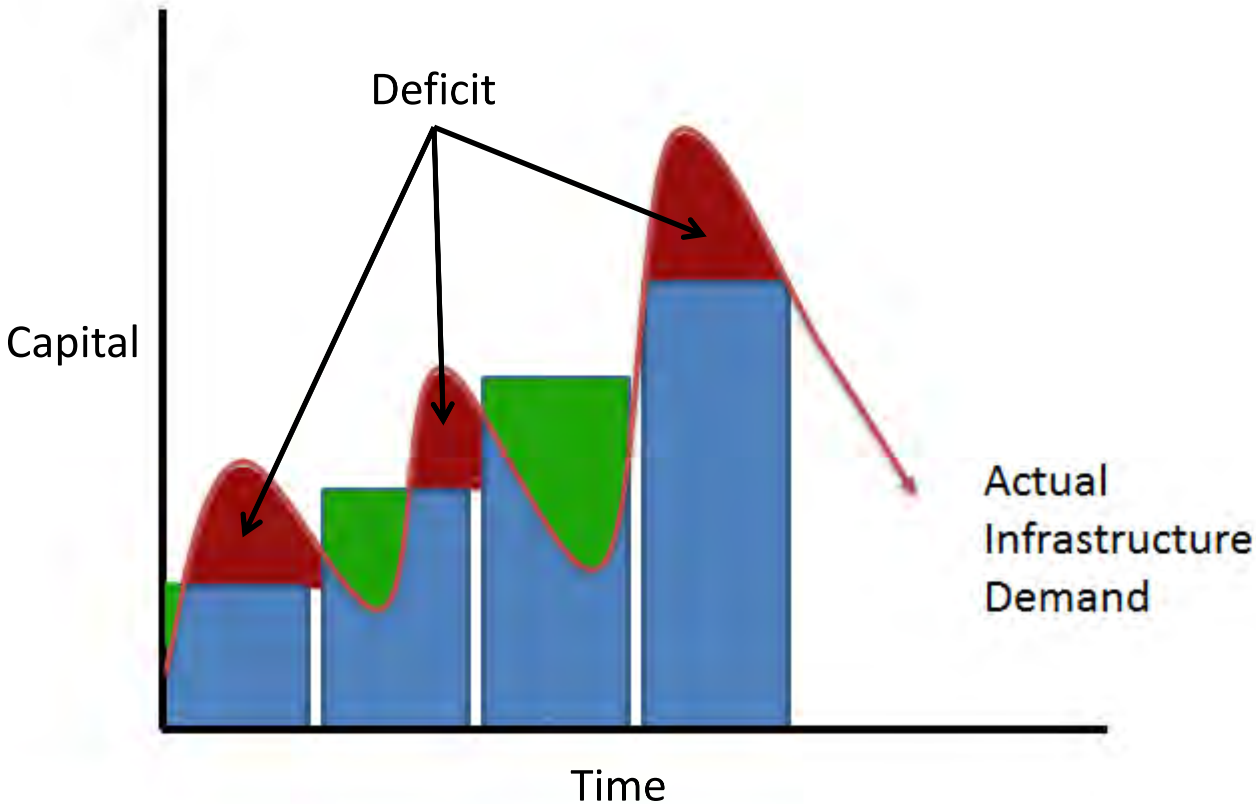
Actual Infrastructure Demand



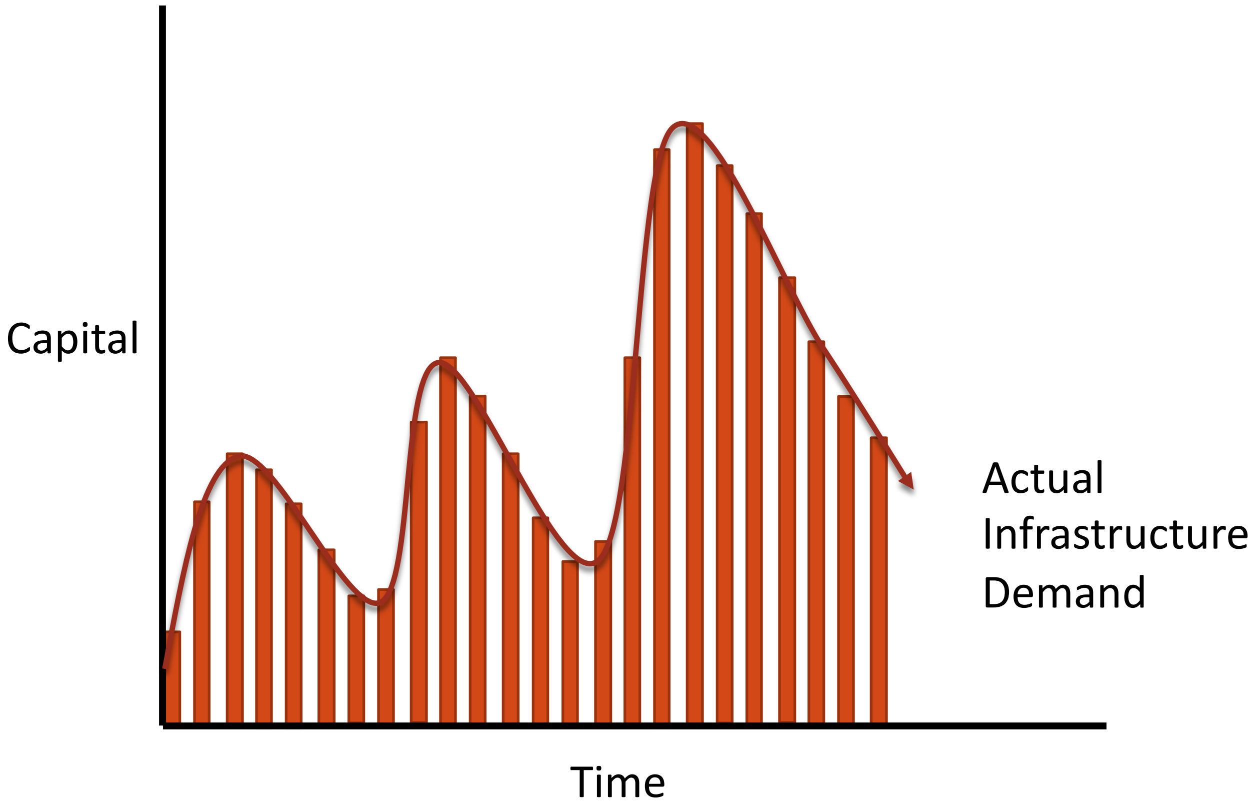
Unacceptable Surplus



Unacceptable Deficit



What We May Expect It to Be



云计算模式

- 公有云
  - 资源以“按需服务”的方式提供给公用服务；服务以一种效用计算的方式被出租使用。
- 私有云
  - 为一个客户单独使用而构建的，因而提供对数据、安全性和服务质量的最有效控制。
- 混合云
  - 安全因素，并非所有的企业信息都能放置在公有云上。

公有云

- 云服务提供商
  - 有效管理内在资源，提高利用率，节省能源
- 终端用户
  - 按需使用，租用计算、存储和服务资源
- 企业用户/服务提供商
  - 创业前期成本大大降低
    - IT 硬件投入低
    - 按需租用
  - 采取自助服务和按使用量付费的使用模式，迅速获得计算资源，无需为配置过大的资源容量而过度投资

私有云 面向大中型企事业单位

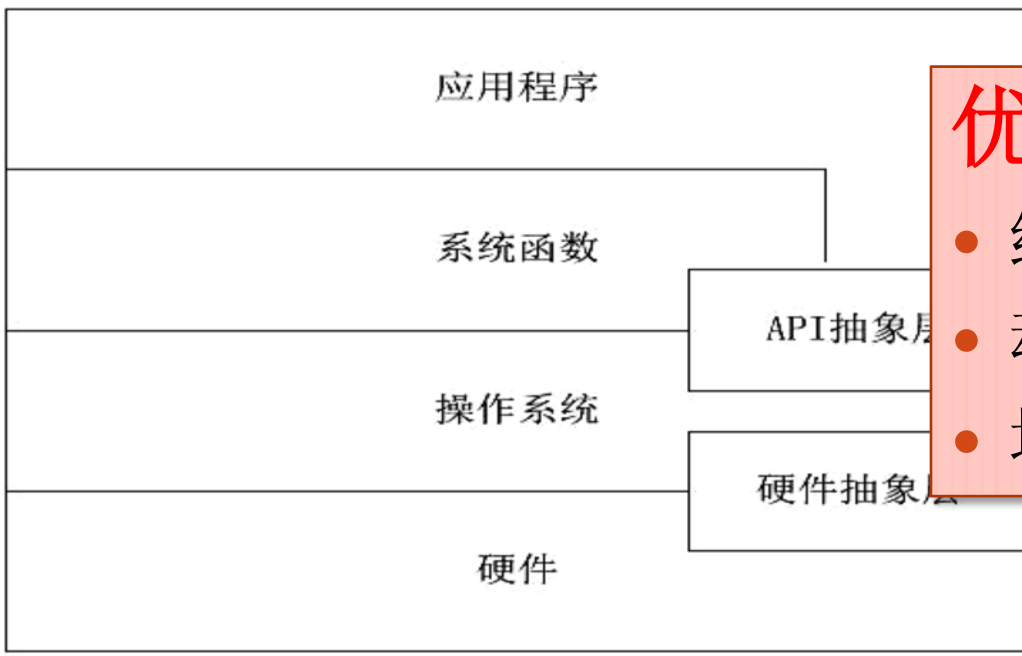
- 高效资源管理
  - 快速申请、回收IT资源；
  - 分时共享提升资源利用率。
- 提升数据安全
  - 云端集中式数据备份
  - 云端控制数据的流动
- 降低软件成本
  - 专业化、集中式系统管理，有效防止病毒等恶意软件。
  - 专业软件的共享。

虚拟化技术

- 虚拟化与云计算
- 虚拟化关键技术
- 虚拟化数据中心

虚拟化简述

- 虚拟化是由位于下层的软件模块，将其封装或抽象，提供一个物理或软件的接口，使得上层的软件可以直接运行在这个虚拟的环境，和运行在原来的环境一样。

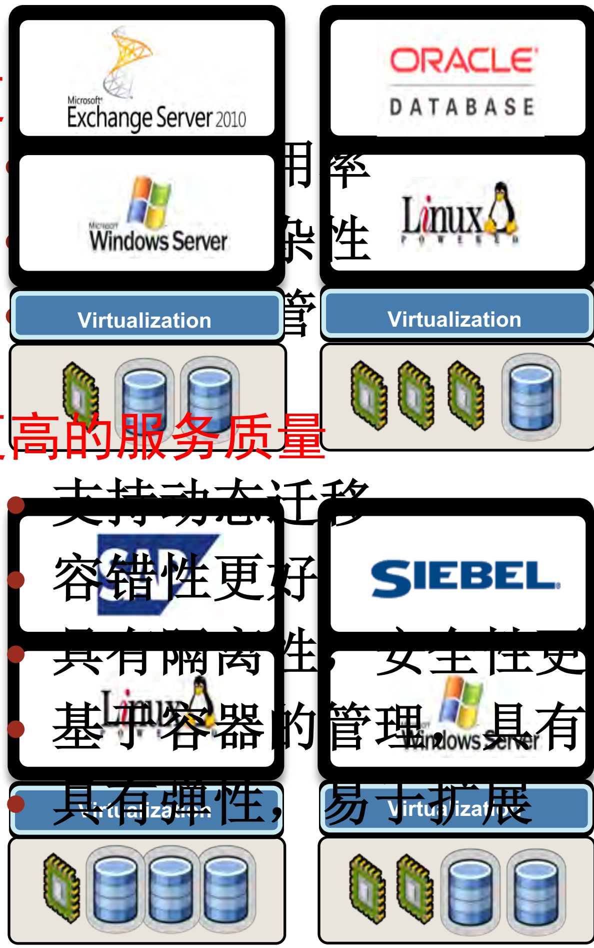


优势

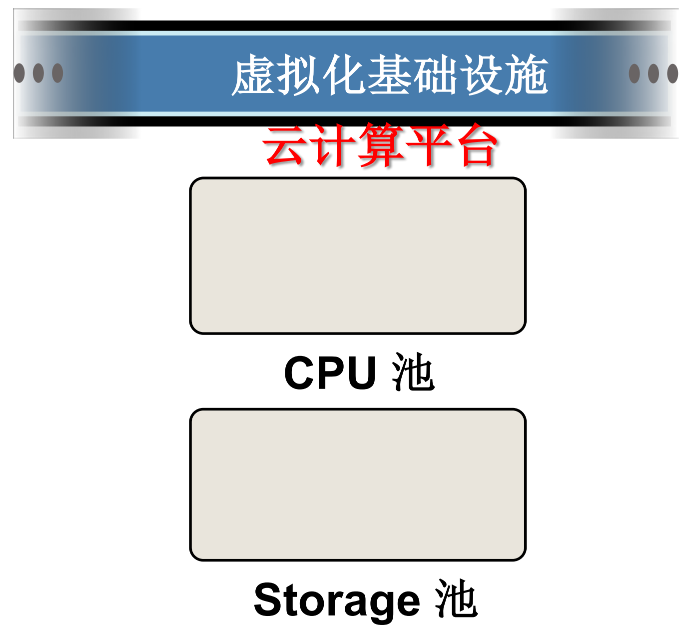
- 细粒度资源分配
- 动态迁移
- 增加资源利用率

虚拟化技术

- 更
  - 用率
  - 杂性
  - 管



- 更高的服务质量
  - 支持动态迁移
  - 容错性更好
  - 具有隔离性，安全性更好
  - 基于容器的管理，具有封装性
  - 具有弹性，易于扩展



虚拟化与云计算

虚拟化特点	为云计算带来的好处
封装与隔离	保证每个用户有安全可信的工作环境
多实例	保证较高的资源利用率 为服务器合并提供基础
硬件无关性	整合异构硬件资源 可实现虚拟机迁移，使资源调度、负载平衡容易实现
特权功能	入侵检测和病毒检测
动态调整资源	细粒度的可扩展性

VMware

- 较早的商业化公司之一，2003年被EMC公司收购，产品目前有3个系列：WorkStation、GSX和ESX。
- 优点
  - 易用性好
  - 完全模拟一台服务器，客户操作系统不作修改就能使用
  - 服务器运行在Windows、Linux和mac上
  - 客户机支持Windows/Linux/FreeBSD/Solaris等
  - ESX不需要任何操作系统，性能相当高
- 缺点
  - 收费

Xen

- 基于 Linux的开源项目，现支持类虚拟化和硬件虚拟化技术，2005年初成了 xensource 公司，专注于Xen产品的开发和推广，目前有Intel、AMD、HP、IBM、Redhat和SuSE 等厂商支持。
- 优点
  - 性能损失很小
  - 支持原生操作系统和打过内核补丁的操作系统
- 缺点
  - 服务器只能运行于Linux
  - 若使用类虚拟化技术，运行于其上的虚拟机需打内核补丁，且不支持未开源的操作系统（如Windows）

KVM

- Kernel-based Virtual Machine，开源的系统虚拟化模块
- Linux 2.6.20之后集成在Linux的各个主要发行版本中
  - 使用Linux自身的调度器进行管理，所以相对于Xen，其核心源码很少。
  - 已成为学术界的主流虚拟机监控器之一。
- KVM的虚拟化需要硬件支持，是基于硬件的完全虚拟化。

数据中心与虚拟化

- 数据中心为云计算的实现提供了基本的计算和存储资源，是支持云计算的重要基础。

数据中心与虚拟化

- 数据中心为云计算的实现提供了基本的计算和存储资源，是支持云计算的重要基础。
- 企业级数据中心的发展趋势是具备高度的灵活性和适应性。
  - 能根据外部需求做出快速变化
  - 虚拟化技术是比较好的解决方法
- 企业关注投入产出率（Return of Investment），严格的控制预算和成本。
  - “绿色”
  - “低碳”

云计算平台

- Google云平台
- Amzon云平台
- Openstack

云平台

- 用户：提供使用云计算服务的入口
  - 按需使用各种类型的服务
- 服务商：以最小代价满足用户请求
  - 提供各种层次和类型的服务，并提供可信性保证
  - 通过资源的合理配置和整合最小化代价

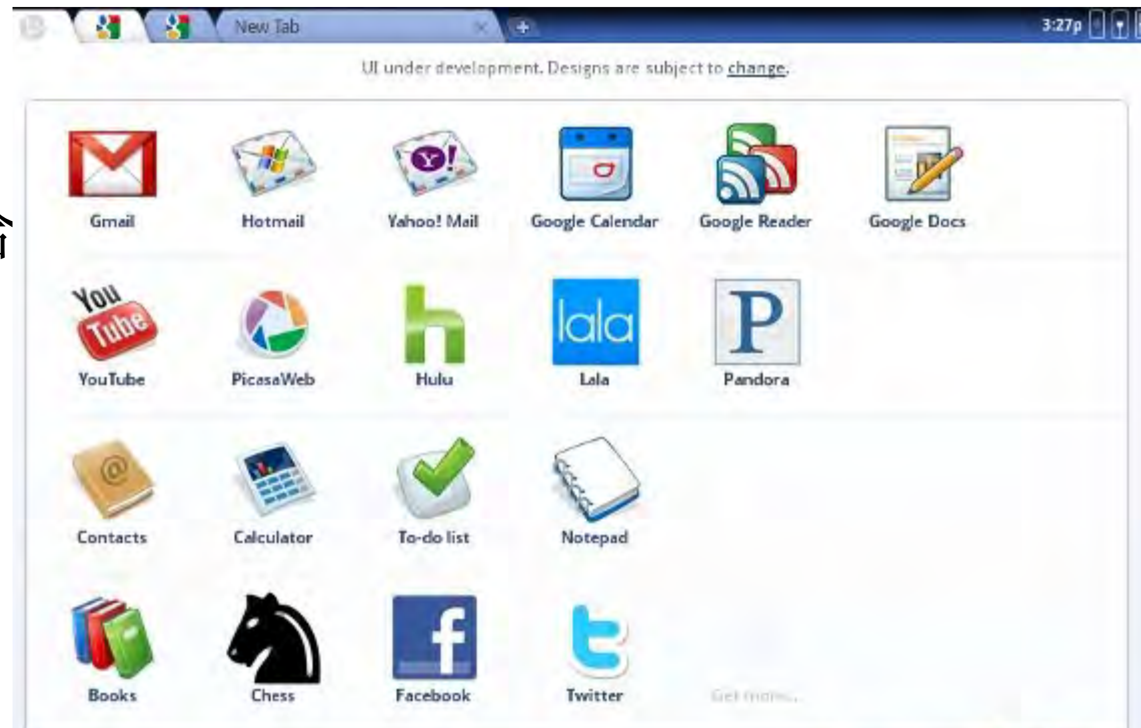
Google App Engine

- App Engine让用户在Google的基础架构上运行自定义网络应用程序
- App Engine可免费使用
  - 500MB持久存储空间
  - 支持每月500万页面浏览量的CPU和带宽
- App Engine提供使用Python和Java语言的运行环境，可用其建立Web站点等网络应用。

Google Chrome OS

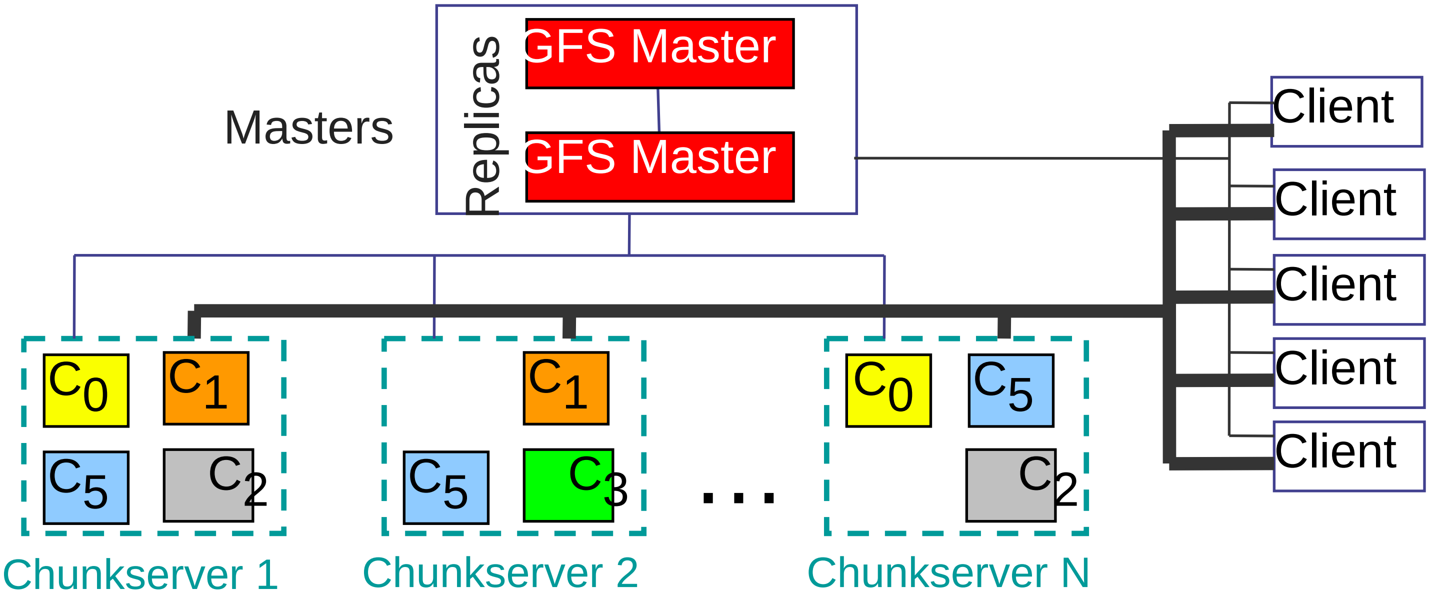
- Google Chrome OS是一个为上网本设计的轻量级开源操作系统。

- 快速开/关机
- 浏览器紧密结合
- 网络紧密相连



Google文件系统（GFS）

- GFS是一个可扩展的分布式文件系统，用于大型的、分布式的、对大量数据进行访问的应用。它运行于廉价的普通硬件上，但可以提供容错功能。它可以给大量的 用户提供总体性能较高的服务。



MapReduce

- MapReduce是一种编程模型，用于大规模数据集（大于1TB）的并行运算。概念“Map（映射）”和“Reduce（化简）”，和他们的主要思想，都是从函数式编程语言里借来的，还有从矢量编程语言里借来的特性。他极大地方便了编程人员在不会分布式并 行编程的情况下，将自己的程序运行在分布式系统上。
- 当前的软件实现是指定一个Map（映射）函数，用来把一组键值对映射成一组新的键值对，指定并发的Reduce（化简）函数，用来保证所有映射的键值对中的每一个共享相同的键组。

其他云平台

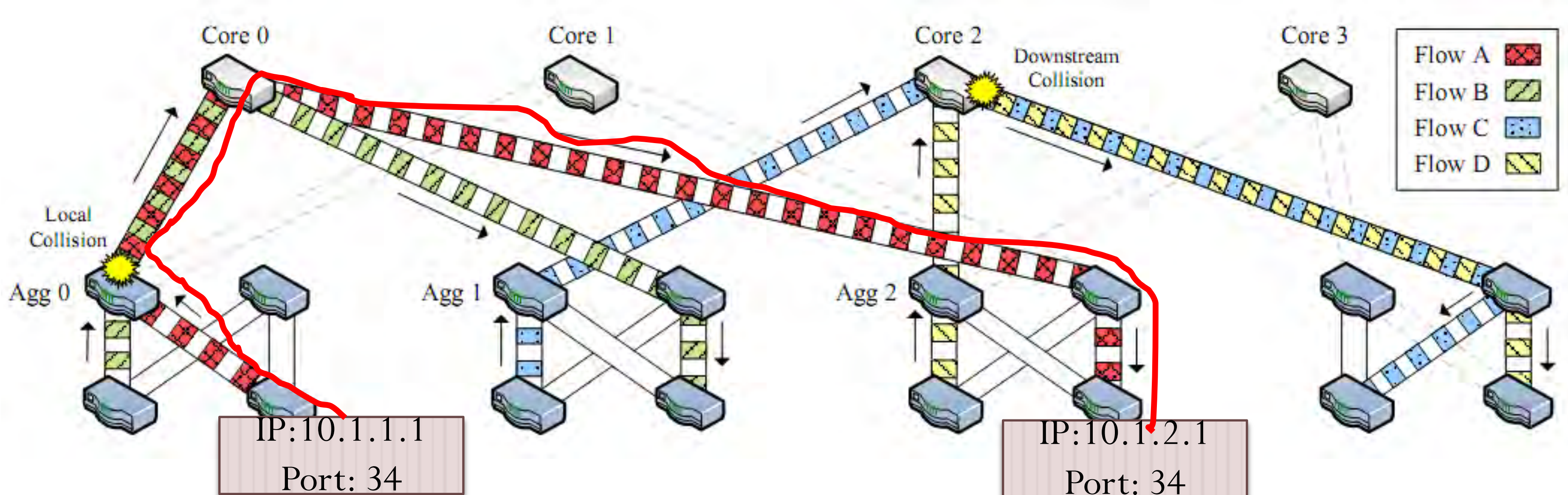
- 微软Windows Azure系统
  - SaaS产品： Dynamics CRM Online、Exchange Online、Office Communications Online及SharePoint Online。
- IBM“蓝云”
  - 对企业现有的基础架构进行整合，通过虚拟化技术和自动化技术，构建企业自己拥有的云计算中心。
- Adobe AIR平台
  - 构建丰富的Internet应用并部署为桌面应用
- Force.com平台
  - PaaS产品： Salesforce

OpenStack

- OpenStack是IaaS(基础设施即服务)组件，让任何人都可以自行建立和提供云端运算服务。
- 此外，OpenStack也用作建立防火墙内的“私有云”（Private Cloud），提供机构或企业内各部门共享资源。

研究问题1：Flow Scheduling in Data Centers

Flows



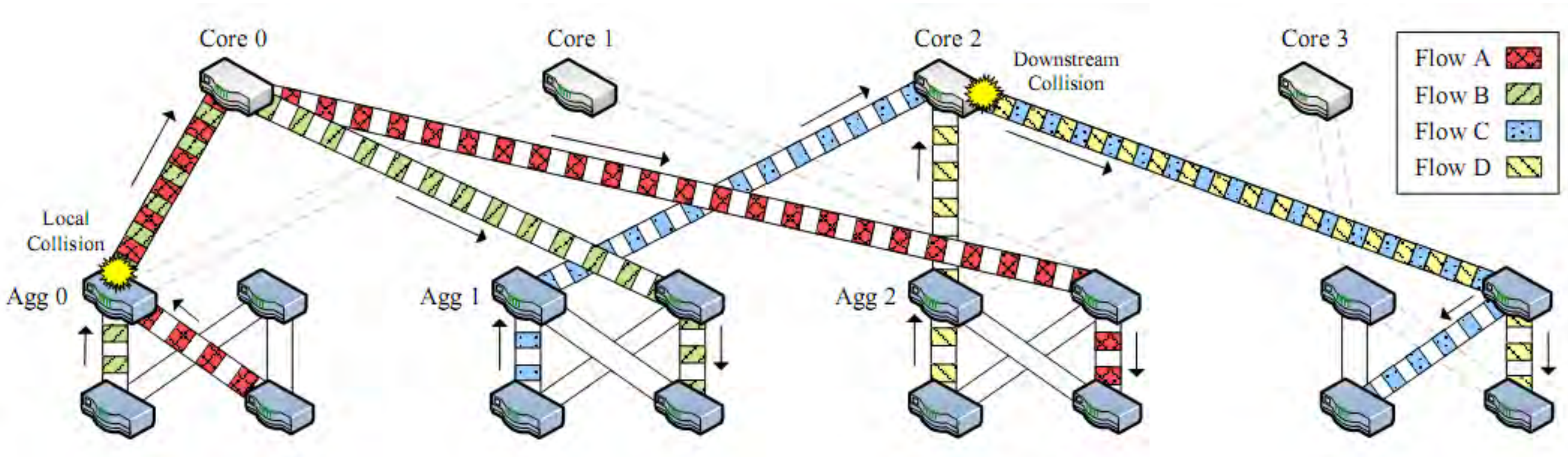
- Virtualized Server
  - Multiple VMs coexist atop a shared PM.
- Application/Service
  - Communications among VMs over the DC network.
- Flow: 5-tuple
  - (src IP, src port, transport protocol, dest IP, dest port)

Flow Scheduling

- Why?
  - Highly congested links/switches & under-utilized links/switches
- How?
  - Change the current (usually “bad”) flow-to-link mapping through moving flows

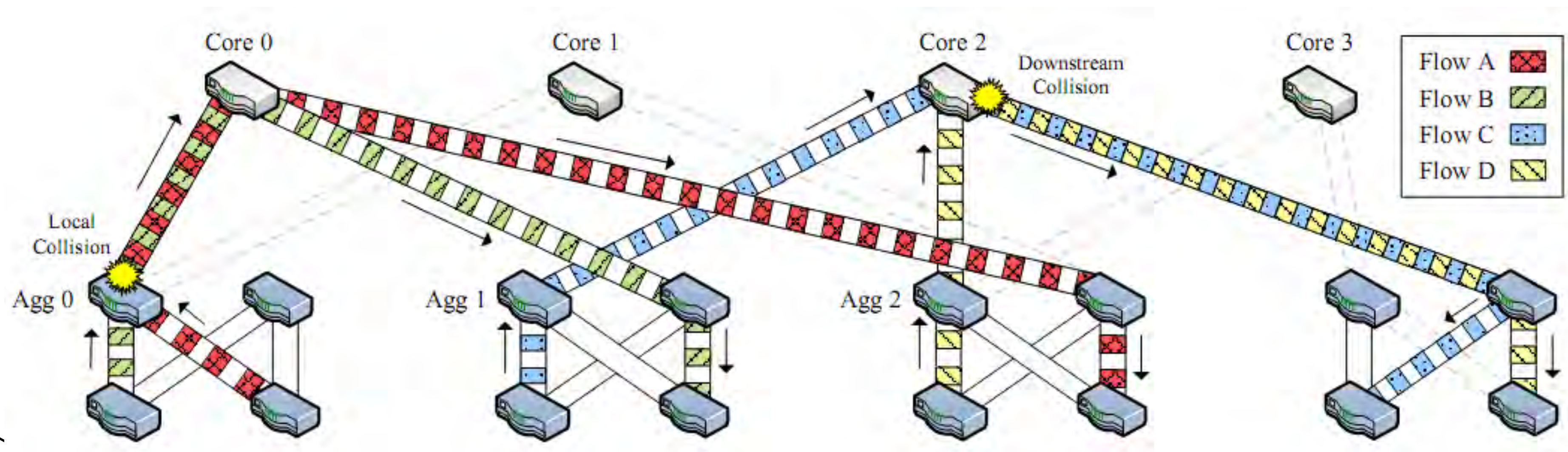
Per-packet vs. Per-flow

- Per-packet: forwarding decisions are made based on the information of individual packet
  - Multipath routing
  - Reordering
- Per-flow: flow-level load balancing
  - E.g., ECMP



Goals

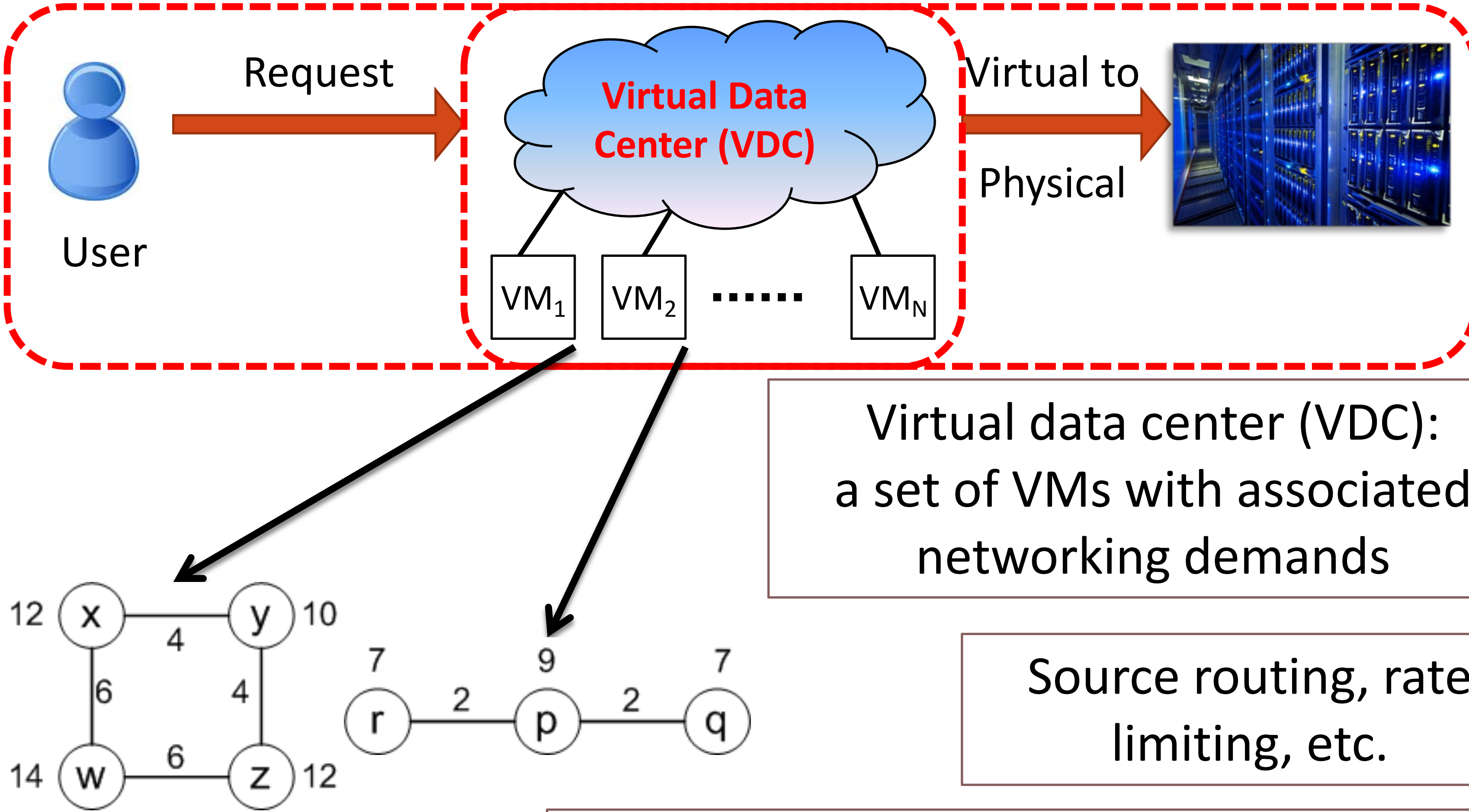
- Predict application performance
- Mitigate congestion
- Maximize throughput
- Meet flow deadlines/shorten flow completion times
- Achieve high network utilization
- …



Content

- Predict application performance
  - [CoNext’08] SecondNet
- Mitigate congestion
  - [NSDI’10] Hedera
- Maximize throughput
  - [SIGCOMM’13] B4
- Minimize flow completion times
  - [Infocom’14] RepFlow

SecondNet

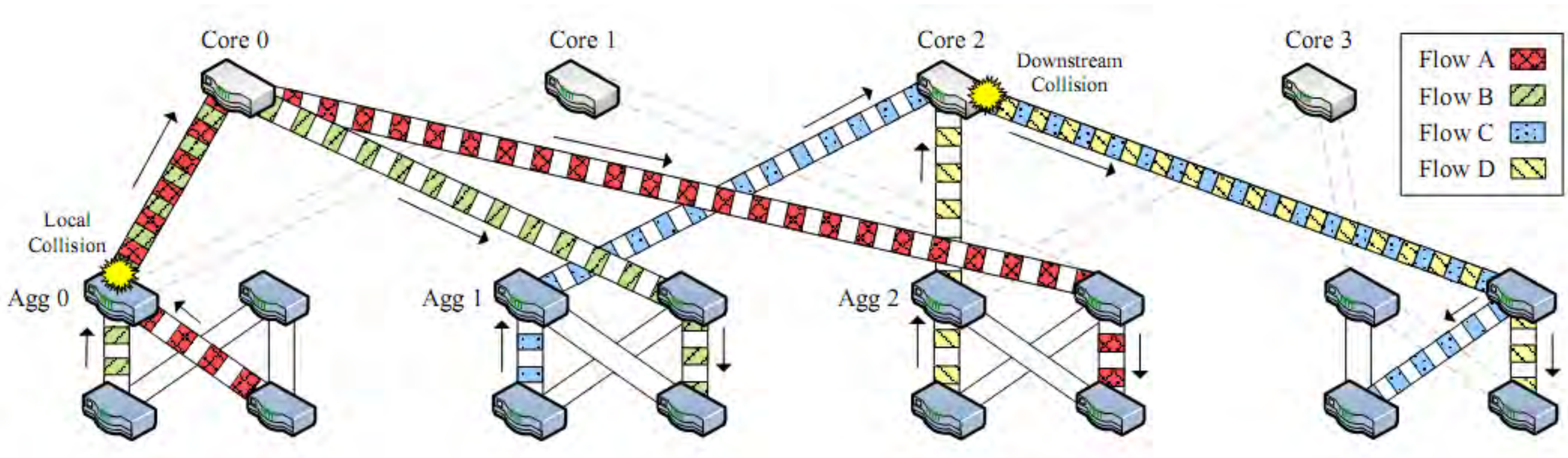


Virtual data center (VDC): a set of VMs with associated networking demands

Source routing, rate limiting, etc.

Not work-conserving; hard to estimate network demands before deployment

Hedera (1/2)



Bottleneck links elongate applications' completions, while some other links are under-utilization.

Hedera (2/2)

- Detect elephant flows
  - # of flows is extremely larg
  - Scalability concern
  - Larger than 10% of host’s NIC or link capacity
- Estimate flow demands
  - Host- vs. network-limited flows
- Schedule flows
  - Global First Fit
  - Simulated Annealing

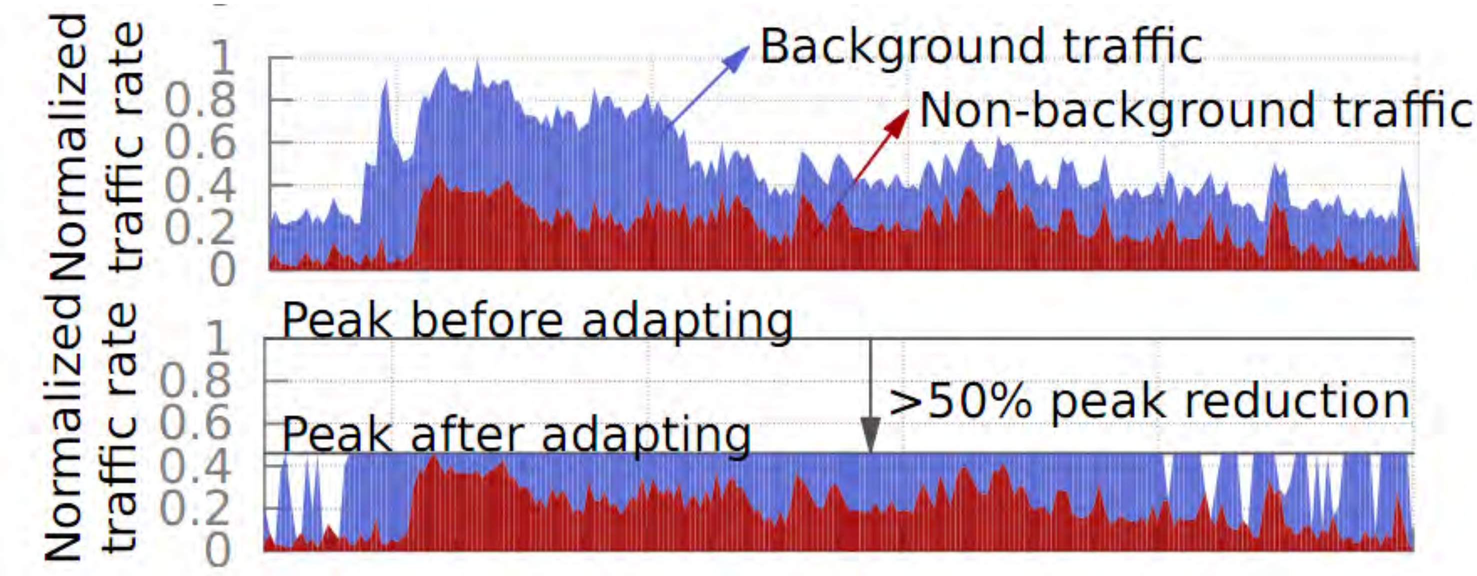
Hedera complements ECMP

B4 (1/2)

- User data copies
  - to remote DC for availability
  - Lowest volume (10% ~ 15%), most latency sensitive, highest priority
- Remote storage access
  - for computation over distribute data sources
- Large-scale data push
  - for synchronizing states across DCs
  - Highest volume, least latency sensitive, lowest priority

B4 (2/2)

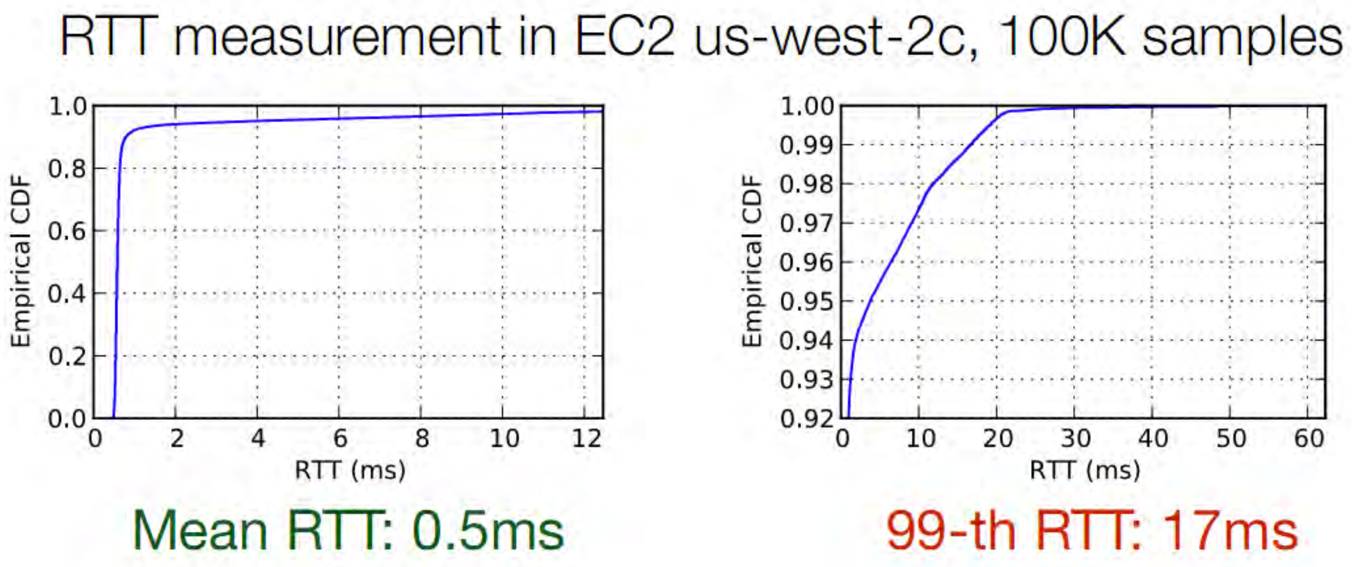
- Key idea: guarantee that highest priority traffic arrive its destination with low delay, and let lowest priority traffic use the remaining resource as much as possible



- Implementation: SDN, TE as overlay (agile deployment, quick failure recovery, easy for test, etc.)

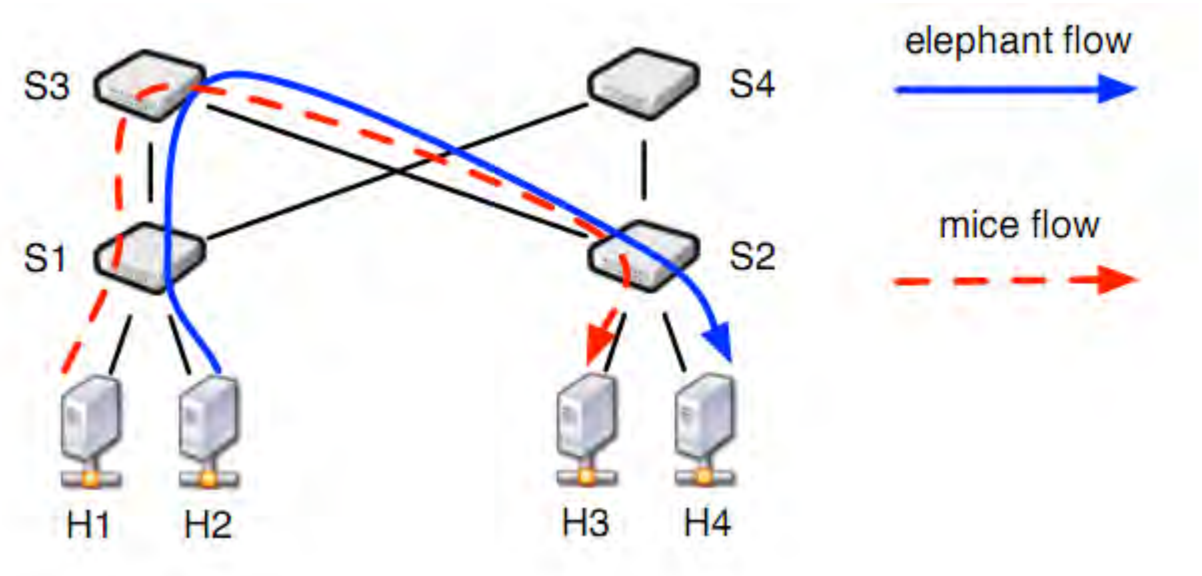
RepFlow: Background

- Short flows are critical for many interactive apps.
  - Delay-sensitive
- Flow completion times (FCT) for short flows are poor in current data centers.
- Goal: reduce FCT for short flows both on average and in the 99th percentile.



RepFlow: Head-of-line Blocking

- Head-of-line blocking in switch ports with ECMP, resulting in long queuing delay.
  - Packets from short flows have to wait in buffer until packets from large flows compete their transmissions on the same egress port.

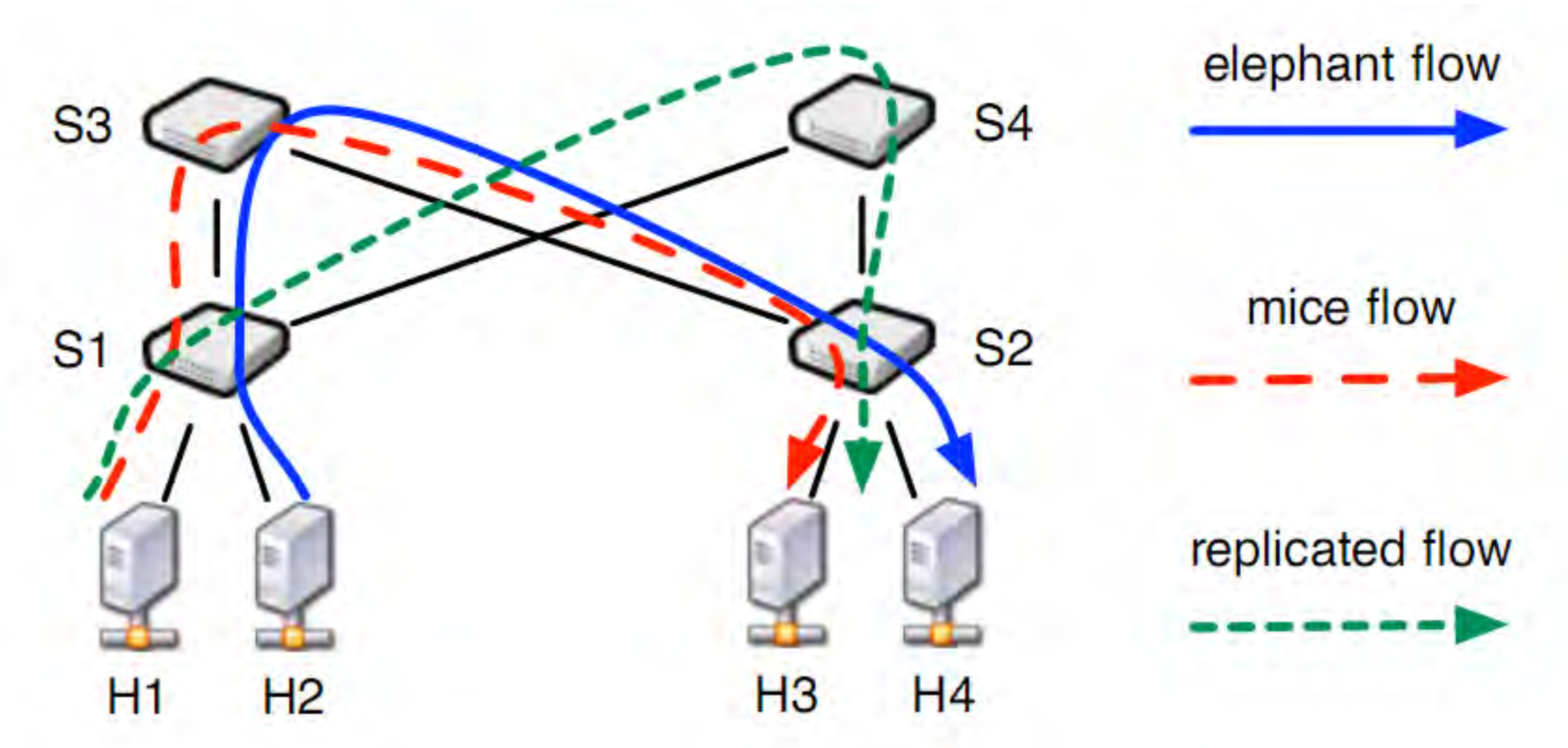


RepFlow: Related Work

- Reducing (tail) FCT in data center networks is an important problem
  - Reduce queue length: DCTCP [SIGCOMM'10]
  - Prioritize flows: D^3 [SIGCOMM'11]
- They all require modifications to end-hosts and/or switches, making it difficult to deploy in reality.

RepFlow: Motivation

Replicate each short flow to exploit path diversity

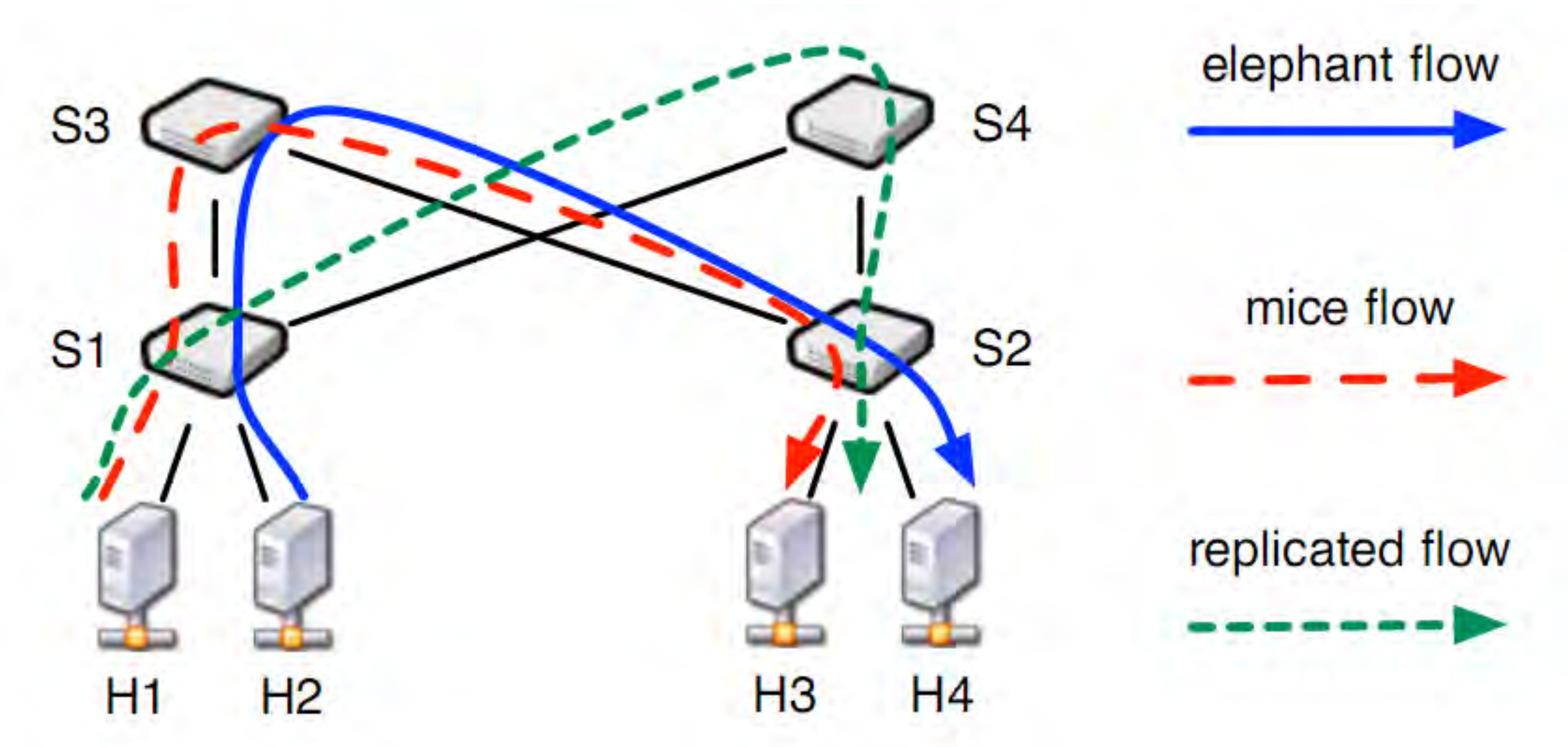


RepFlow: Design

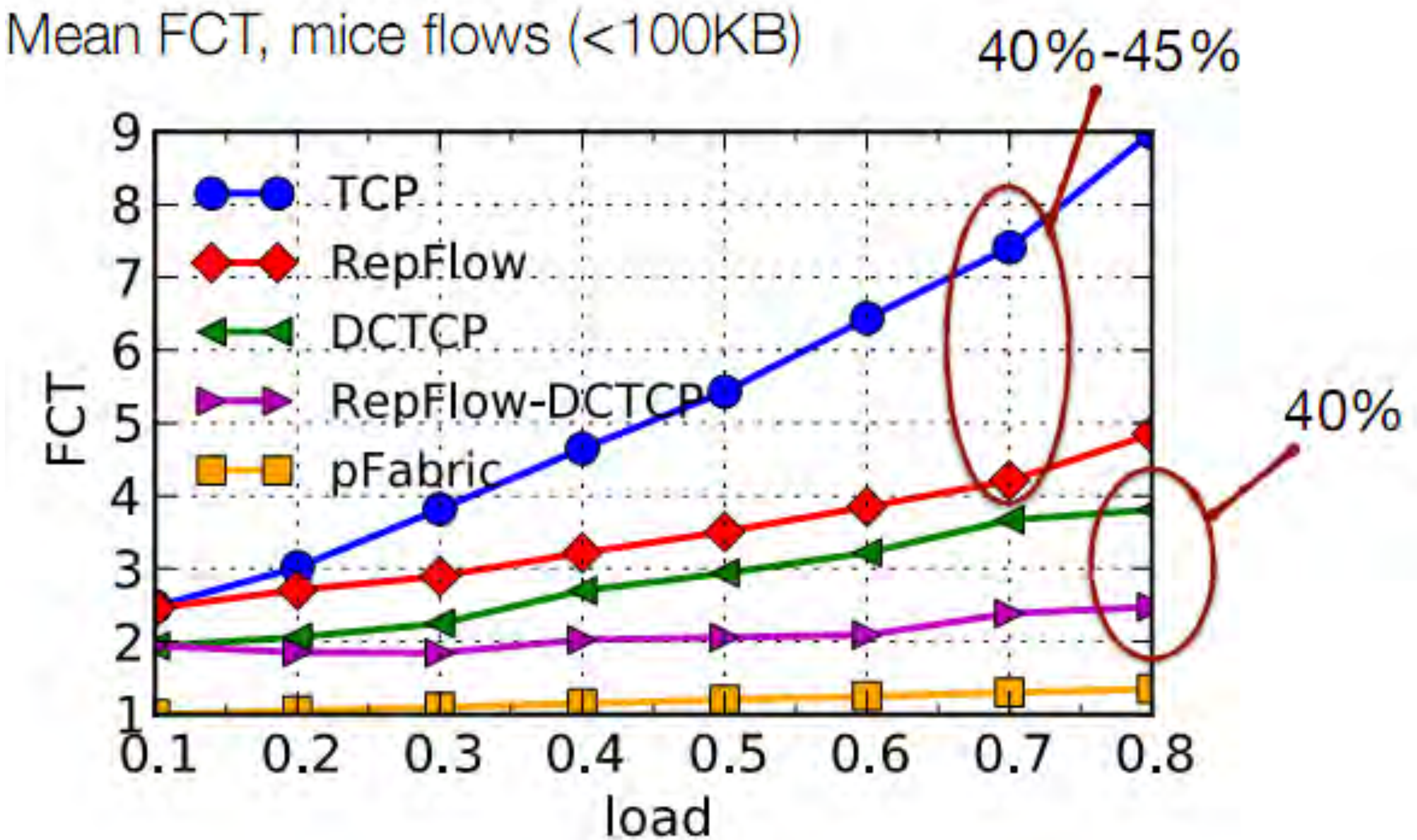
- Which flows?
  - Less than 100KB, consistent with many existing papers
- When?
  - Always! (Short flows only accounts for a tiny fraction of total bytes in production systems.)
- How?
  - Two TCP sockets, different dest ports so they will be hashed to distinct paths by ECMP

RepFlow: Practical?

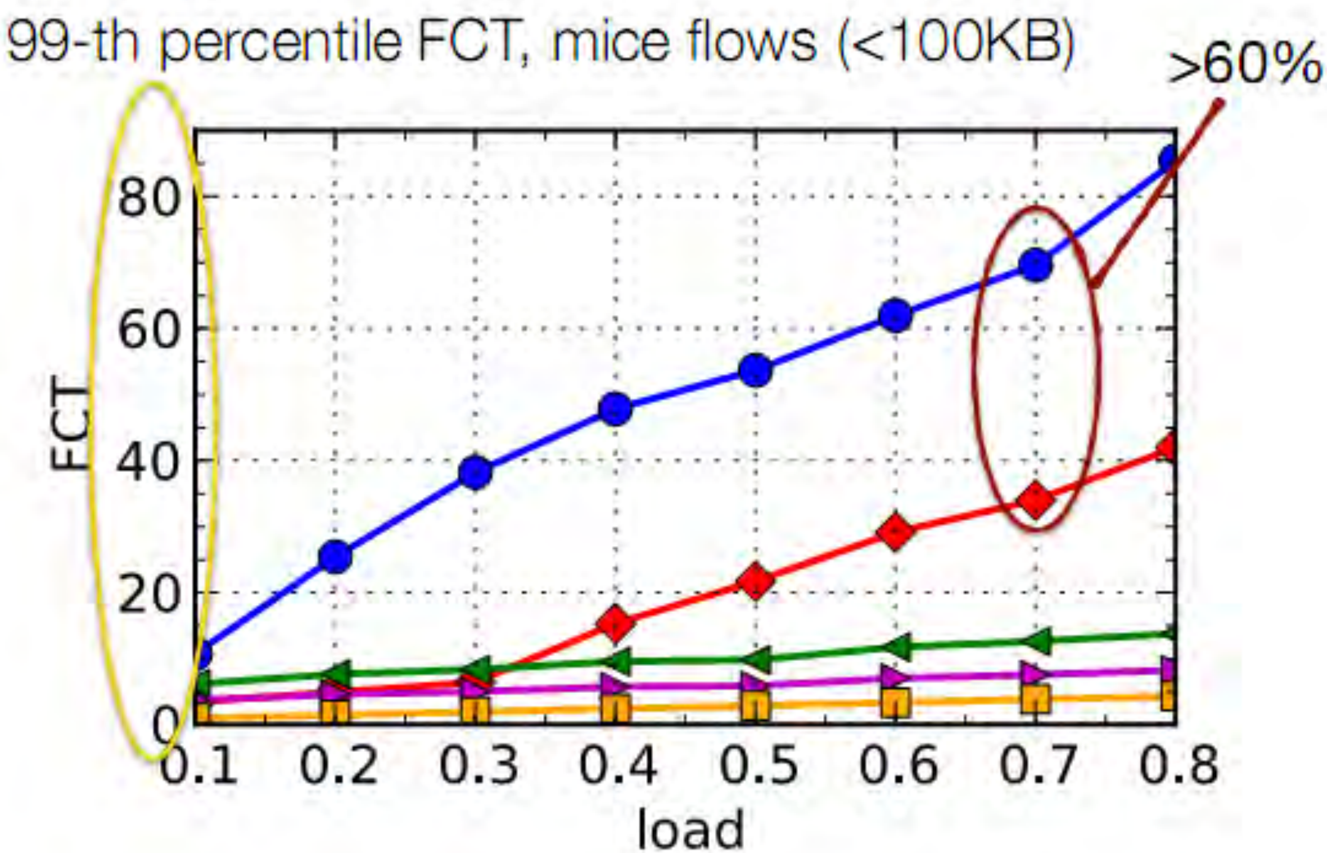
- Application layer implementation at end-hosts (sources and destinations)
  - No changes at switches
  - Works with TCP and ECMP



RepFlow: Effective?



RepFlow: Effective?



The power of two choices:

The expected maximum load of

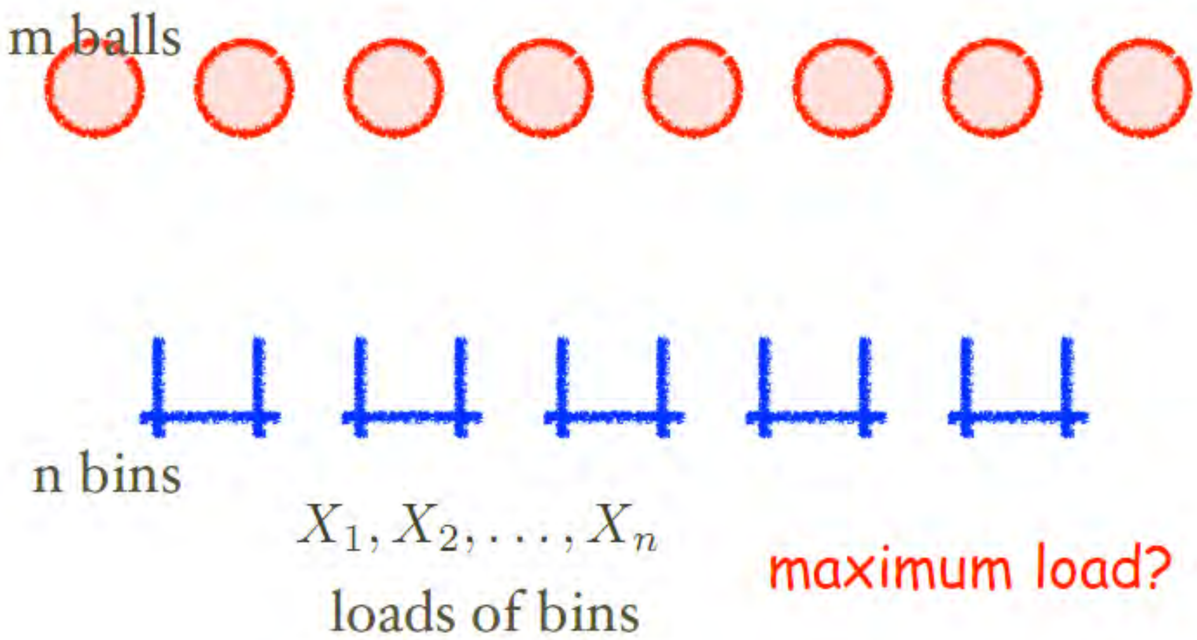
One choice:

$$\Theta(\frac{\ln n}{\ln \ln n})$$

Two choices:

$$\Theta(\ln \ln n)$$

RepFlow: The Power of Two Choices



Balls – flows
Bins – links

The power of two choices:

The expected maximum load of

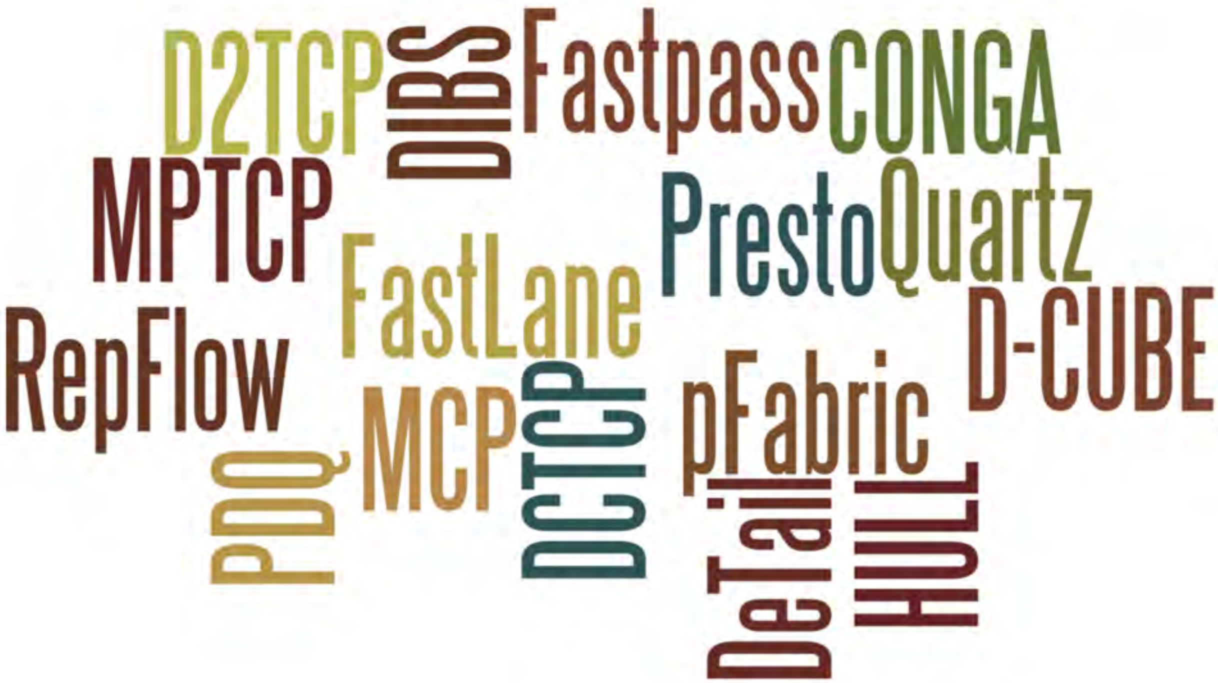
One choice:

$$\Theta\left(\frac{\ln n}{\ln \ln n}\right)$$

Two choices:

$$\Theta(\ln \ln n)$$

Datacenter fabric proposals



Which one does the operator pick?

Is there a single fabric that provides flexible and fast bandwidth allocation control?

Yes ! NUMFabric provides a flexible fabric that is also fast.

NUMFabric, SIGCOMM 2016

2. HOW TO CHOOSE UTILITY FUNCTIONS?

NUMFabric adopts NUM [33] as a flexible framework for expressing fine-grained bandwidth allocation preferences as an optimization problem. In the most basic form, each network flow is associated with a *utility* as a function of its rate. The goal is to allocate rates to maximize the overall system utility, subject to link capacity constraints:

$$\begin{aligned} \text{maximize} \quad & \sum_i U_i(x_i) \\ \text{subject to} \quad & \mathbf{Rx} \le \mathbf{c}. \end{aligned} \qquad (1)$$

Here, $\mathbf{x}$ is the vector of flow rates; $\mathbf{R}$ is the $\{0,1\}$ routing matrix, i.e., $R(i,l) = 1$ if and only if flow i traverses link l; and $\mathbf{c}$ is the vector of link capacities. The utility functions, $U_i(\cdot)$ are assumed to be smooth, increasing, and strictly concave.[1] A flow is defined generically; for example, a flow can be a TCP connection, traffic between a pair of hosts, or traffic sent or received by a host.

Allocation Objective	NUM objective
Flexible α-fairness [47]	$\sum_i x_i^{1-\alpha}/(1-\alpha)$
Weighted α-fairness	$\sum_i w_i^{\alpha} x_i^{1-\alpha}/(1-\alpha)$
Minimize FCT [3]	$\sum_i x_i/s_i$
Resource pooling [68]	$\sum_i y_i^{1-\alpha}/(1-\alpha)$, where $y_i = \sum_{p\in Path(i)} x_{ip}$
Bandwidth functions [35]	$\sum_i \int_0^{x_i} F_i(\tau)^{-\alpha} d\tau$

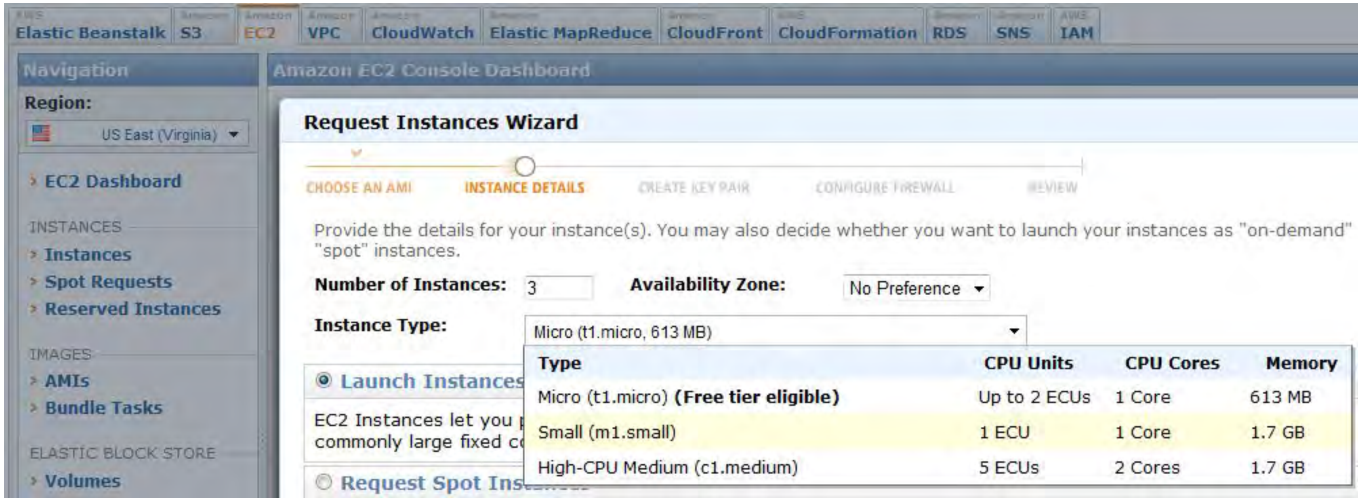
Table 1: Example utility functions for several resource allocation policies. The case $\alpha = 1$ is to be interpreted in the limit $\alpha \to 1$; e.g., $\sum_i \log x_i$ for the first row.

PIAS, NSDI 2015

研究问题2：Bandwidth Guarantee in Data Centers

The Interface between Providers & Users

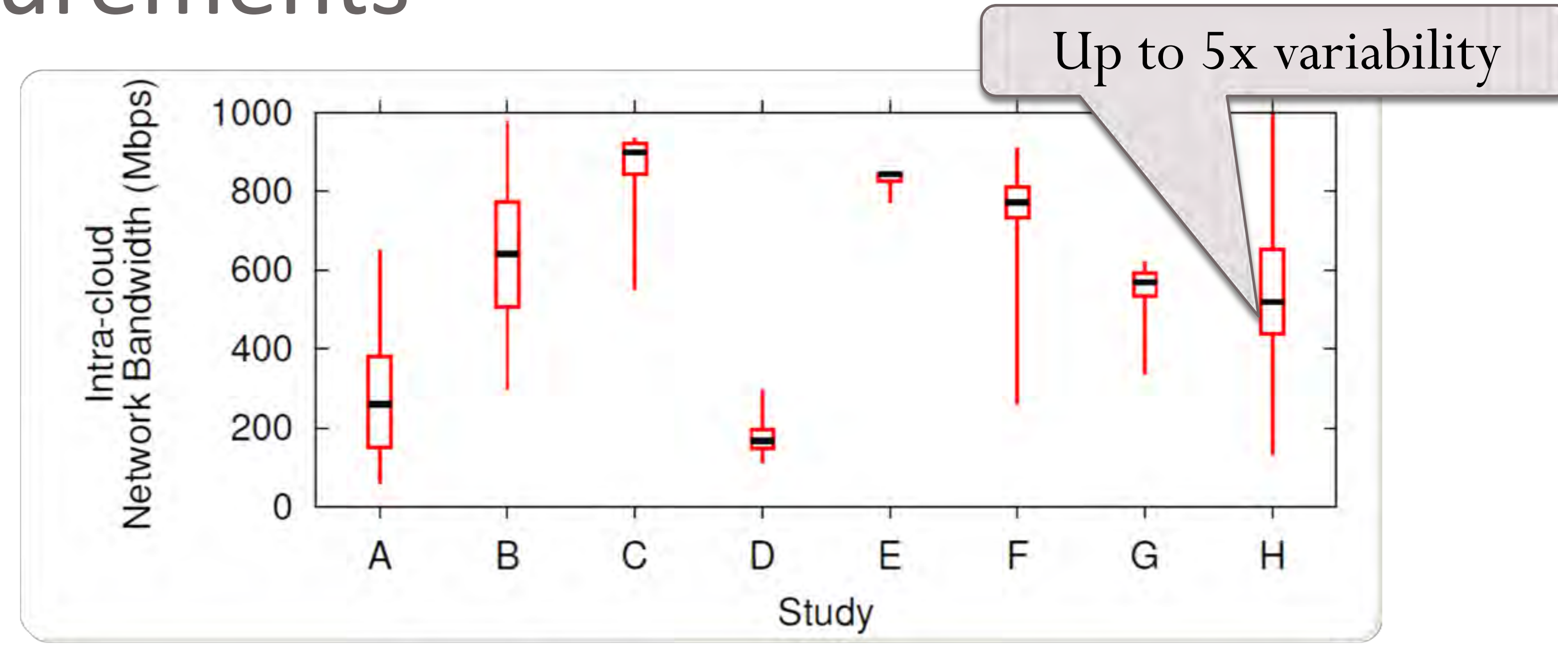
Amazon EC2 Interface



- Billing is based on per-VM (e.g., processing cores, memory, storage) and per-hour.
  - Amazon EC2 small instance: $0.085/hour

Bandwidth is NOT guaranteed!

Measurements



Study	Study	Provider	Duration
A	[Giurgui'10]	Amazon EC2	n/a
B	[Schad'10]	Amazon EC2	31 days
C/D/E	[Li'10]	(Azure, EC2, Rackspace)	1 day
F/G	[Yu'10]	Amazon EC2	1 day
H	[Mangot'09]	Amazon EC2	1 day

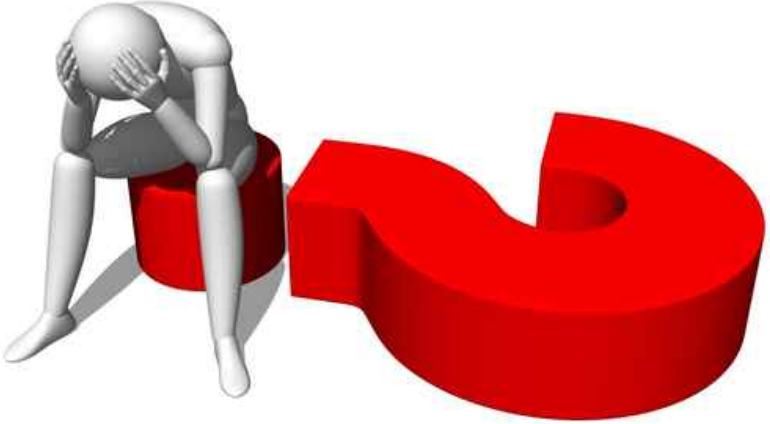
Negative Consequences

of Unpredictable Performance

- Cloud users: pay more
  - Implicitly pay for network congestion.
  - Will not migrate their important applications to data centers.
- Cloud providers: earn less
  - Cannot achieve high resource utilization.
  - Lose potential customers.

Double loss!



There must be a way out somewhere!

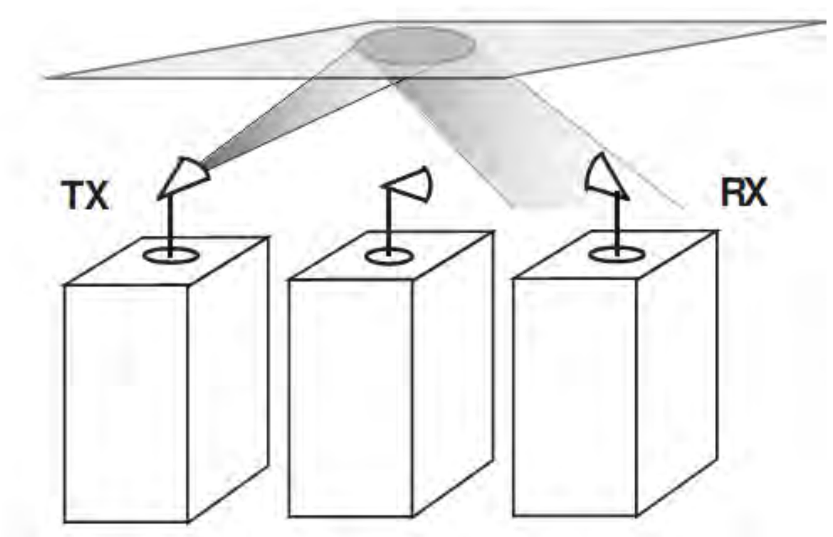
- Bandwidth augment
  - E.g., multi-gigabit, 60 GHz wireless links
- Network architecture design
  - E.g., Fat-tree, VL2, DCell, Bcube
- Network virtualization
  - E.g., SecondNet, Oktopus, Proteus
- Network aware virtual machine placement

Hardware

Software

Augmenting with 60 GHz [1]

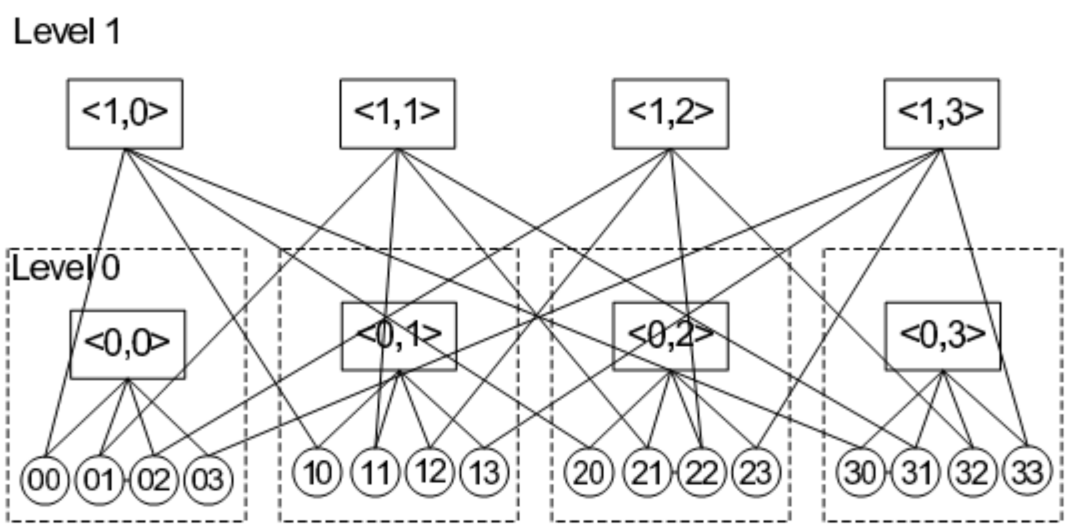
- For congestion loss caused by short traffic bursts
- 60 GHz: millimeter wave
  - Unlicensed; Highly secure; Virtually interference-free; High rate; etc.
- 3D beamforming



[1] Xia Zhou, et al., Mirror mirror on the ceiling: flexible wireless links for data centers, ACM SIGCOMM 2012

Bcube: A Generalized Hypercube[2]

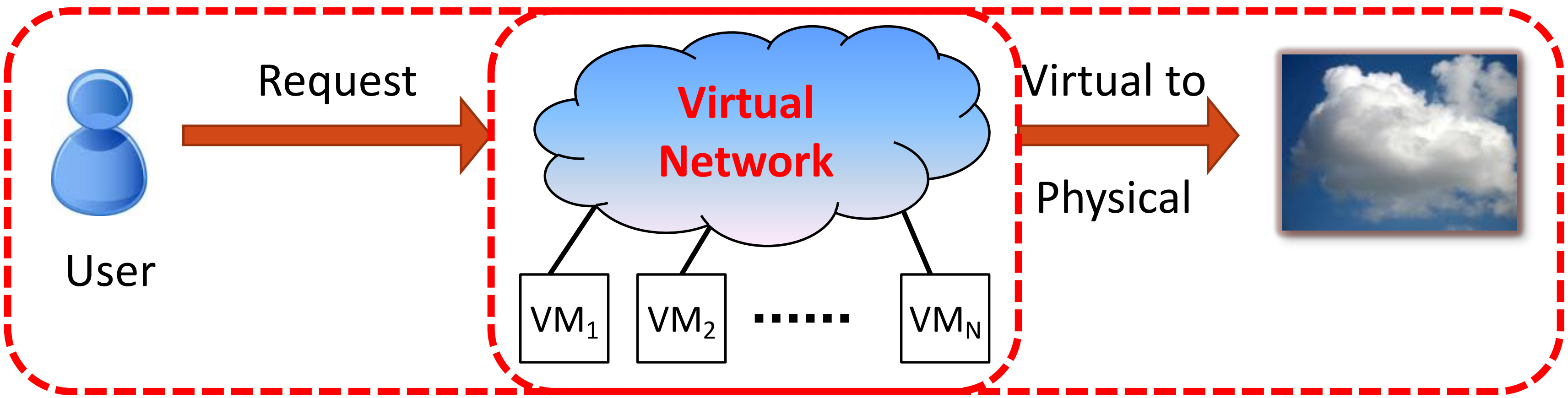
- High network capacity for various traffic patterns
  - Unicast, groupcast, shuffling
- Lots of good properties
  - E.g., there are k+1 parallel paths between any two servers in a Bcube_k .



[2] Chuanxiong Guo, et al., BCube: A High Performance, Server-centric Network Architecture for Modular Data Centers, ACM SIGCOMM 2009

Towards Predictable Datacenter Networks[3]

- Allocate & Enforce!
  - Virtual network abstraction: simple vs. arbitrary
  - Allocation strategy: static vs. dynamic
  - Enforcement : rate limiting by hypervisors and switches



[3] Hitesh , et al., Towards Predictable Datacenter Networks, ACM SIGCOMM 2011

Network Aware Virtual Machine Placement

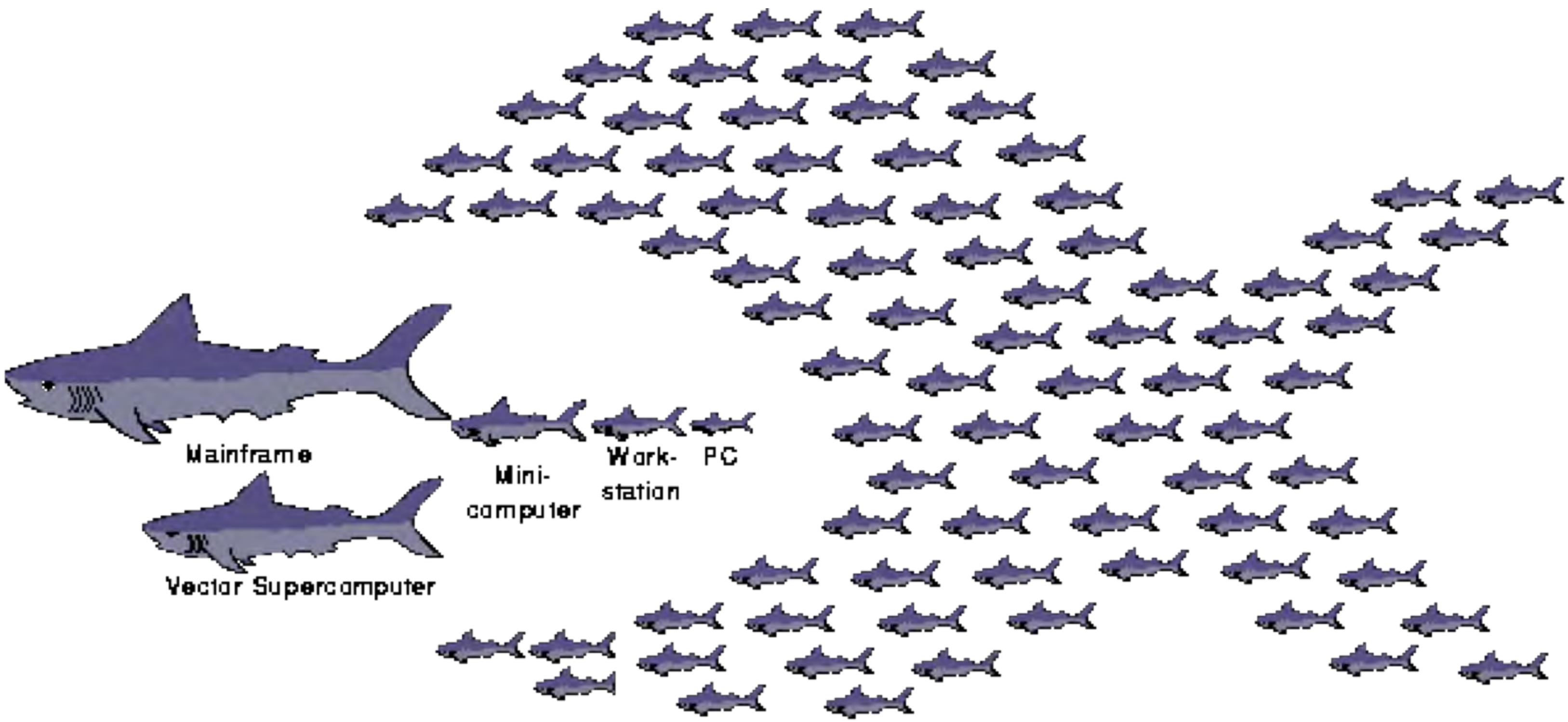
- Input
  - Traffic matrix among VMs
  - Cost matrix among slots
- Output
  - A VM-to-slot mapping that minimizes (or maximizes) an objective
- Two examples
  - Xiaoqiao Meng, et al., Improving the scalability of data center networks with traffic-aware virtual machine placement, IEEE Infocom 2010
  - Mansoor Alicherry, et al., Network aware resource allocation in distributed clouds, IEEE Infocom 2012

云计算未来

机遇与挑战

- 服务稳定性（Availability）
- 数据的不通用性（Data Lock-in）
- 数据保密与可信（Data Confidentiality and Auditability）
- 数据传输瓶颈（Data Transfer Bottlenecks）
- 性能不可预测（Performance Unpredictability）
- 可扩展的存储（Scalable Storage）
- 大规模分布式系统的Bug（Bugs in Large-Scale Distributed Systems）
- 可扩展性（Scaling Quickly）

网络的发展


Mainframe
Mini-
computer
Work-
station
PC
Vector Supercomputer

NOW