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

ClickOS and the Art of Network Function Virtualization

Joao Martins, Mohamed Ahmed, CostinRaiciu, VladimirOlteanu, et al.

NSDI, 2014

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

Introduction

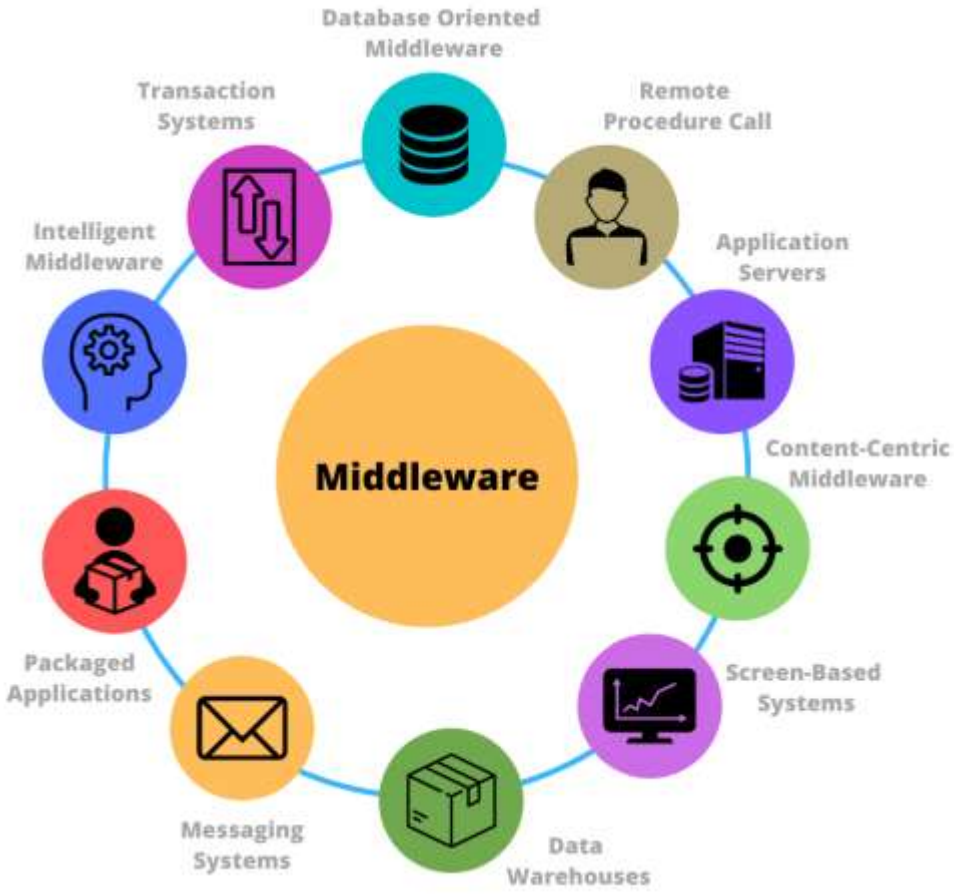
中间件（Middlebox）在现代网络中十分普及

- 安全防护类：防火墙（Firewall）、入侵检测系统（IDS）、流量清洗（Traffic Scrubber）
- 流量控制类：速率限制器（Rate Limiter）、负载均衡器（Load Balancer）
- 地址管理类：网络地址转换（NAT），用于应对 IPv4 地址枯竭
- 性能优化类：流量加速器（Traffic Accelerator）、缓存（Cache）、代理（Proxy）等

Introduction

硬件中间件存在局限性

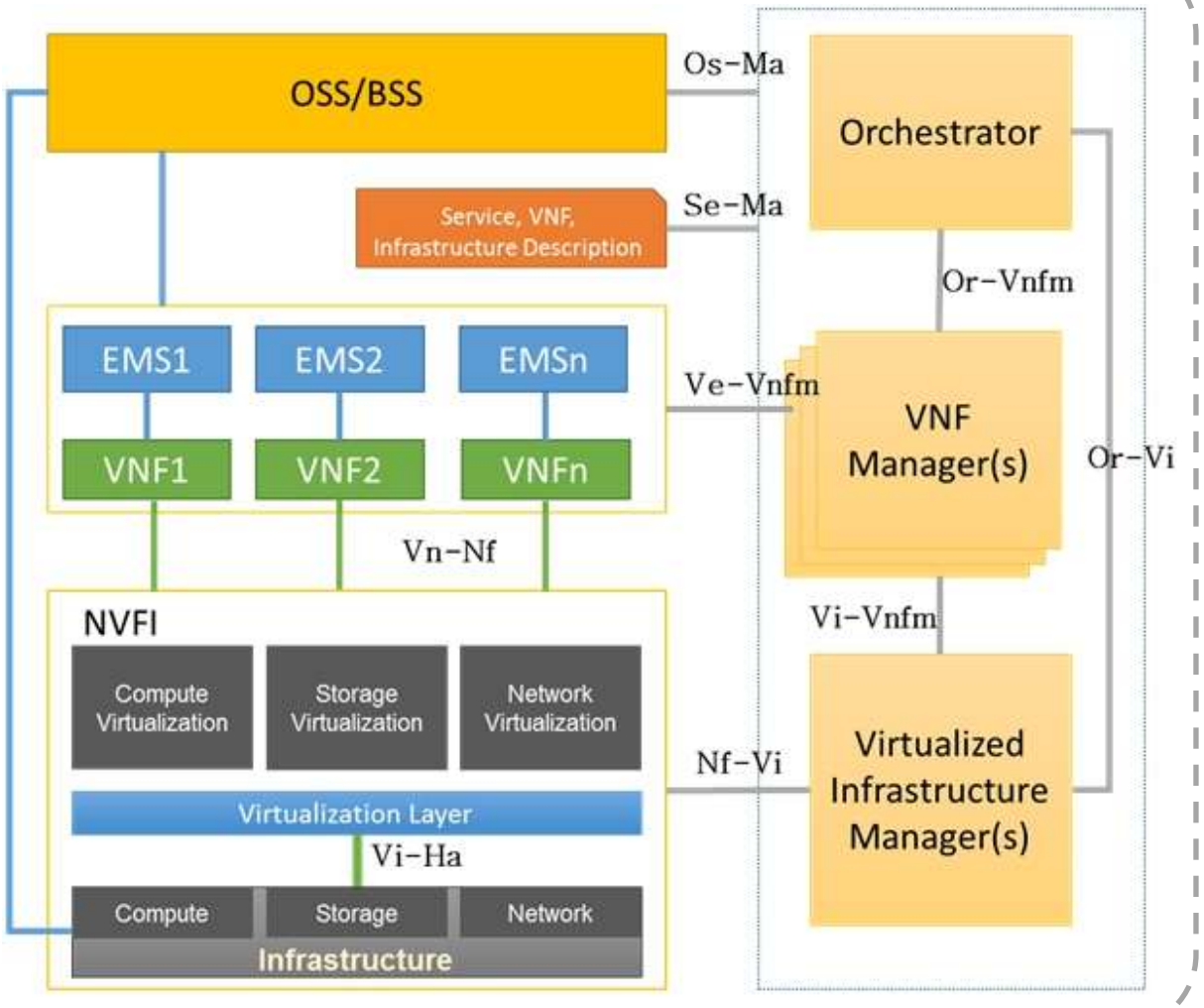
- 采购与维护昂贵
- 功能升级依赖新硬件更新周期（通常4年）
- 难以动态扩展以应对流量波动，造成资源浪费
- 开发新型硬件设备需要大量投资，提高了进入市场的技术门槛，抑制了创新。



Introduction

NFV（Network Function Virtualization）理念

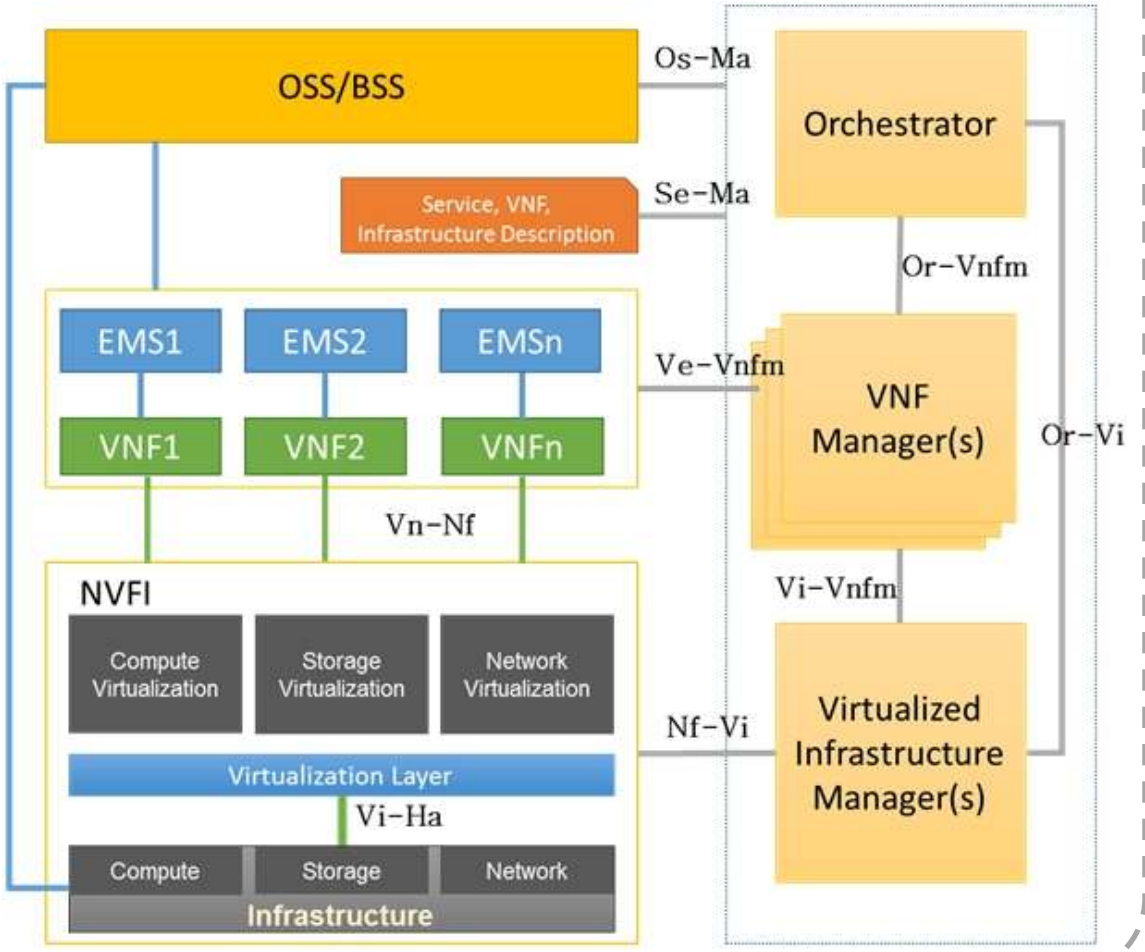
- 为解决上述问题，业界提出了 NFV（网络功能虚拟化）方案
- 将中间件功能从专有硬件迁移至运行在通用服务器上的软件中
- 该理念得到电信运营商和设备厂商的广泛支持，并推动了相关标准的制定



Introduction

NFV平台的设计要求

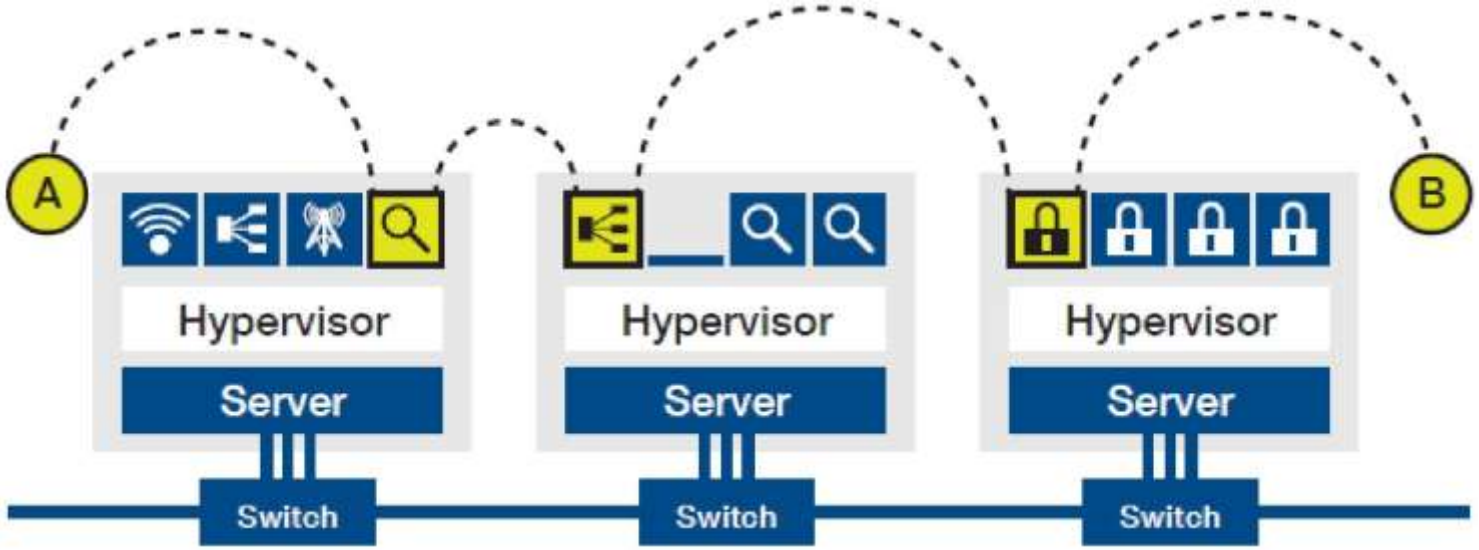
- 多租户隔离
  - 必须能在同一物理服务器上并行运行多个、甚至来自不可信任的第三方虚拟中间件
- 安全与性能保障
- 高吞吐低延迟
- 支持多种操作系统与API接口



Introduction

现有虚拟化平台的不足

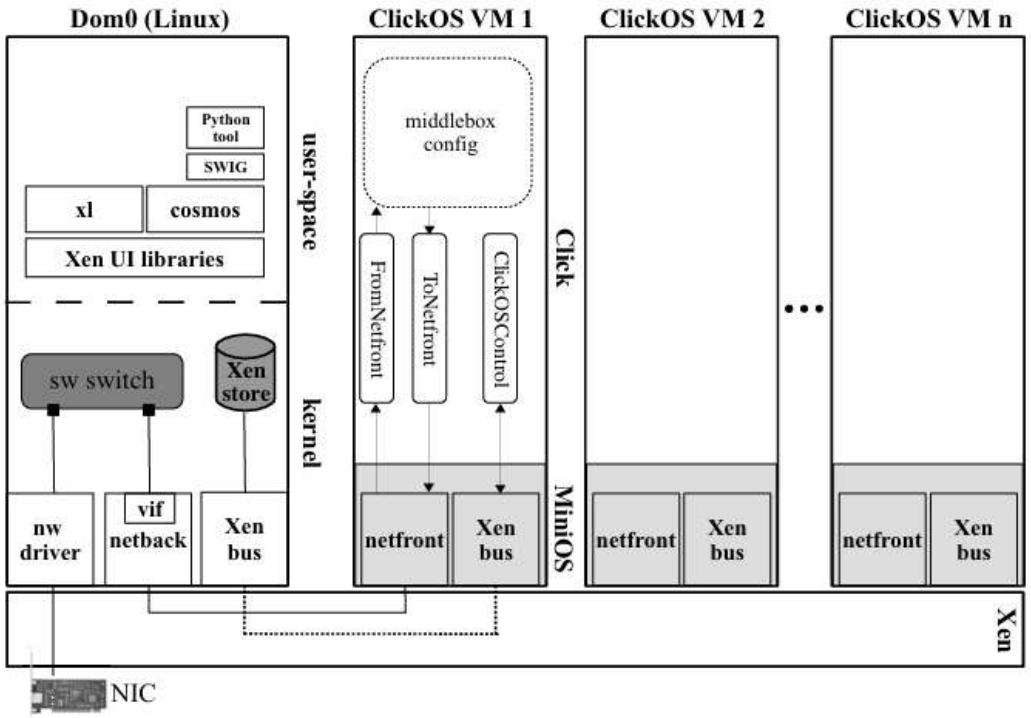
- Xen、KVM 等虚拟化技术具备安全与隔离性，但其仅支持少量租户，并且网络性能无法满足中间件需求
- 这些通用虚拟化平台与操作系统（如 Linux）并未针对中间件的高性能数据平面场景优化



Introduction

ClickOS的目标与优化思路

- 通过重构 Xen 的 I/O 子系统（包括虚拟交换机、驱动和数据路径），显著提升网络中间件性能，使虚拟化中间件实现接近硬件线速的吞吐能力
- 使用Click 框架，Click通过模块化、可组合的方式实现网络功能，使中间件开发更灵活。故ClickOS 将 Click 与轻量级虚拟机结合，在性能与开发效率之间取得平衡



Outline

Problem Statement

构建平台的核心目标

构建一个高性能、灵活且通用的软件中间件平台，能够在普通硬件上运行（即基于通用x86服务器的NFV平台）

核心要求

- 灵活性（Flexibility）：能支持不同类型的软件中间件、操作系统与框架
- 隔离性（Isolation）：在多租户环境中提供CPU、内存与设备的安全隔离
- 高性能（High Throughput & Low Delay）：中间件常部署在高带宽环境，应支持多10Gbps级别传输，同时保证端到端延迟极低
- 可扩展性（Scalability）：能根据负载快速伸缩，以提高资源利用率

Problem Statement

编程方式问题

- 当前大多数中间件是作为应用程序或内核扩展运行在通用操作系统（如Linux）上。这种方式灵活，但增加了系统复杂性，并且浪费大量系统资源

功能复用

- 不同中间件之间存在大量功能重叠，如数据包过滤、地址重写、速率控制等
- 因此，NFV平台应支持代码与处理逻辑的重用机制。这样能降低开发成本，减少冗余计算，从而提高整体性能与资源利用率

Outline

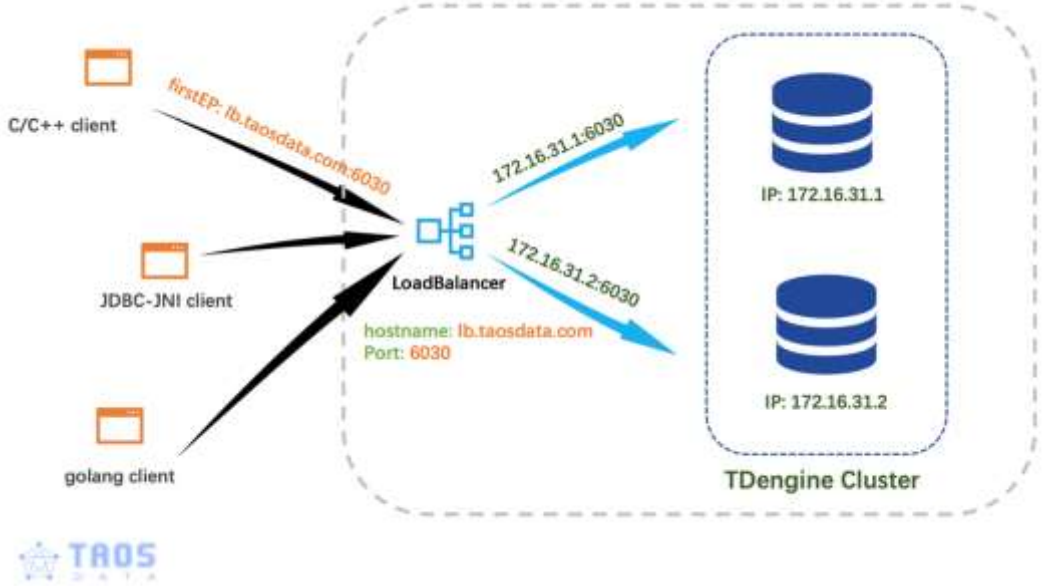
Related Work

在构建NFV平台时，已有许多相关研究可供借鉴。

核心挑战在于如何在同一硬件上隔离不同中间件。

可行的技术路径主要有两类：

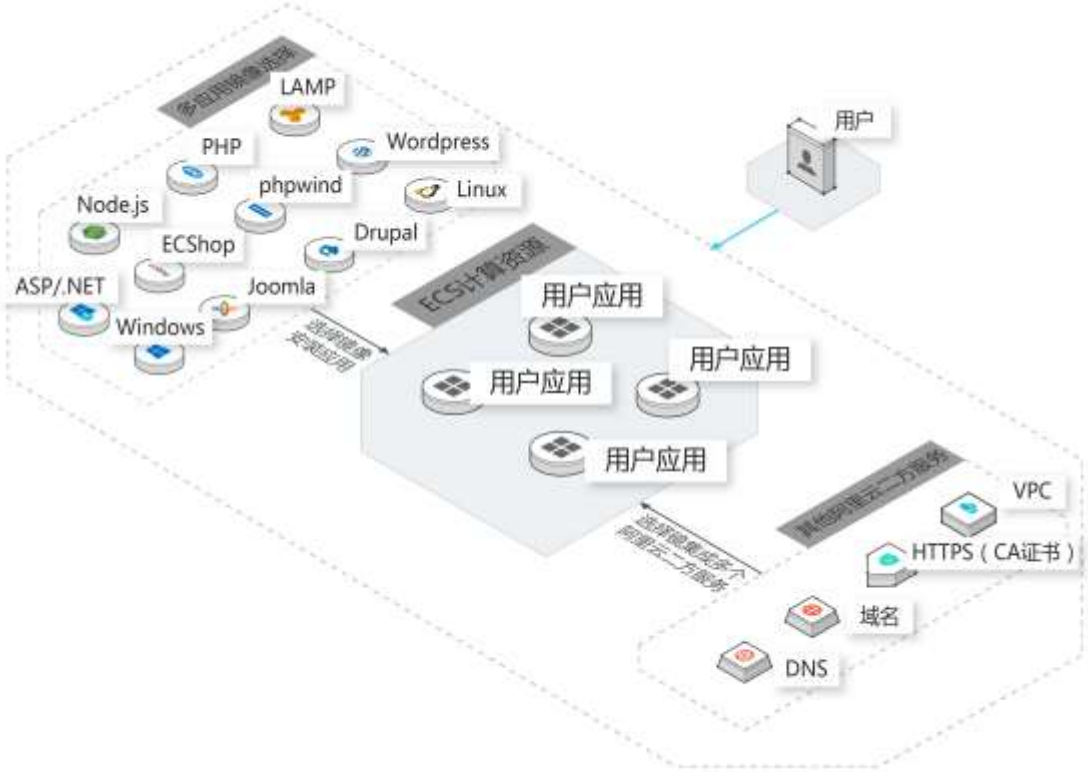
- 基于容器（Containers）的轻量隔离
  - 具备轻量化和高性能特点
  - 要求所有实例运行在相同操作系统内核之上，无法满足多租户与异构环境需求
- 基于虚拟机（Hypervisors）的强隔离
  - 更强的隔离性与灵活性，能够运行不同操作系统的中间件
  - 性能较低，特别是在网络I/O上存在显著瓶颈



Related Work

轻量操作系统的探索

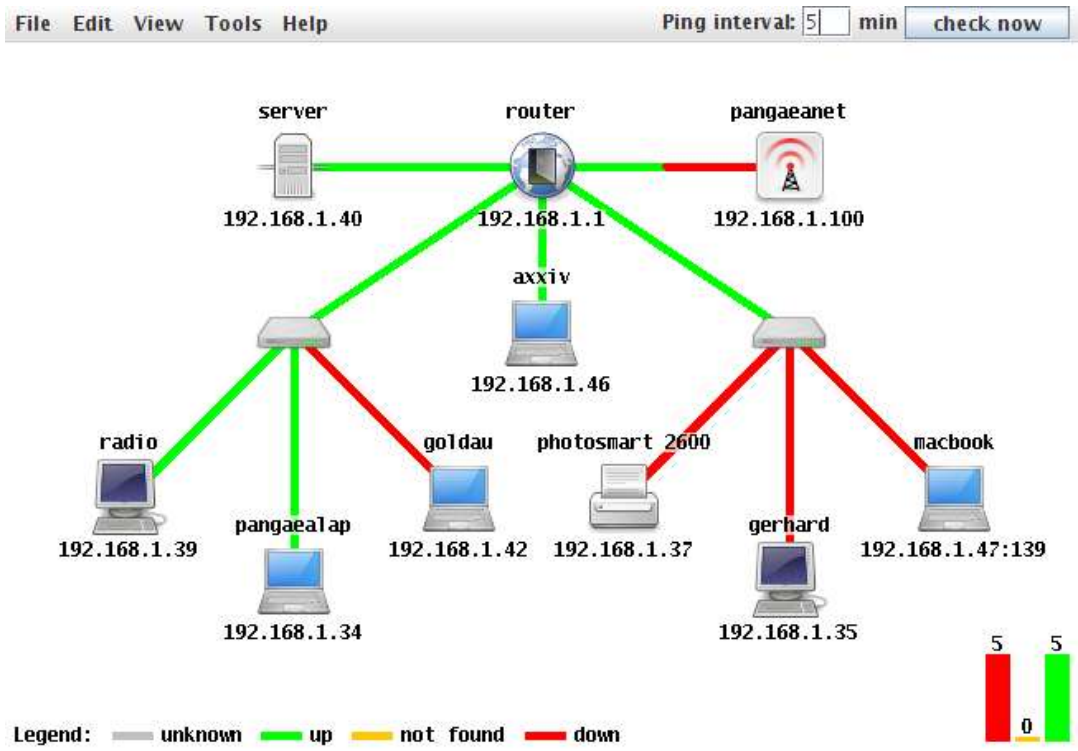
- 研究者尝试通过设计“最小化操作系统（Minimalistic OSes）”来提高性能，只保留执行任务所需的核心功能
- Mirage、Erlang on Xen、HalVM 等都尝试将轻量OS运行在虚拟化环境中
- 普遍存在驱动支持不足、网络性能较差的问题，且多数并非专为中间件处理优化



Related Work

高性能网络I/O的相关研究

- 多项研究（如RouteBricks、PacketShader、DPDK、netmap等）关注于提高软件网络处理的性能
- 通过用户态直接访问网卡内存、减少内核态切换、批处理数据包等方式提升速率
- ClickOS借鉴了这些思想，尤其是netmap的用户态内存映射机制，构建了更直接高效的虚拟化数据通道



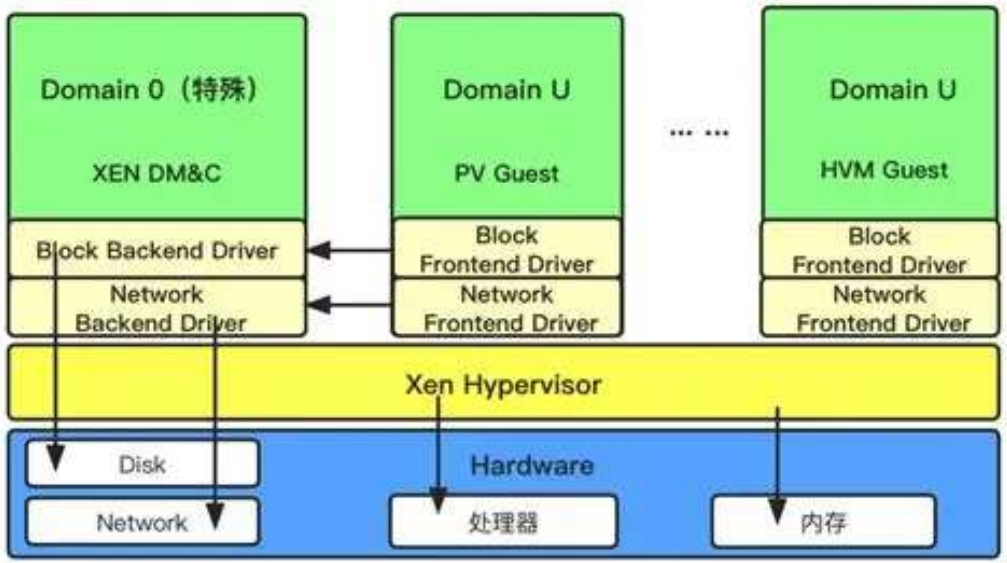
Related Work

Xen网络性能改进工作

- 研究通过“grant重用”和改进I/O调度策略等方式减少开销，提升Xen虚拟机的网络性能
- ClickOS在设计中综合采用了这些优化手段

软件交换机的优化研究

- 部分研究（如Hyper-Switch）尝试将虚拟交换机整合进Hypervisor内核，以减少上下文切换
- 其采用的Open vSwitch数据平面仍存在性能瓶颈



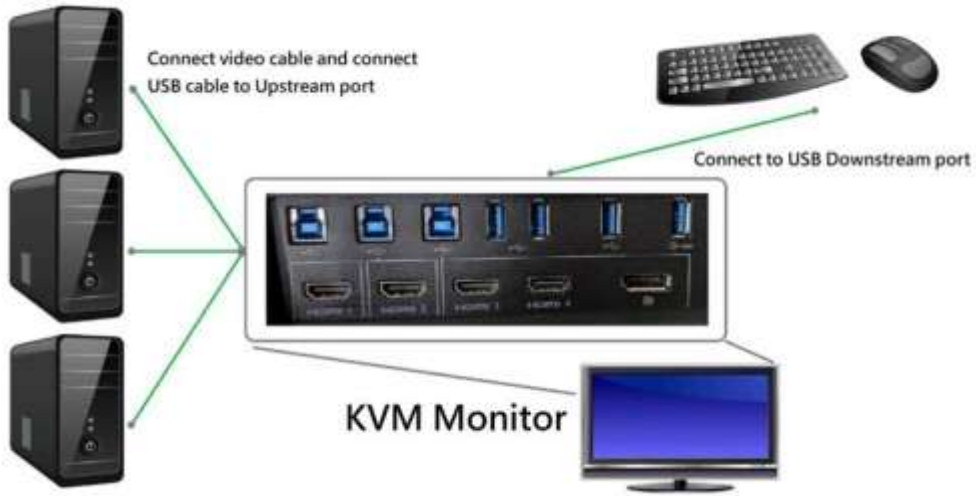
Related Work

KVM环境下的优化尝试

- 有研究试图在KVM中优化网络I/O（如vpf_ring等）
- 这些工作多针对虚拟机流量监控或通用加速，未聚焦中间件虚拟化

软件中间件的先前研究

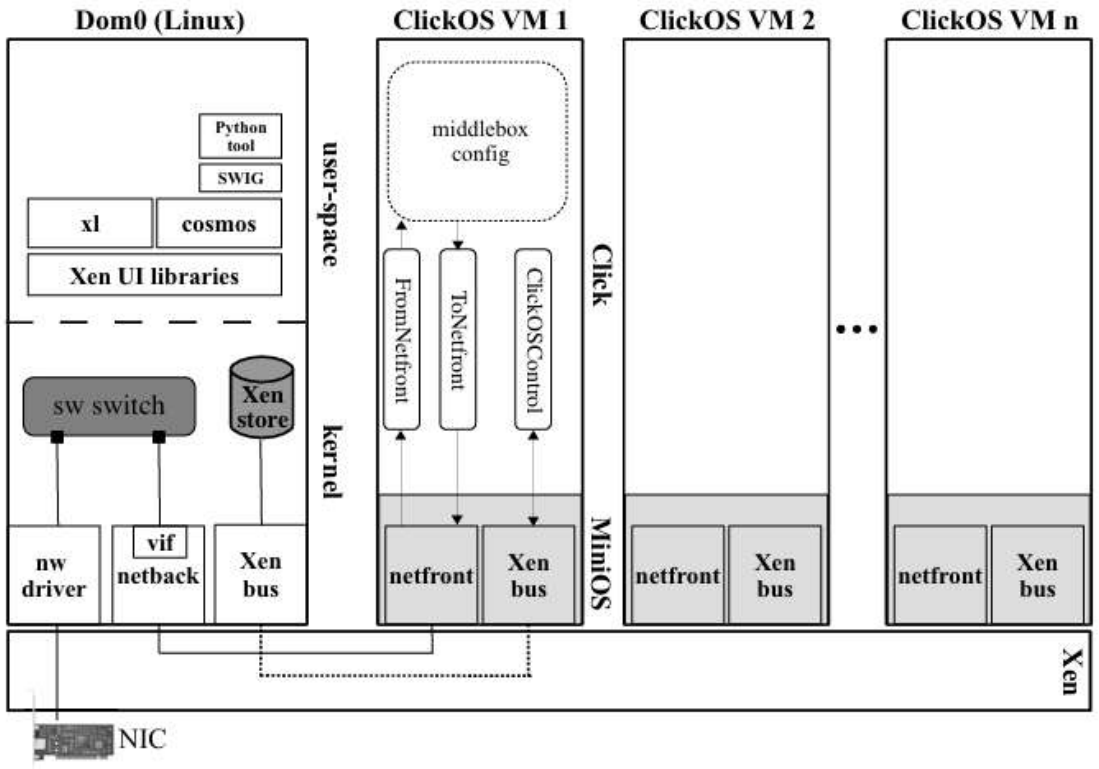
- Comb框架提出了中间件整合架构，但缺乏多租户隔离能力，性能也不理想（两核仅4.5Gb/s）
- Vyatta、Cisco等商业方案虽支持软件中间件，但公开性能数据有限



Outline

ClickOS Design

- ClickOS 的设计目标是同时实现 灵活性、隔离性与多租户支持
- 为此，系统采用虚拟化（hypervisor-based）方案，即在硬件与中间件软件之间加入一层虚拟化层
- 这种方式可能带来额外延迟，但通过合适优化可以平衡性能与隔离性



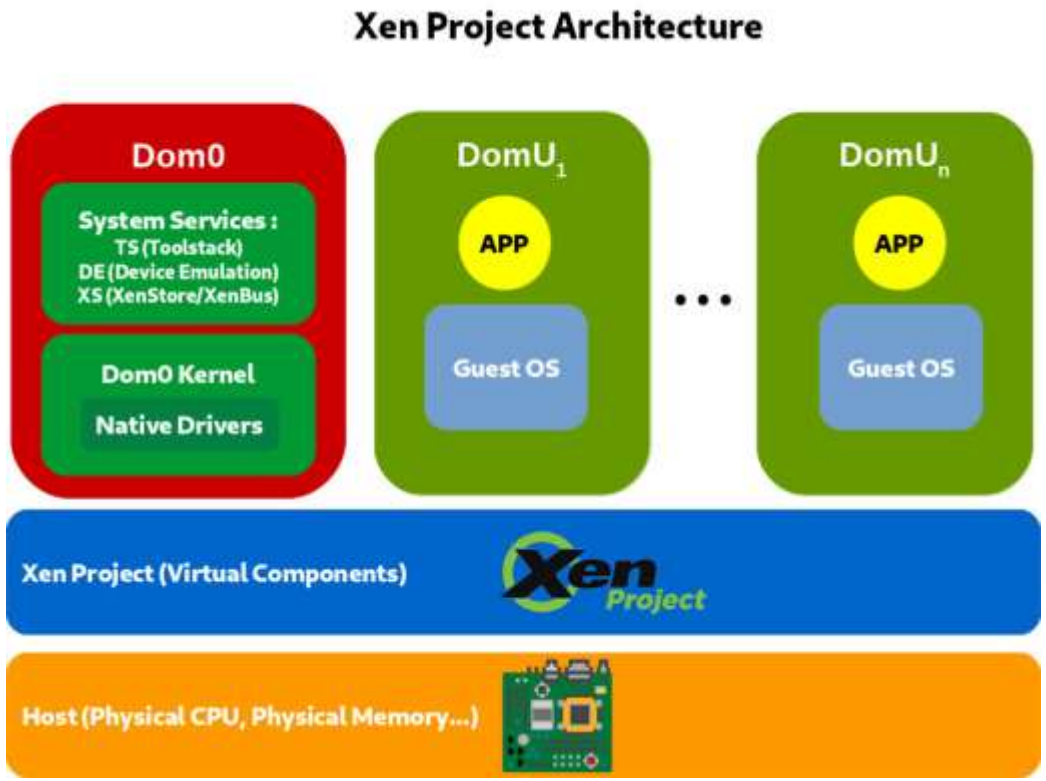
ClickOS Design

半虚拟化（para-virtualization）

- 与完全虚拟化相比，半虚拟化更适合 NFV 场景
- 通过对客户操作系统进行少量修改，避免频繁的 VM Exit 和指令模拟，降低虚拟化开销

Xen 作为基础平台

- Xen 支持精细的性能隔离与安全机制，适合构建低延迟、高吞吐的中间件系统
- Xen 默认配置的性能仍然有限，因此 ClickOS 在其 I/O 子系统上进行了深入重构



ClickOS Design

中间件功能的可编程抽象

- 目前中间件大多是以用户态应用或内核模块的形式编写，虽然灵活，但开发复杂、调试困难
- C 语言虽性能高，但难以复用已有功能
- ClickOS 需要选择最佳编程抽象

Click 框架的优势

- Click 模块化路由器框架由多个可复用的element组成，支持快速组合形成复杂网络功能

Middlebox	Key Click Elements
Load balancer	`RatedSplitter, HashSwitch`
Firewall	`IPFilter`
NAT	`[IP|UDP|TCP]Rewriter`
DPI	`Classifier, IPClassifier`
Traffic shaper	`BandwidthShaper, DelayShaper`
Tunnel	`IPEncap, IPsecESPEncap`
Multicast	`IPMulticastEtherEncap, IGMP`
BRAS	`PPPControlProtocol, GREEncap`
Monitoring	`IPRateMonitor, TCPCollector`
DDoS prevention	`IPFilter`
IDS	`Classifier, IPClassifier`
IPS	`IPClassifier, IPFilter`
Congestion control	`RED, SetECN`
IPv6/IPv4 proxy	`ProtocolTranslator46`

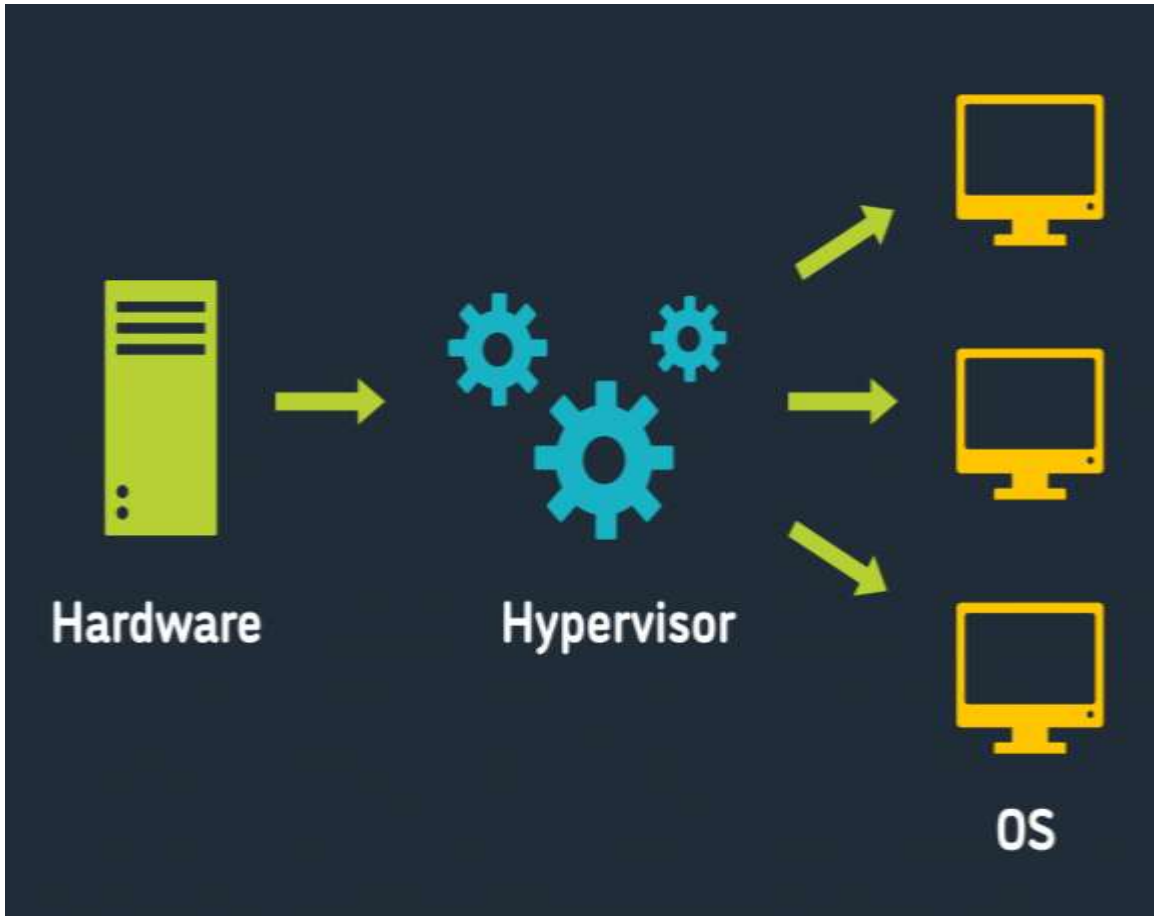
ClickOS Design

Click 的局限与ClickOS的改进

- Click 框架默认依赖 Linux 运行
- 为实现域隔离，我们必须在Linux虚拟机内运行每个Click中间件，违背了可扩展性要求，造成高开销

ClickOS的改进方向是：将 Click 从 Linux 中剥离出来，运行在一个极简的专用虚拟机上

- 思考：Click需要操作系统提供怎样的支持，才能实现广泛的中间件处理能力？



ClickOS Design

Click需要的支持

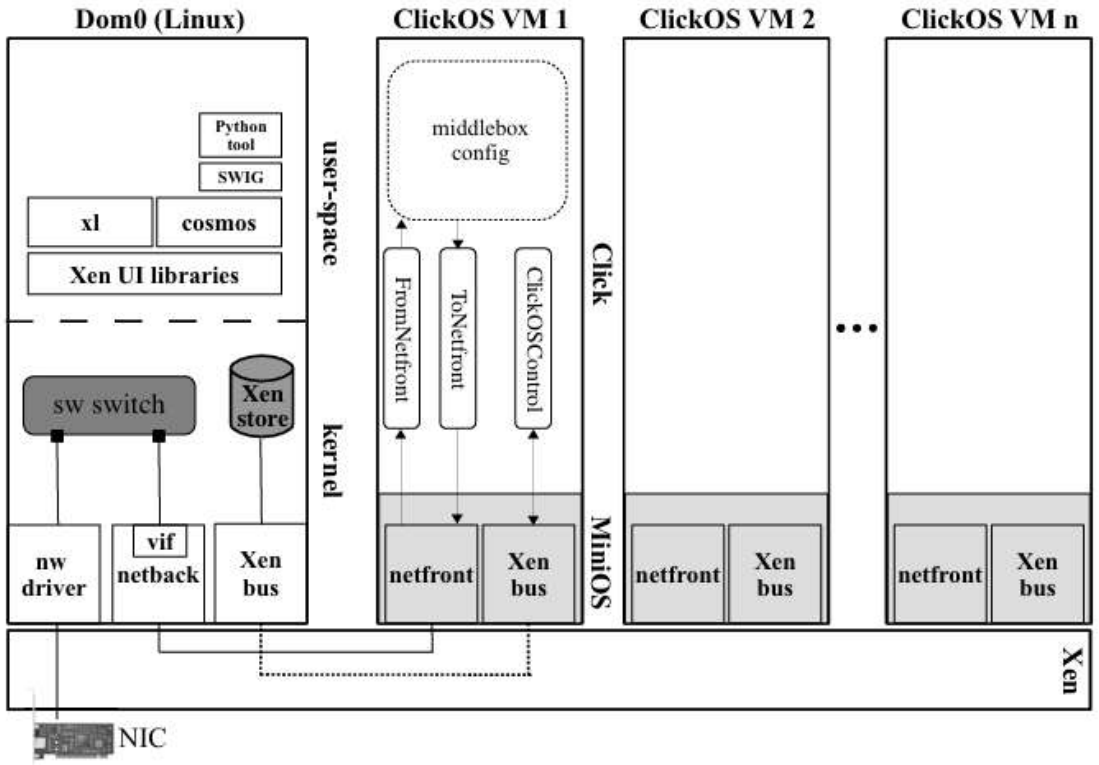
- 驱动支持（Driver support）：能够与不同类型的网络接口（NIC）交互
  - Xen通过半虚拟化技术解决了这个问题，客户机通过连接到驱动域的唯一硬件无关驱动程序访问所有网卡类型，驱动域直接与硬件进行通信
- 基本内存管理（Memory management）：能为数据结构和数据包分配内存
- 简单调度器（Scheduler）：用于在执行 Click element代码与处理中断之间切换

- 几乎所有操作系统都能满足上述后两个要求，因此无需从零构建一个全新的 OS。我们只需要一个极简且启动快速的操作系统
- Xen 自带的MiniOS体积小、结构精简，非常适合构建高效的虚拟化中间件
- 因此，MiniOS 成为 ClickOS 虚拟机的基础操作系统

ClickOS Design

ClickOS 架构概览

- Xen 网络 I/O 子系统的优化：使传统虚拟机能够实现高速网络通信
- 基于 Click 的专用中间件虚拟机：针对高性能包处理进行优化
- 管理工具集（Cosmos 工具）：用于构建、部署与管理 ClickOS 虚拟机，支持插入、删除及查询中间件状态



Outline

ClickOS Virtual Machines

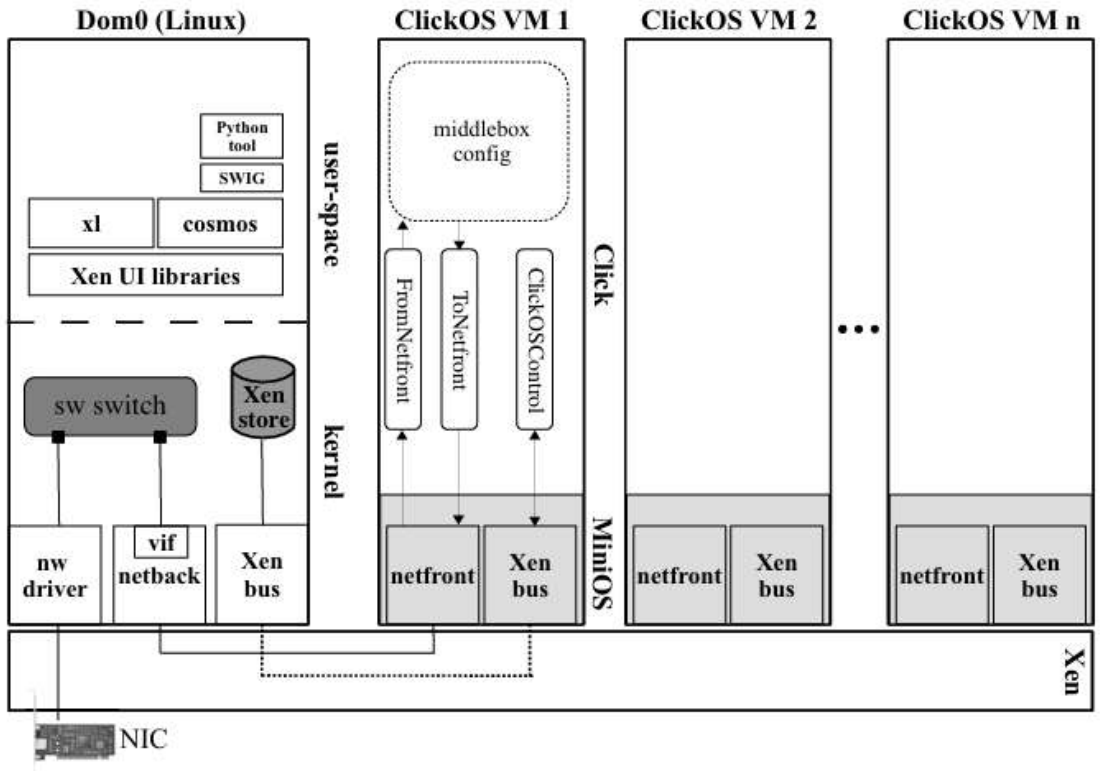
Xen 的体系结构

Xen 将系统划分为：

- dom0（特权域）：特权虚拟机，负责管理虚拟机（VM）和设备驱动
- domU（用户域）：用户虚拟机，运行用户应用或中间件

Xen采用分离驱动模型：

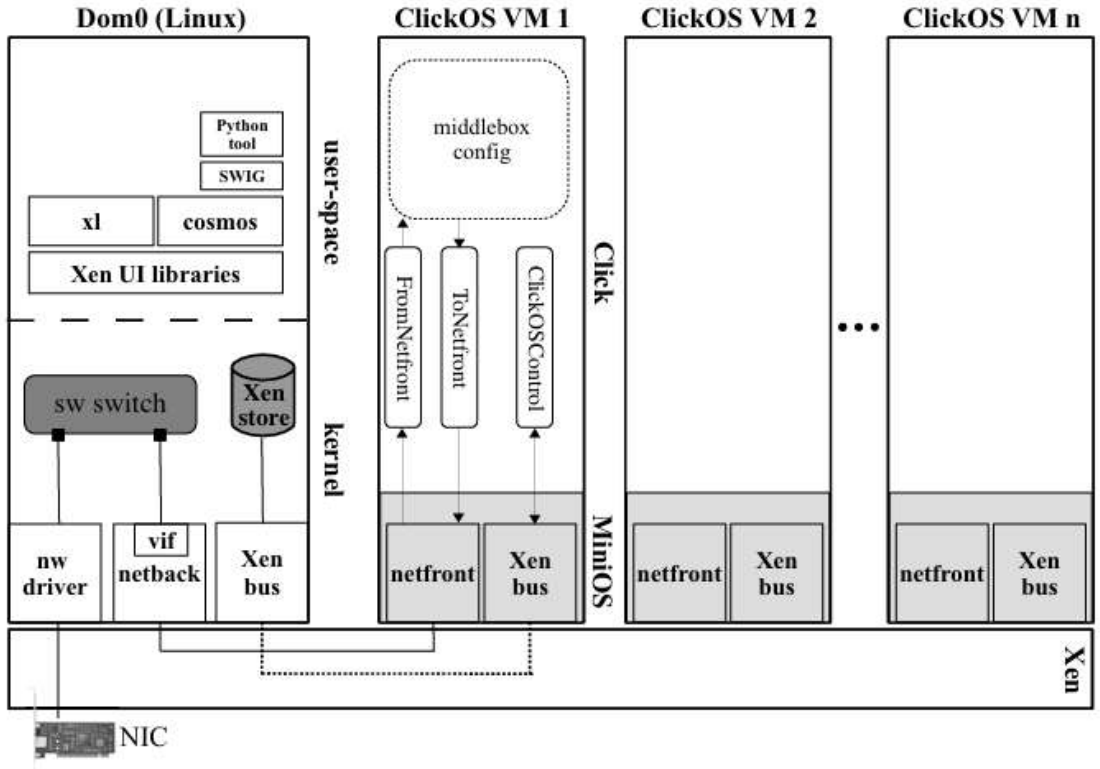
- dom0 中运行驱动程序的后端（netback），在 domU 中运行前端驱动（netfront）
- 两者通过共享内存（shared memory）与基于环的通用API的通信机制进行交互



ClickOS Virtual Machines

构建 ClickOS 虚拟机

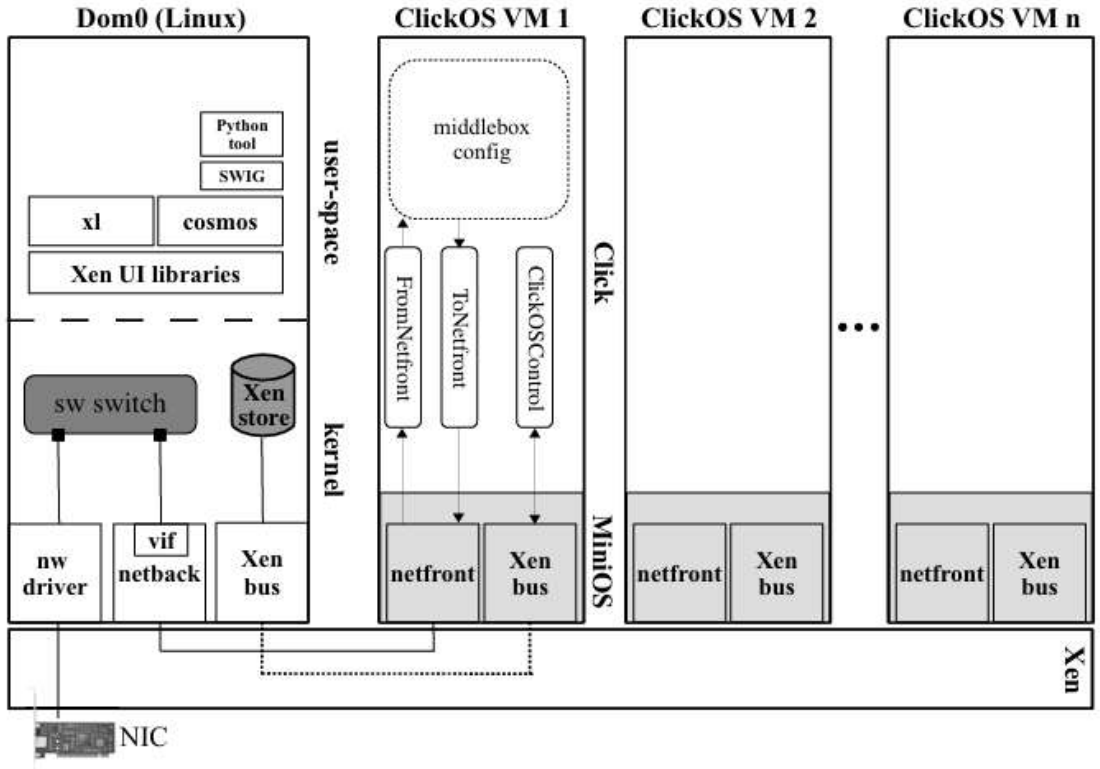
- ClickOS虚拟机均由在MiniOS之上运行的Click模块化路由器软件构成
- 由于 MiniOS 本身并不支持大多数标准的 C++ 库（如 glibc）
- 文章开发了一个新型交叉编译器，以支持C++编译并解决标准库的兼容问题，使得只需链接应用程序的入口点即可在虚拟机启动时运行，从而快速便捷地构建基于MiniOS的任意虚拟机
- 最终生成的ClickOS镜像支持216/282个Click element，其余多数element需依赖文件系统运行



ClickOS Virtual Machines

构建并启动 ClickOS 虚拟机

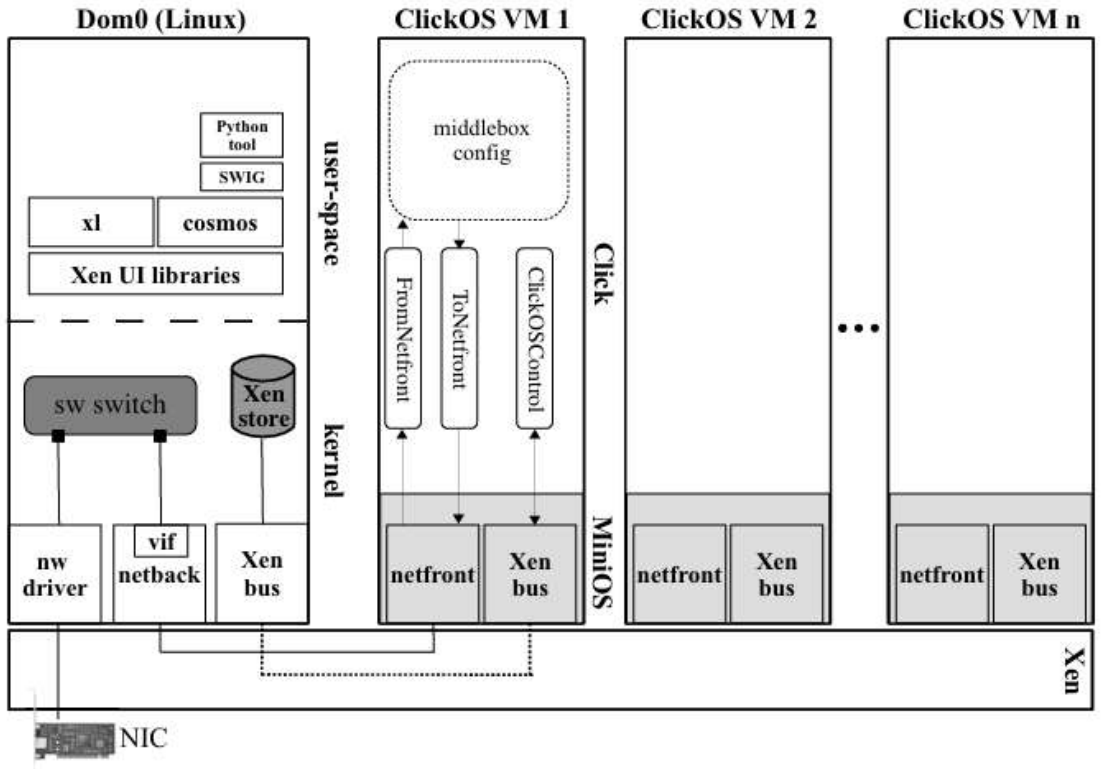
- ClickOS 虚拟机的构建过程包括读取配置、镜像文件并写入Xen Store一组条目
- 将虚拟机连接到后端交换机，使其与物理网卡建立连接
- MiniOS 启动后，会创建一个控制线程来监控虚拟机的配置文件，控制线程会在Xen存储库中创建安装条目，以便用户能够在ClickOS虚拟机中安装Click配置
- 专用控制线程会通过 Xen Store 进行配置安装，Xen Store 用于存储虚拟机的控制信息和配置



ClickOS Virtual Machines

构建并启动 ClickOS 虚拟机

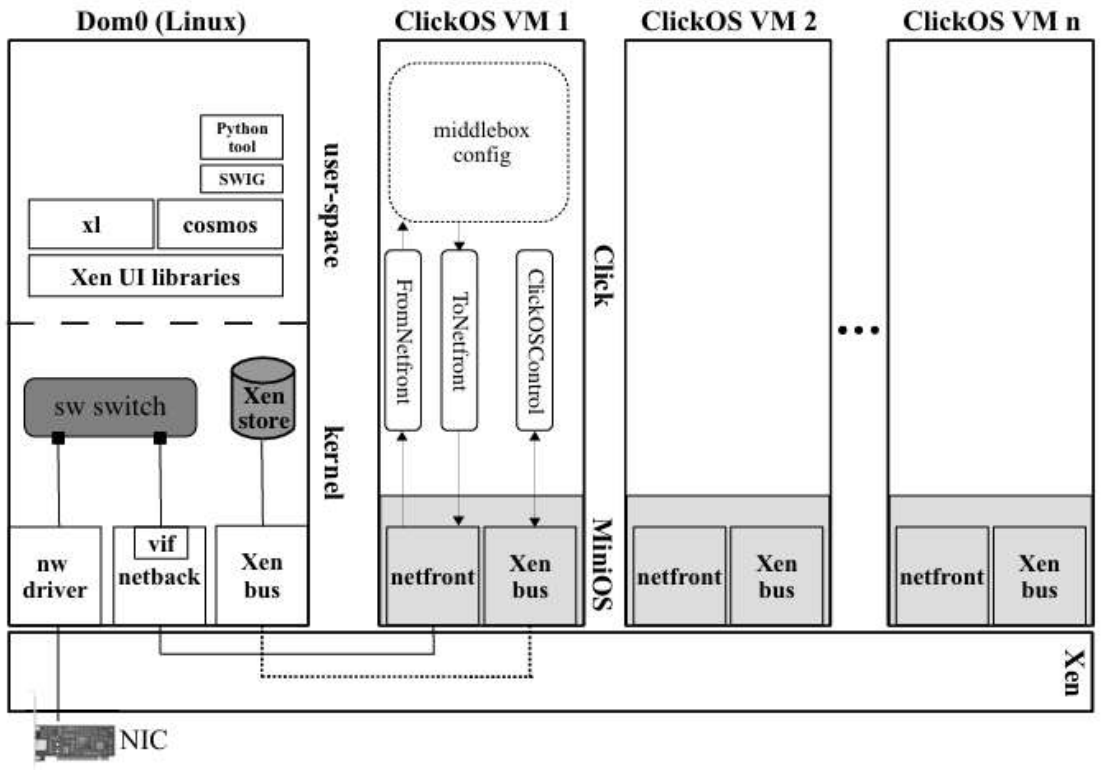
- 安装条目创建后，控制线程会对其设置监视以监控变更。当该条目被写入时，线程会启动第二个MiniOS线程来运行Click实例，使得单个ClickOS虚拟机内可运行多个Click配置
- 控制线程实时监控，当配置文件发生变化时，会动态地加载新的Click 配置，并启动相应的中间件处理
- 若要移除配置，只需向Xen Store条目写入空字符串即可



ClickOS Virtual Machines

Xen Store 与 Click 配置的交互

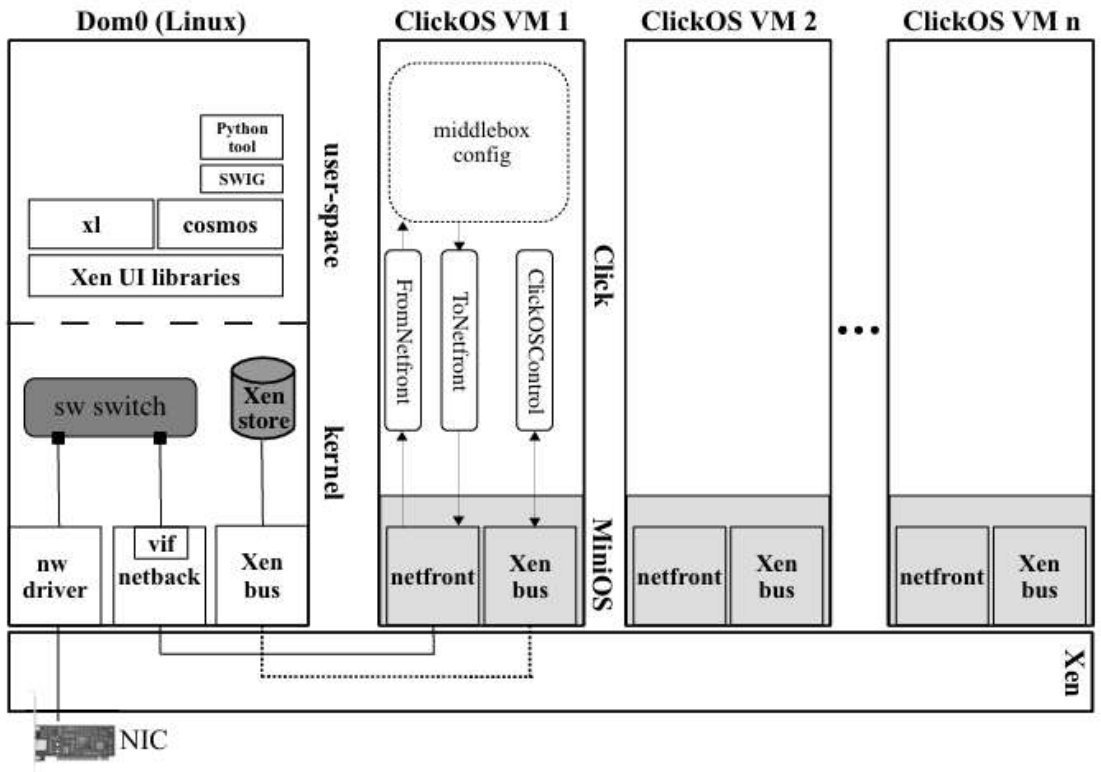
- 需要支持Click element handlers，这些handlers用于在element中设置和获取状态
- 利用Xen存储，针对每个虚拟机，我们在配置中为每个element及其handlers创建额外的条目
- 文章进一步开发了一个名为ClickOS Control的新Click element，该element会被透明地插入到所有配置中。该element负责一端与Xen存储上的读写操作交互，并将这些操作传达给Click内部相应的element handlers



ClickOS Virtual Machines

ClickOS 控制与管理工具

- 为了简化用户与 ClickOS 的交互，减少管理复杂性
- ClickOS 提供了名为 Cosmos 的控制管理工具，它基于 Xen 提供的管理接口，允许用户插入、删除与检查中间件配置
- Cosmos直接构建在Xen用户界面库之上，因此在处理请求时不会产生额外开销
- Cosmos 还支持 Python 脚本，以便进行自动化操作



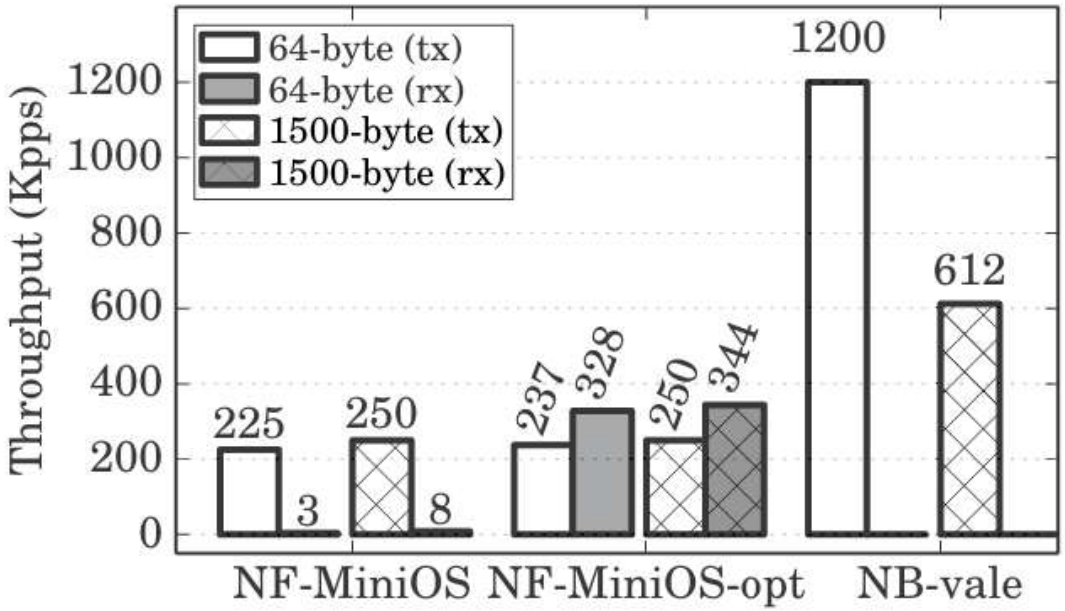
Outline

Xen Networking Analysis

- 为了找出 Xen 网络 I/O 子系统中的性能瓶颈
- 文章通过系统测试逐步分析虚拟网络的各个组件，包括：虚拟网卡驱动、软件交换机、后端驱动（netback）、前端驱动（netfront）等

测试环境与初步实验：

- 在一台普通服务器上进行（Intel Xeon E3 3.1GHz + 10Gb NIC）
- 使用 Xen 4.2、Open vSwitch 作为默认交换机，并运行一个 ClickOS 虚拟机

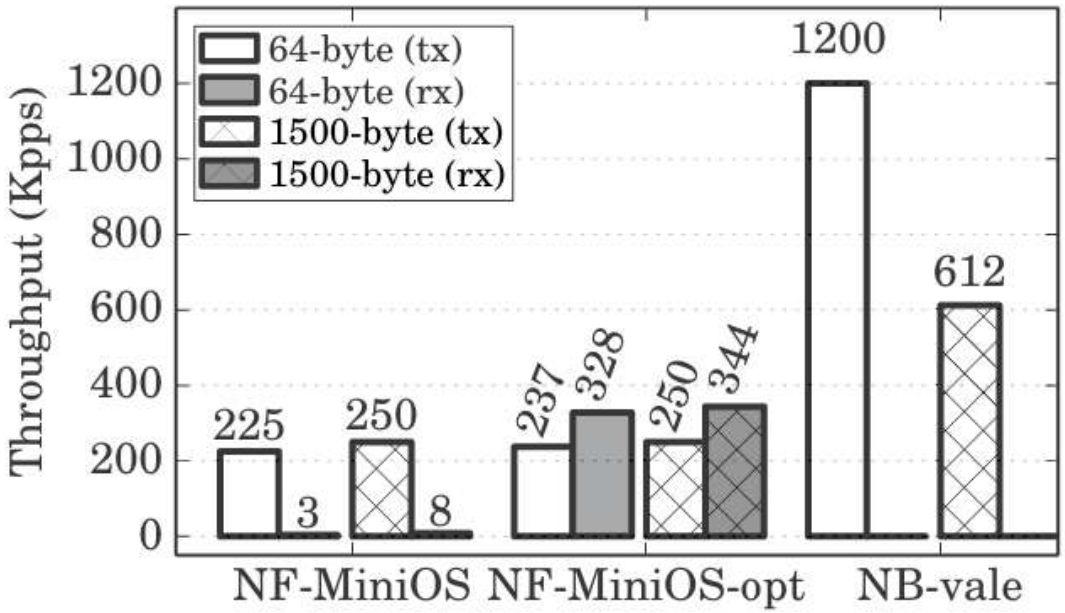


Xen Networking Analysis

- “NF-MiniOS”实验表明：MiniOS 的 netfront 驱动性能极差，尤其在接收方向，仅能处理约8 Kpps，远低于10Gb链路速率

为改进性能，文章优化了 MiniOS 的 netfront 驱动，引入了 内存 **grant 重用（grant reuse）** 技术：

- 默认模式下，每次数据包交换都要重新申请与释放共享内存（grant），造成大量 hypercall 开销
- 优化后，ClickOS 在初始化时一次性申请并复用缓冲区，减少系统调用次数，使接收速率显著提升
- 优化结果“NF-MiniOS-opt”：接收速率有显著提升，但仍存在瓶颈

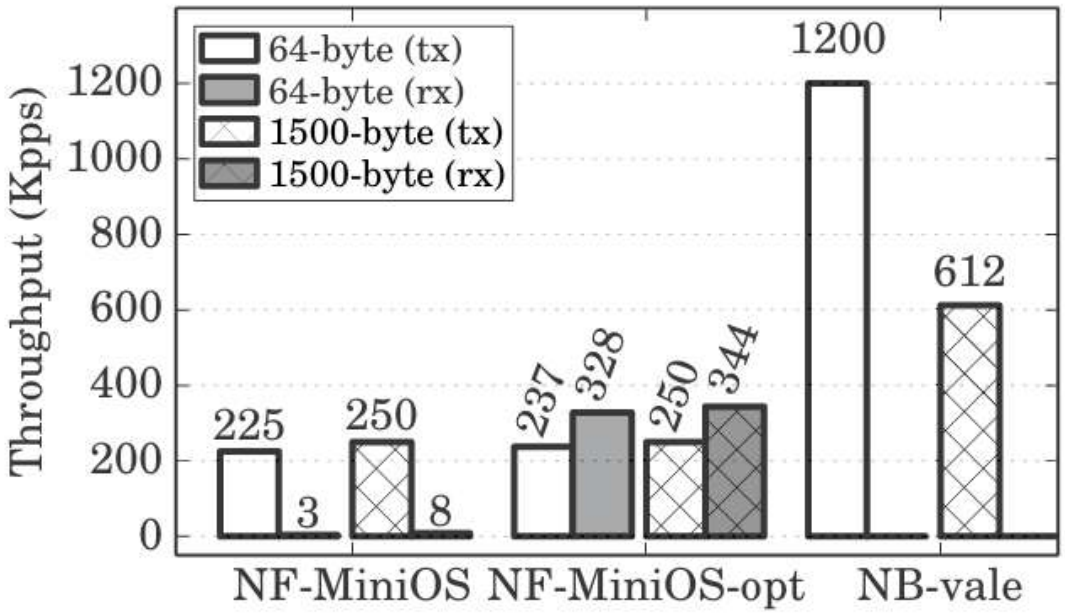


Xen Networking Analysis

- 上面结果表明：问题不只在驱动层，还可能出现在 Xen 的软件交换机和后端驱动（netback）中

分析Xen 的后端交换机制：

- Xen 默认使用 Open vSwitch（OvS） 作为虚拟交换机，测试发现 OvS 只能达到约300 Kpps 的性能上限
- 文章将其替换为 VALE —— 一种基于netmap的高性能虚拟交换机
- 优化结果“NB-vale”表明：64B数据包的传输速度显著提升至1.2 Mp/s，证实交换机至少是部分原因所在，但数据仍远未达到线路速率



Xen Networking Analysis

- 为深入探究，文章对netback驱动进行了逐函数分析，以确定主要性能损耗的来源
- Xen 的 I/O 管线中最大瓶颈来源于：
  - hypercall 调用频繁
  - 交换机与虚拟接口（vif）间存在多次数据复制
  - 使用Xen环形API，例如该API要求对所有双向传输的报文进行响应
  - sk_buff管理，这对于虚拟机将报文传输至网络后端并非必需

description	function	ns
get vif	poll_net_schedule_list	119
handle frags if any	netbk_count_requests	53
alloc skb	alloc_skb reserve_skb	384
alloc page for packet data	xen_netbk_alloc_page	293
build grant op struct	fills *gnttab_copy*	96
extends the skb with the expected size	__skb_put	96
build grant op struct (for frags)	xen_netbk_get_requests	61
add the skb to the Tx queue	__skb_queue_tail	53
checks for packets received	check_rx_xenvif	206
packet grant copy	HYPERCALL	24708
dequeue packet from Tx queue	__skb_dequeue	94
copy pkt data to skb	memcpy	90
put a response in the ring	fills *xen_netif_tx_response* notify_via_remote_irq	52
copy frag data	xen_netbk_fill_frags	179
calc checksum	checksum_setup	78
forward pkt to bridge	xenvif_receive_skb	3446

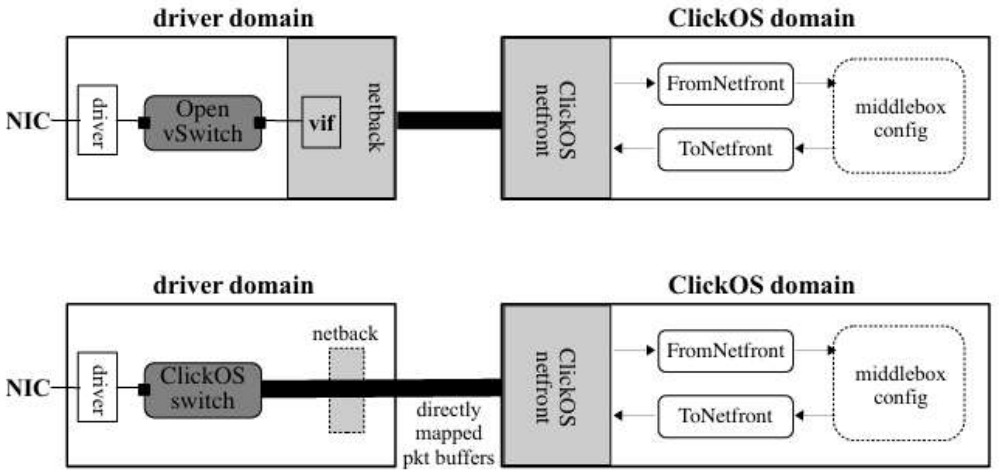
Outline

Network I/O Re-Design

- Xen 的原始网络管道包含多个非必要的中间层（如 vif、netback、switch），导致性能受限
- ClickOS 的设计目标是：打通数据通路，让数据包从网卡直接进入虚拟机内存空间

ClickOS 的网络I/O重设计主要包含三步：

- 替换默认交换机：使用高性能的 ClickOS switch 替代 Open vSwitch
- 移除netback层：只保留控制面，不再参与数据传输
- 映射缓冲区：将环形缓冲区（ring buffers）直接映射到 VM 的内存中

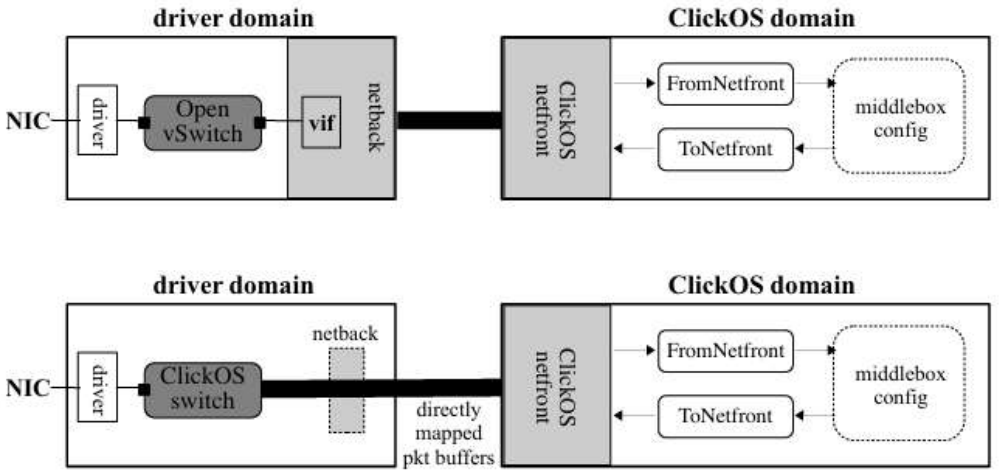


Network I/O Re-Design

ClickOS 专用交换机改进：

ClickOS 使用基于 VALE/netmap 的高性能虚拟交换机，并进行了扩展优化：

- 支持物理 NIC 直接连接到交换机
- 将交换端口数从 64 扩展至 256
- 允许每个虚拟机动态配置 ring 缓冲区大小（最多 2048 slots）
- 交换逻辑被改为静态 MAC 地址映射，以消除学习机制的性能开销



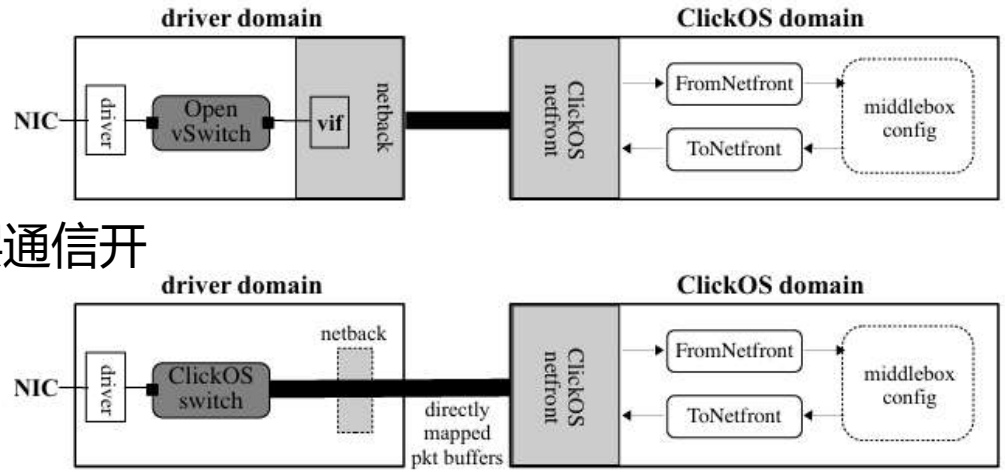
Network I/O Re-Design

netback 驱动重构：

ClickOS 将传统 netback 驱动改造成主要执行控制平面任务的轻量模块：

- 负责分配接收/发送缓冲区
- 设置内存 grant 并与前端共享地址
- 建立事件通道（event channel）以通知 VM。

  数据平面传输由交换机与netfront直接完成，从而移除了多层通信开销



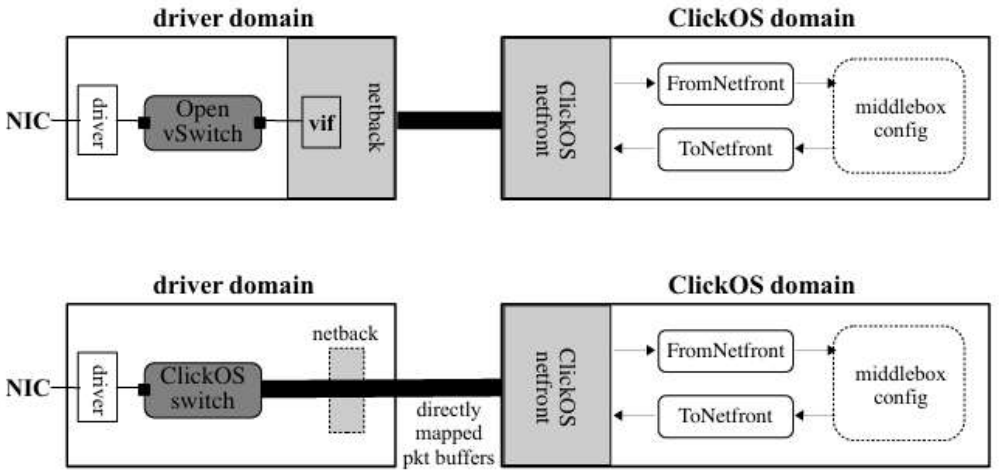
netback 驱动优化：

- ClickOS 采用内核线程与CPU核心1:1绑定的模型，这避免了公平性问题
- 发送（Tx）与接收（Rx）通道分离，使多核可并行处理双向流量，提升可扩展性

Network I/O Re-Design

Netfront驱动重构：

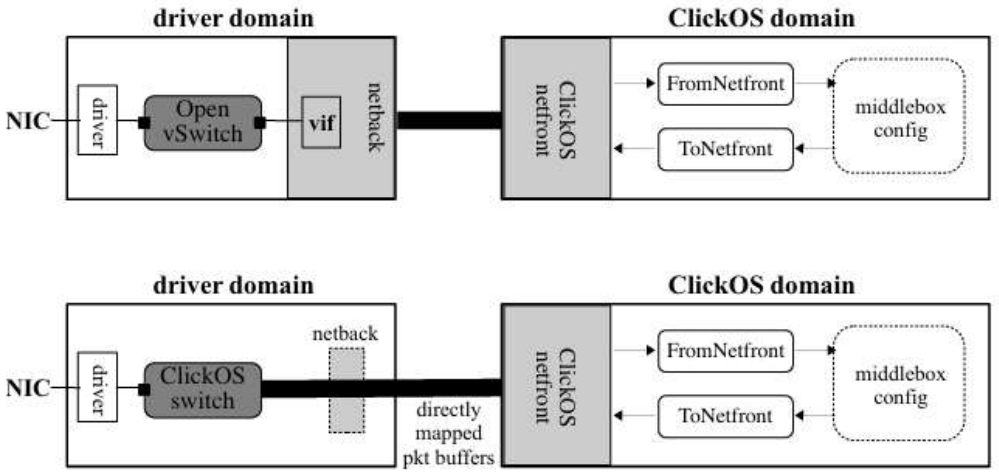
- 重新设计Netfront驱动，使其可以直接映射交换机暴露的 ring 缓冲区
- 加了对netmap API的支持，使得前端驱动具备与用户态程序一致的接口（open、mmap、poll等），极大地简化了开发与调试



Network I/O Re-Design

Netfront驱动优化：

- 异步发送 —— 减少阻塞与上下文切换
- grant 重用 —— 初始化时一次性分配缓冲区并持续使用
- 支持 Linux 前端 —— 新增 Linux 版本的 netfront 驱动，保证兼容性与性能一致



Outline

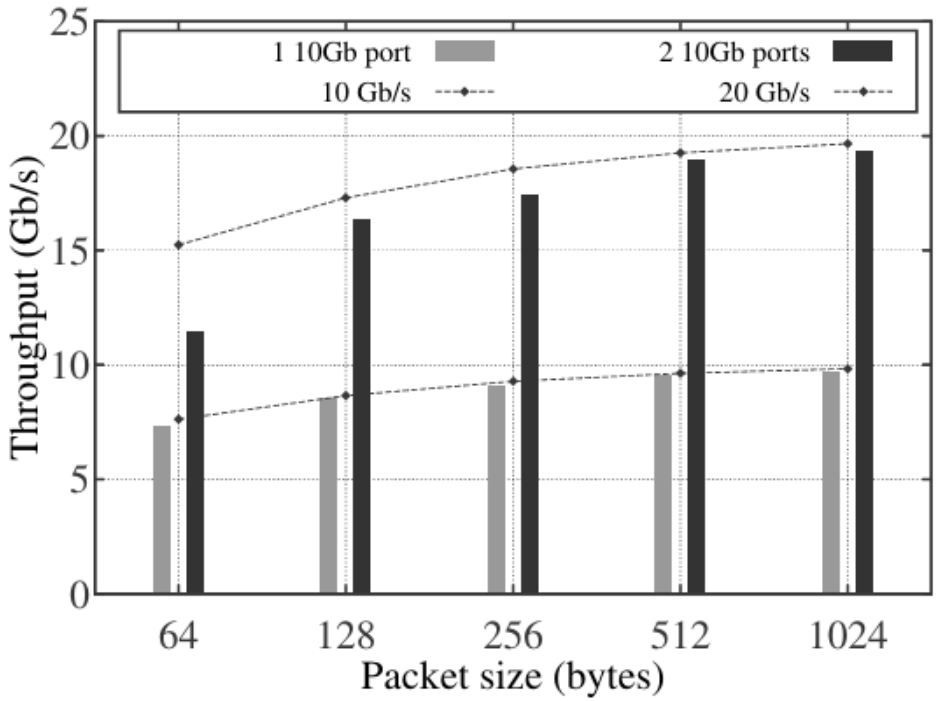
Base Evaluation

评估目标与测试平台

- 目的是评估 ClickOS 的基础性能，包括启动速度、延迟、吞吐率和系统扩展能力
- 低端单CPU Intel Xeon E3-1220服务器，配备4核3.1GHz处理器及16GB DDR3-ECC内存（大多数测试）
- 中端单CPU Intel Xeon E5-1650服务器，配备6核3.2GHz处理器及16GB DDR3-ECC内存（交换与扩展性测试）
- 用于 dom0 和 domU 的 Linux 3.6.10、Xen 4.2.0、Click 2.0.1，以及用于数据包生成和速率测量的 netmap 的 pkt-gen 应用程序
- ClickOS 交换机：运行pkt-gen的用户空间进程向交换机生成数据包，并通过单个10Gb/s以太网端口传输；随后由一台独立的低端服务器再次使用pkt-gen接收数据包并测量速率。接着增加另一组pkt-gen用户进程和10Gb/s以太网端口以测试扩展性

Base Evaluation

- 使用一个和两个10 Gb/s网卡端口时的ClickOS交换机性能
- 在单端口对情况下，该交换机在所有数据包尺寸下都能使10Gb/s管道达到饱和。对于双端口对情况，除最小尺寸数据包（达到线速的70%）外，交换机在所有数据包尺寸下都能占满总计20Gb/s的管道容量



Base Evaluation

内存占用

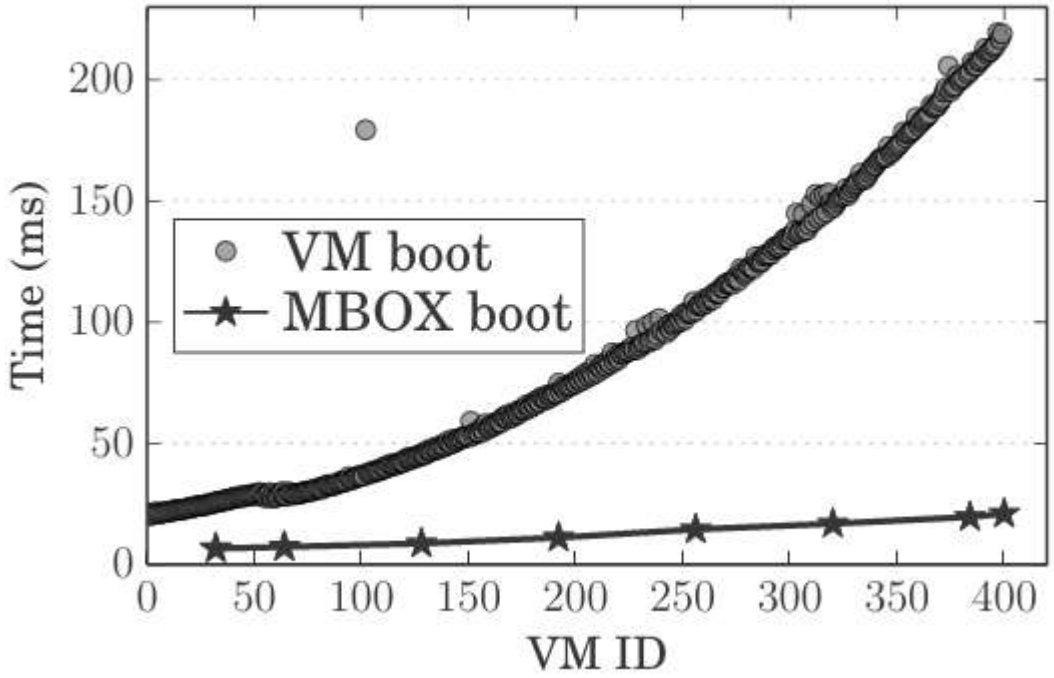
- 需要分配一定内存用于netmap环形数据包缓冲区。具体内存需求取决于环形缓冲区的大小
- 单个ClickOS虚拟机可能需要处理非常高包率的可能性不大，因此在实践中，较小的环尺寸可能就足够了

Ring size	Required memory (KB)	# of grants
64	264	65
128	516	129
256	1032	258
512	2064	516
1024	4128	1032
2048	8260	2065

Base Evaluation

启动时间

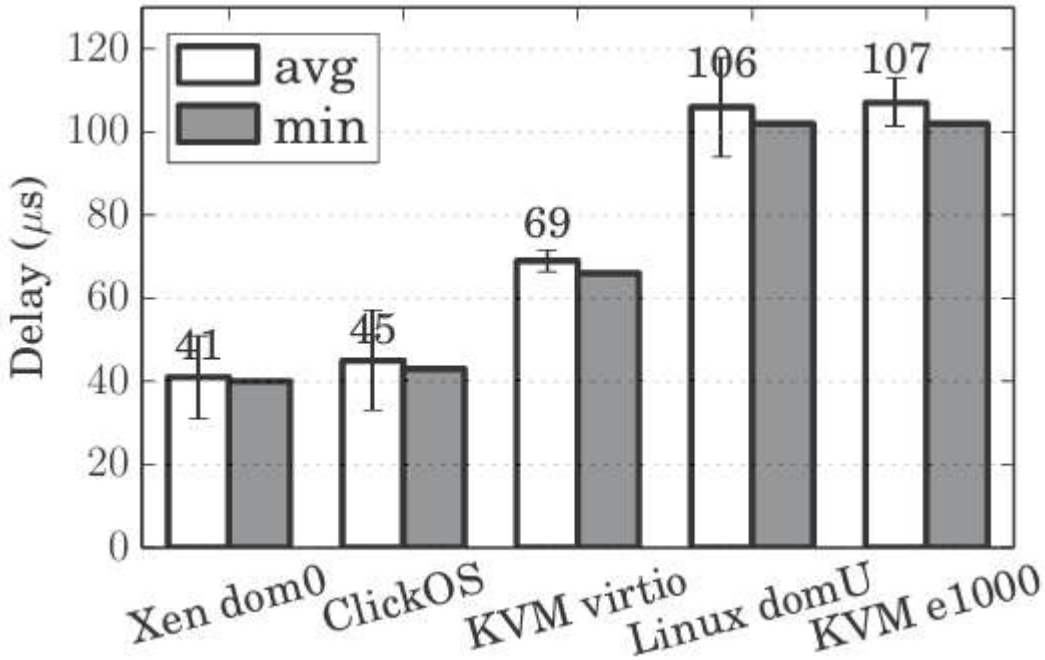
- 着虚拟机数量的增加，启动时间和初始化时间均呈上升趋势，第400个虚拟机的启动时间最长达到219毫秒，初始化时间最长达到20.0毫秒
- 这种增长源于Xen存储器的资源争用问题，该问题存在优化空间



Base Evaluation

延迟

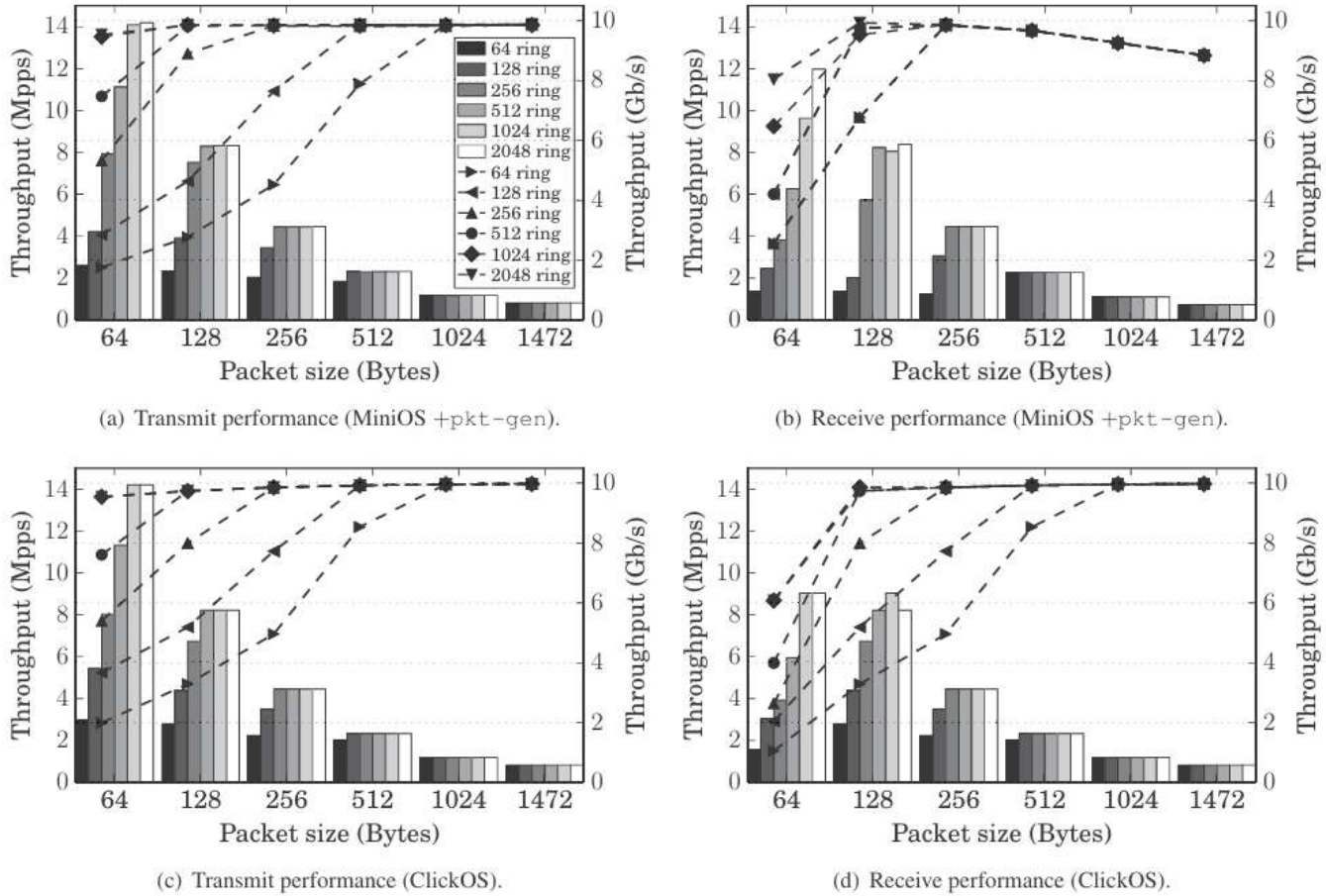
- ClickOS、Linux Xen虚拟机、dom0以及使用e1000或virtio驱动的KVM的空闲虚拟机ping延迟
- dom0的延迟仅为41微秒，因为它无需经过netback和netfront驱动程序的额外开销
- 而在测量标准未优化的Xen Linux虚拟机netback/netfront驱动程序时，这种开销确实存在



Base Evaluation

吞吐量

- 在单个 CPU 核心上，当调整环形缓冲区大小时，运行在 MiniOS/ClickOS 之上的单个虚拟机数据包生成器的性能表现



(a) Transmit performance (MiniOS +pkt-gen).

(b) Receive performance (MiniOS +pkt-gen).

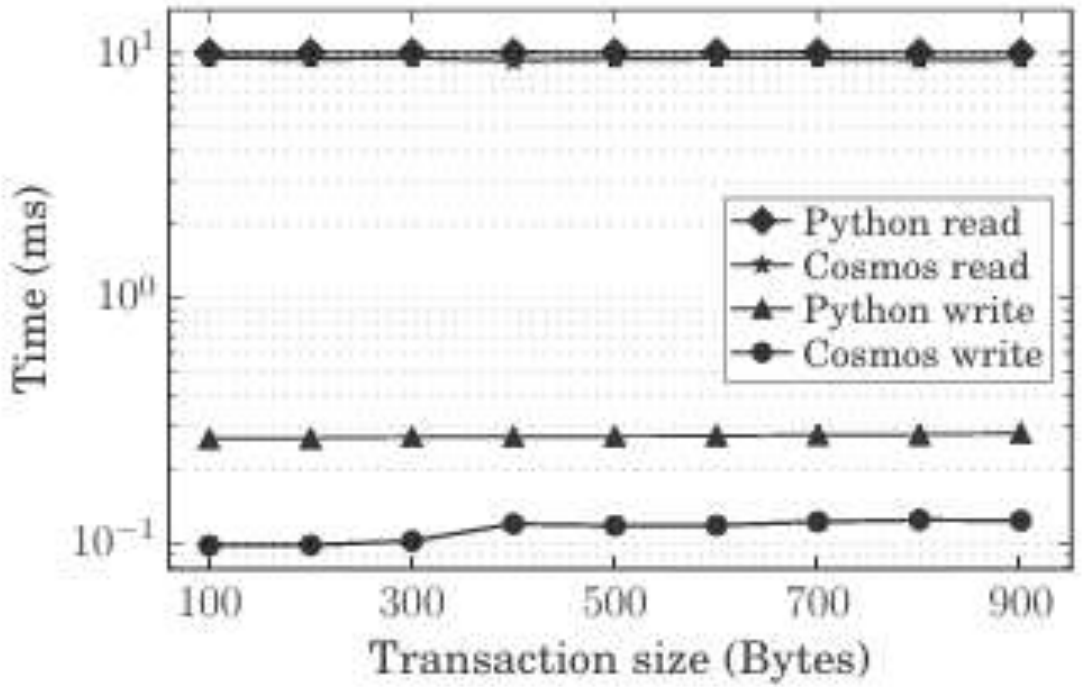
(c) Transmit performance (ClickOS).

(d) Receive performance (ClickOS).

Base Evaluation

状态插入

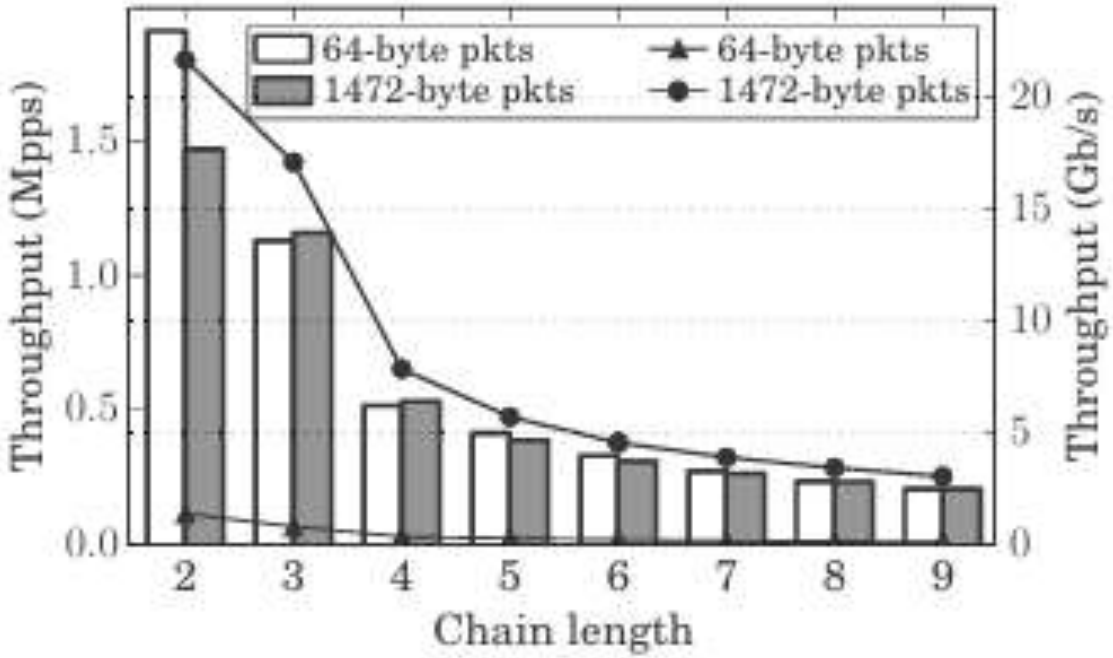
- 不同事务大小下的 ClickOS 中间件状态插入（写入）和检索（读取）
- 数值波动很小



Base Evaluation

链接

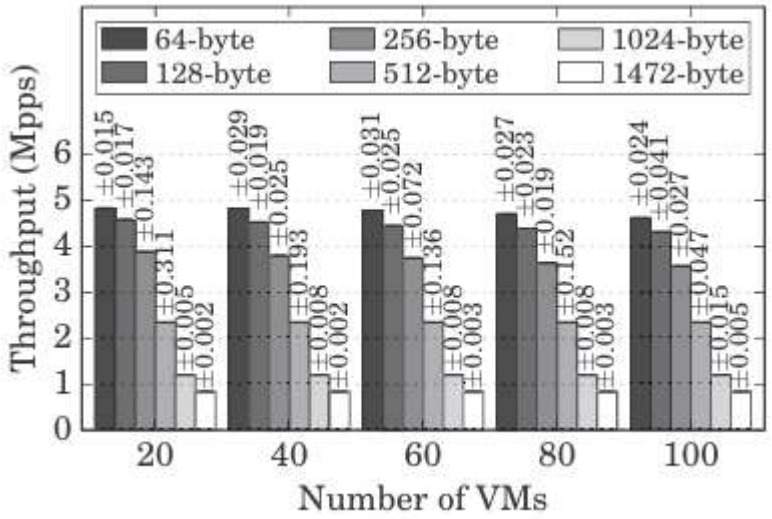
- 实例化一个ClickOS虚拟机以尽可能快地生成数据包，另一个用于测量数据包，并逐步增加仅作转发用途的中间ClickOS虚拟机数量
- 链式结构越长传输速率越低

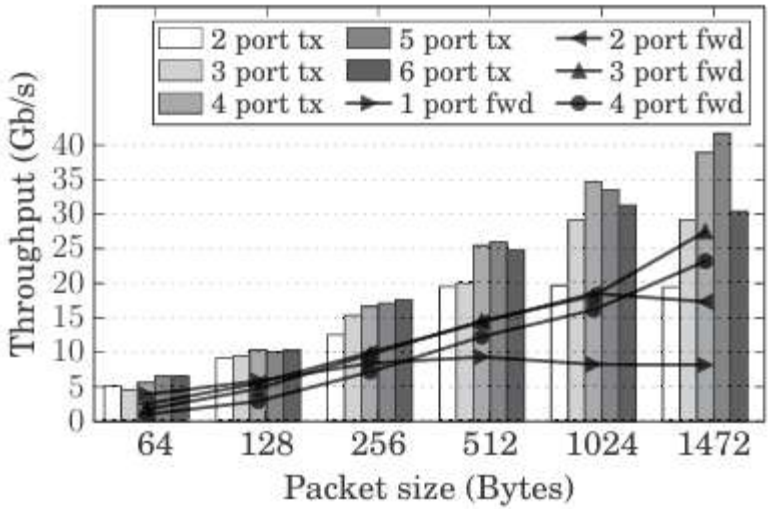


Base Evaluation

扩展

- 在一个核心和一个10 Gb/s端口上运行多个ClickOS数据包生成器虚拟机
- 使用多个10 Gb/s端口且每个端口对应一个ClickOS虚拟机时的累计吞吐量，场景为（1）发送流量（tx）或（2）转发流量（fwd）



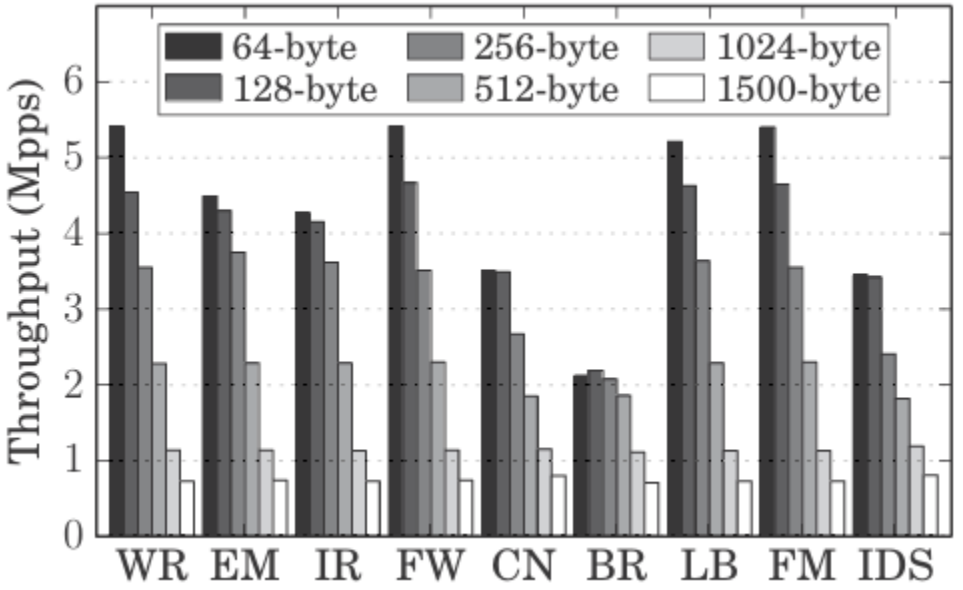


Outline

Middlebox Implementations

- 目标是验证 ClickOS 能否支持多种不同类型的中间件功能
- 实现并测试了五种典型中间件：防火墙（Firewall）、NAT、负载均衡器（Load Balancer）、IDS（入侵检测系统）、BRAS（宽带远程接入服务器）
- 这些中间件功能覆盖了 NFV 的主要应用场景，包括安全、流量管理和接入服务

- 用单CPU核心时不同ClickOS中间件及数据包大小的性能表现
- ClickOS表现优异，在512字节及更大尺寸的数据包测试中，几乎所有配置都能达到接近线速的水平。对于更小尺寸的数据包，其线速占比虽有所下降，但ClickOS仍能保持每秒数百万包的处理能力



Outline

Review

- ClickOS 实现了一个高性能、轻量化、虚拟化的中间件平台，彻底改变了网络功能虚拟化（NFV）的性能瓶颈现状
- 通过对 Xen 网络 I/O 架构的深入重构，ClickOS 在虚拟化环境中实现了接近裸机性能的网络转发能力
- ClickOS 的创新主要体现在三个方面：
  - 极简虚拟机设计 —— 采用 MiniOS，去除多余功能，仅保留网络处理核心
  - I/O 通道重构 —— 实现零拷贝、事件分离与缓冲区共享
  - 模块化功能框架 —— 基于 Click element化逻辑，快速组合多种中间件

Review

- ClickOS 并非实验性原型，而是一个可实际部署的工程系统。其模块结构已与 Xen 和 netmap 框架集成，可在商用服务器上直接运行
- 未来优化方向，包括：
  - 支持更复杂的多核调度与并行处理
  - 扩展对 SR-IOV 等硬件虚拟化技术的兼容
  - 改进 ClickOS 管理框架 Cosmos，实现动态服务迁移
  - 探索与 SDN（软件定义网络）的深度融合