OVERVIEW OF VIRTUALIZATION TECHNOLOGIES AND THEIR IMPLICATIONS ON FUTURE INTERNET

Future Internet Camp
Summer 2007
Sue Moon

Tutorial Overview

- Why do we need a testbed?
- What has been done?
  - Click
  - Xorp
  - Xen
  - PlanetLab
- What will we build?

Tutorial Overview

- Why do we need a testbed?
- What has been done?
  - Click
  - Xorp

1hr

10 min break

  - Xen
  - PlanetLab
- What will we build?

Why do we need a testbed?

- Utilities of a testbed
  - rigorous, transparent, and replicable testing of scientific theories, computational tools, and other new technologies. [Wikipedia]
- What are testbeds we had and have?
- What are emerging needs?

Testbed as a trial platform

- ARPAnet
  - packet switching
- MBone, ABone, XBone
  - Multicast, active networking, overlay networks
- NIMI (National Internet Measurement Infrastructure)
  - Many-point measurement experiment

Research Infrastructure

- vBNS/Abilene
  - interconnecting research sites
- KOREN/KREONET
  - interconnecting research sites
  - IPv6, mobile IP, HDTV

Emerging Needs

- Competition for global services
  - CDNs
  - P2P-everything
- Many technology choices for a single service
  - IPTV
    - Core-multicast + edge-multicast
    - Core-multicast + edge-multicast + p2p-assisted
  - VoD
    - Multicast/Unicast + p2p-assisted
- Testbed for new services

New Testbed

- Must have global coverage
- Has little overhead in join and use

⇒ PlanetLab

Review

- Click
  - Open source router project (forwarding)
- Xorp
  - Open source router project (routing)
- Xen
  - Efficient virtualization

THE CLICK MODULAR ROUTER

R. Morris, E. Kohler, J. Jannotti, F. Kaashoek

SOSP 1999

Courtesy of Changhyun Lee

Motivations

- Routers are expected to be
  - Prioritizing, filtering packets, acting as firewalls, etc.
  - Flexible to changes (e.g. dropping policy)

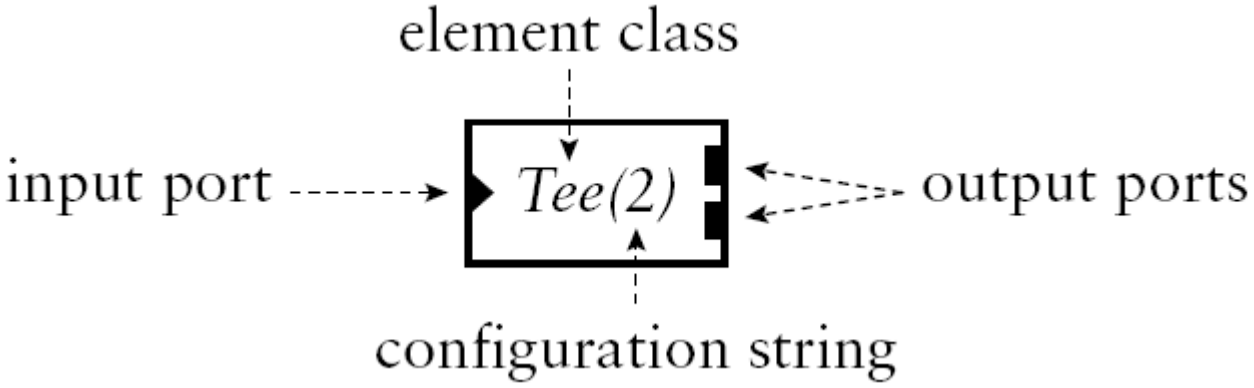
- Limitations of current routers
  - Closed, static, and inflexible design
  - Functions cannot be specified
  - No space for extensions

Solution: Click Modular Router

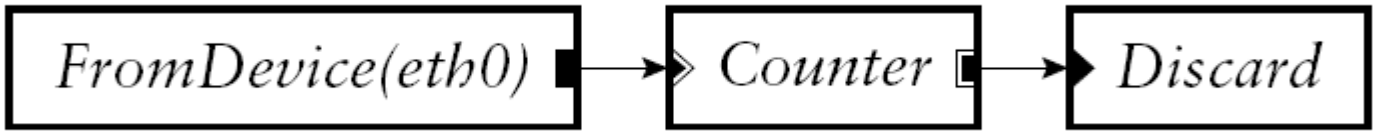
- Click: Software architecture for building routers
  - Elements
    - A unit of router processing
  - Simple mechanisms
    - To build a router
      - User connects a collection of elements into a graph
    - To extend a configuration
      - User writes new elements or compose existing ones
- Two specific features
  - Pull processing
  - Flow-based router context

Overview

- A sample element for copying packets

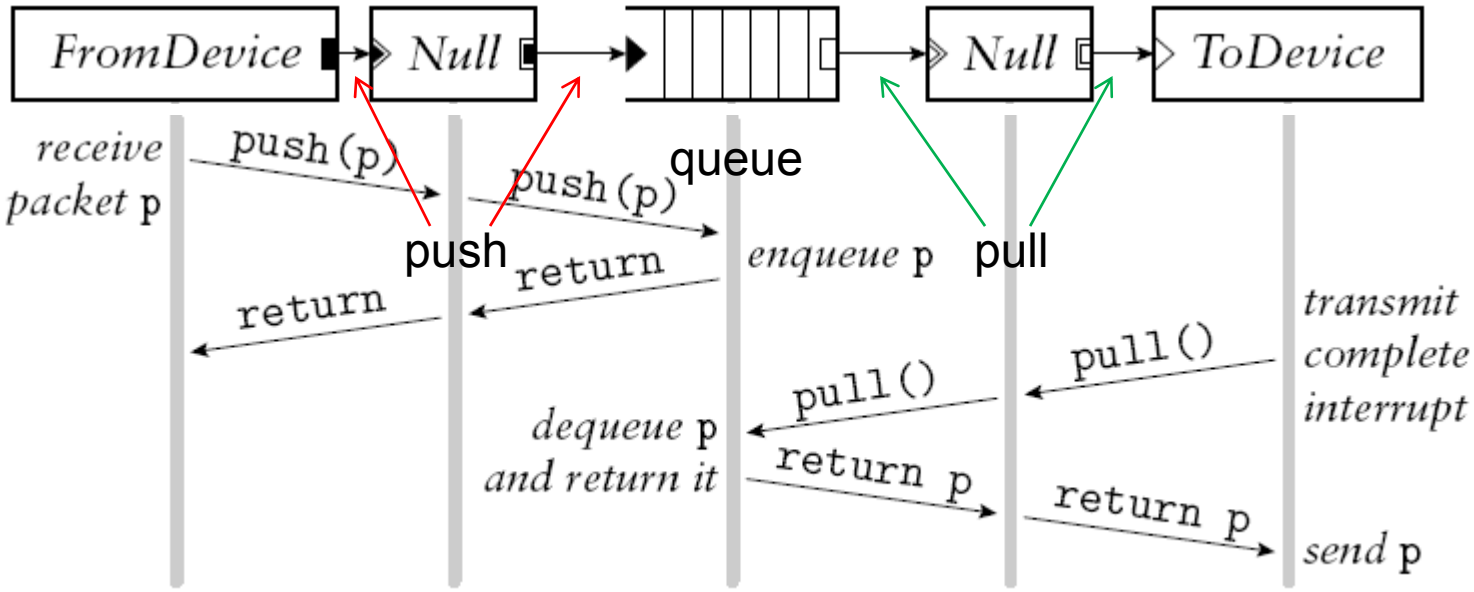


- A sample router configuration using 3



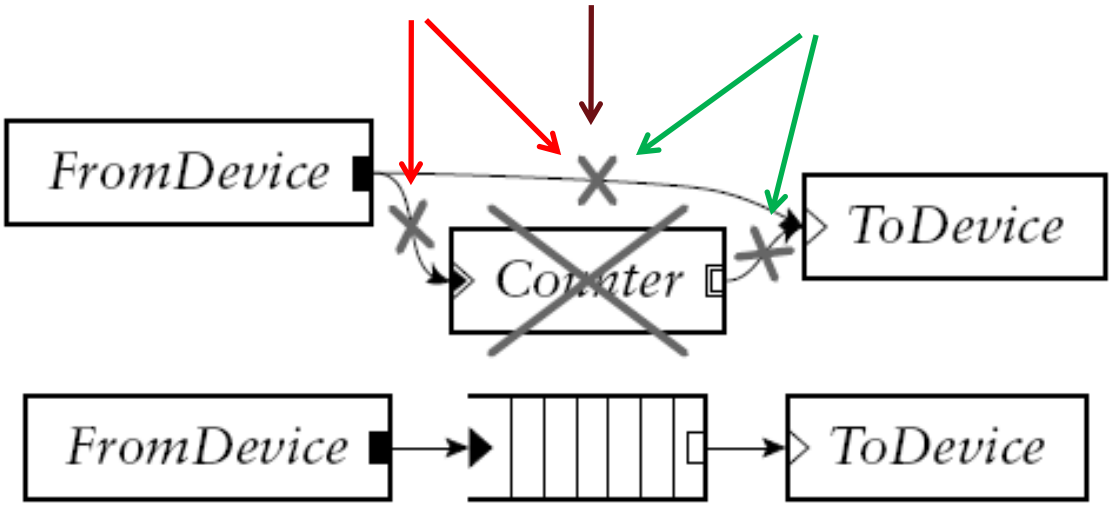
Push and Pull Processing

- Push and pull call initiate packet transfer
  - In a push connection (black ports), the upstream element hands a packet to the downstream element
  - In a pull connection (white ports), the downstream element asks the upstream element to return a packet



Push and Pull Processing (cont'd)

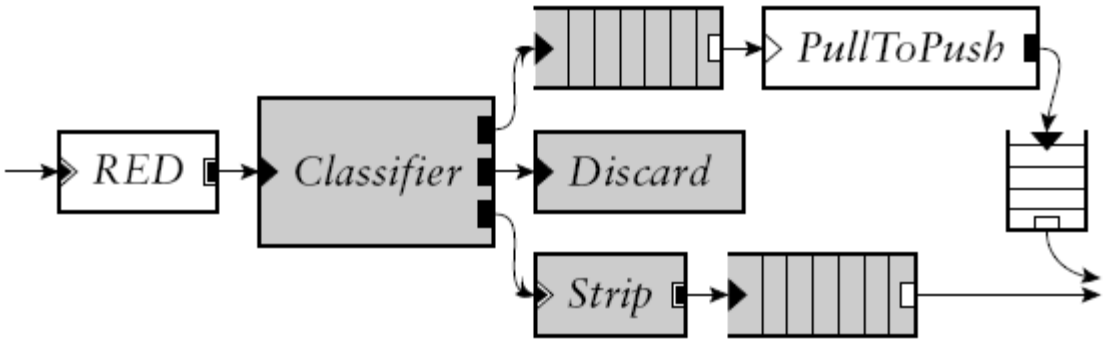
- Invariants are automatically checked during router initialization



Examples of invariant violations

Flow-based Router Context

- Some elements need to find others that might not be directly connected to them
  - Downstream, upstream search during initialization
    - E.g. RED, packet-upstream notification for Queue



Examples of search from RED

Language

- Click Configurations are written in a simple textual language with two constructs
  - Declarations: what elements are created
  - Connections: how elements are connected

```
# a trivial router that drops everything
src :: FromDevice(eth0);
ctr :: Counter;
sink :: Discard;
src -> ctr;
ctr -> sink;

# the same, with anonymous elements
FromDevice(eth0) -> Counter -> Discard;
```

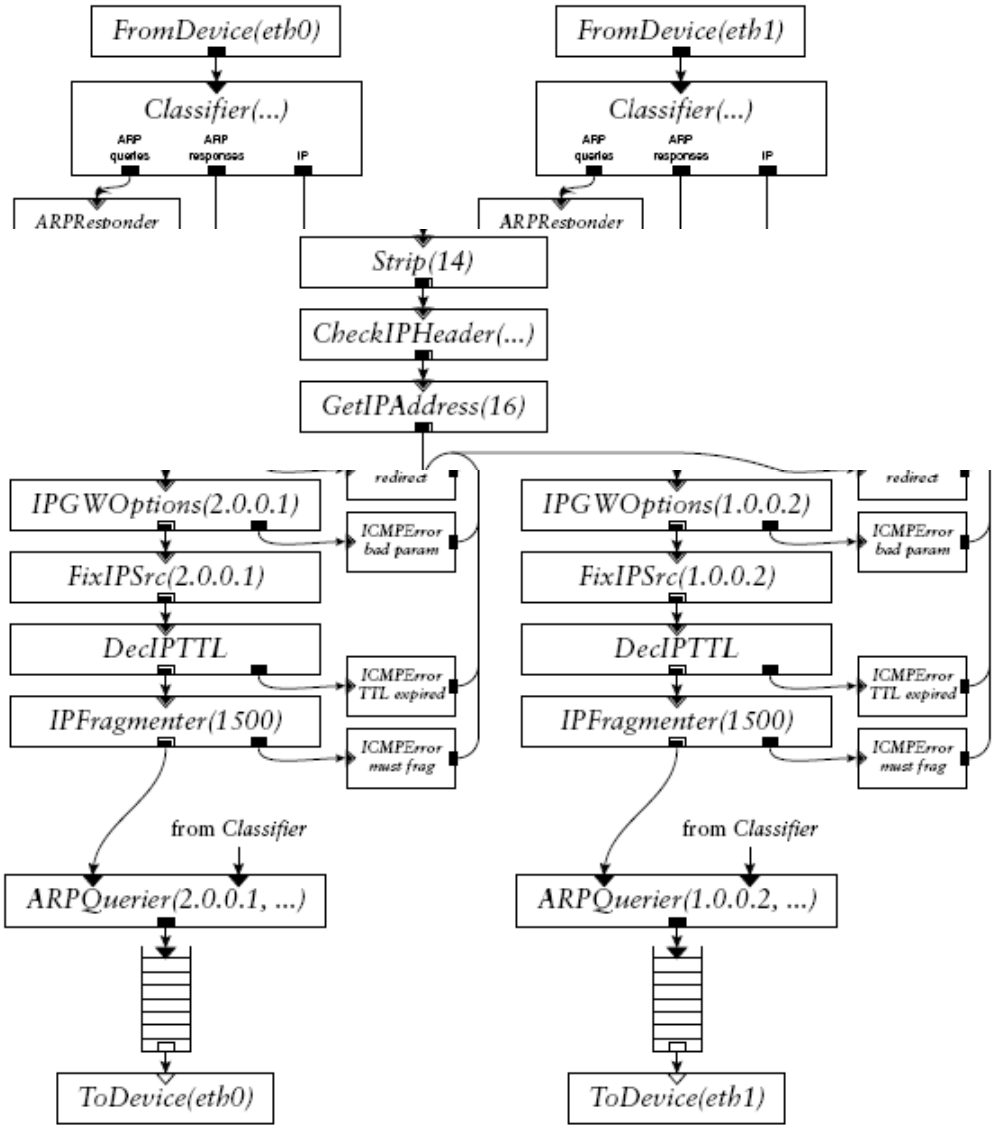
Simple example of configuration

Primitive Elements

- There are many…

ARPQuerier(...)
ARPResponder(x y)
CheckIPHeader(...)
CheckPaint(p)
Classifier(...)
DecIPTTL
Discard
DropBroadcasts
EtherSpanTree(...)
EtherSwitch
FixIPSrc(addr)
FromDevice(device)
GetIPAddress(...)
HashDemux(...)
ICMPError(type, code)
IPEncap(...)
IPFragmenter(mtu)
IPGWOptions
LookupIPRoute
Meter(r)
Paint(p)
PrioSched
Queue(n)
RED(...)
RoundRobinSched
SetIPDSCP(c)
Shaper(n)
Strip(n)
Suppressor
Tee(n)
ToDevice(device)
ToLinux

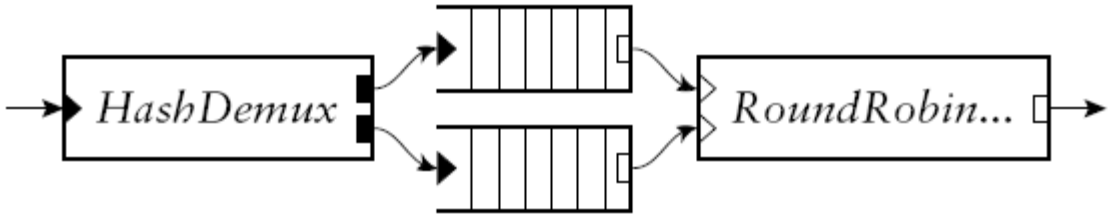
Example IP Router



Click Extensions

Scheduling

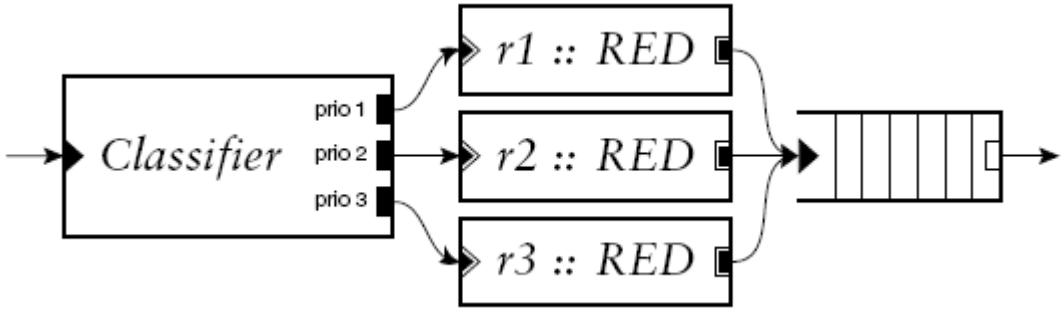
- Packet scheduler can be implemented as a single element
  - A pull element with multiple inputs and one output
    - E.g. round-robin scheduling, priority scheduling



Example of fair queue

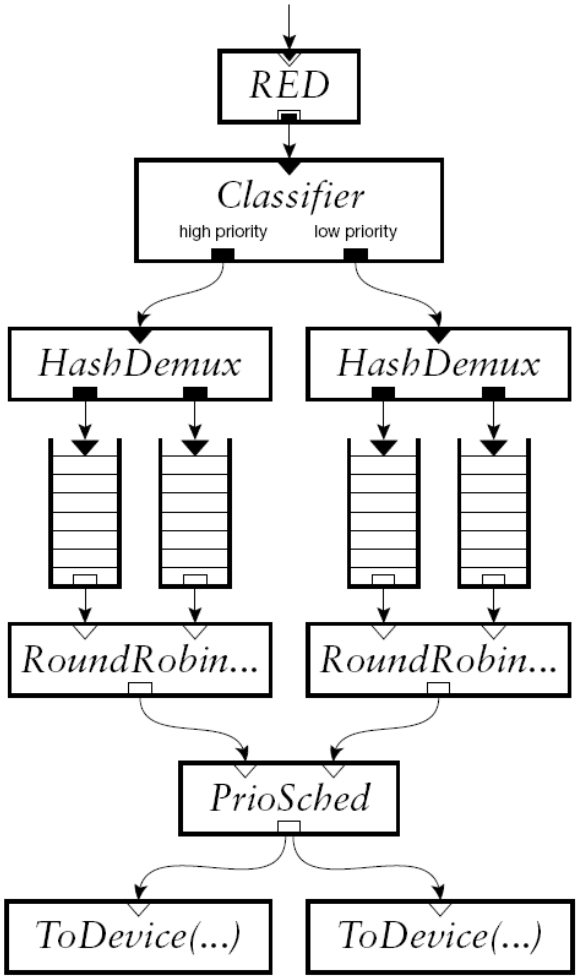
Dropping Policies

- Various dropping mechanisms can be implemented with click's flexibility

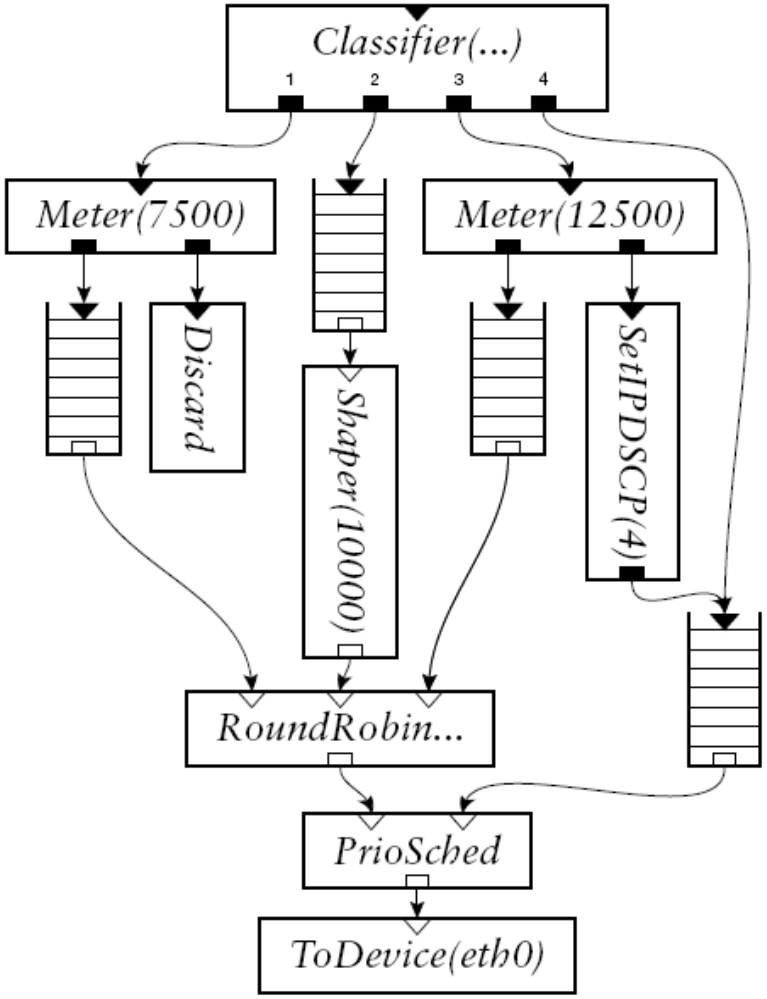


Example of weighted RED

Example of combination



Differentiated Services



- Combinations of classification, tagging, shaping, dropping, queuing, and scheduling functions
- Separates incoming traffic into 4 streams, based on the IP DSCP

Limitations

- Small elements are not appropriate for all problems
  - Since Click relies on packet flow, large elements are required when control flow doesn't match packet flow
- Difficult to implement shared objects that don't participate in packet forwarding
  - E.g. routing table
- Mechanism to schedule CPU time among competing push and pull paths is needed

Implementation

- Linux kernel environment
  - Linux networking code passes control to Click when
    - Packet arrives
    - Network interface becomes ready to send
    - Timer expires
  - Linux runs Click code in a bottom-half handler
    - Network device receives a packet
    - Device copies the packet into buffer and interrupts
    - Interrupt code appends the buffer to an input queue
    - Wakes up the bottom half
    - Bottom half passes packets to *FromDevice*

Performance Evaluation Summary

- Click forwards 73,000 packets/sec
  - Only 10% slower than Linux
  - Outperforms some commercial routers

    (e.g. Cisco 2621 router's forwarding rate is 25,000 packets/sec)

- Adding a new element is cheap enough
  - Click is a practical solution with flexibility

Conclusion

- Click modular router
  - A software router framework
  - open, extensible, and configurable
  - Just 10% slower than Linux 2.2.10
  - Impose a low cost for incremental additions to configurations
  - Current v1.5 (May 2006)

XORP: AN OPEN PLATFORM FOR NETWORK RESEARCH

M. Handley, O. Hodson, E. Kohler

1st HotNets Workshop

Courtesy of Yoonchan Choi

What is XORP?

- Set of routing protocol implementations, an extensible programming API, and configuration tools
- XORP does not implement forwarding system
  - Use Click

Architecture and Requirements

- Routing infrastructure
  - At least partially open to research extensions
  - How to make it
    - Individual researchers could convince big router vendors to implement their extensions
    - Vendors could open their internal APIs
    - Researchers could deploy currently available routing daemon such as Zebra or GateD, on a conventional PC running on OS such as Linux or FreeBSD
    - Researchers could develop a new open-source router

Architecture and Requirements

- Four major challenges
  - Features
    - Routing protocols, management interfaces, queue management, and multicast
  - Extensibility
    - Every aspect of the router should be extensible
    - Multiple extensions should be able to coexist
  - Performance
    - Forwarding performance
    - Scalability in routing table size
  - Robustness
    - Must not crash or misroute packets

XORP Overview

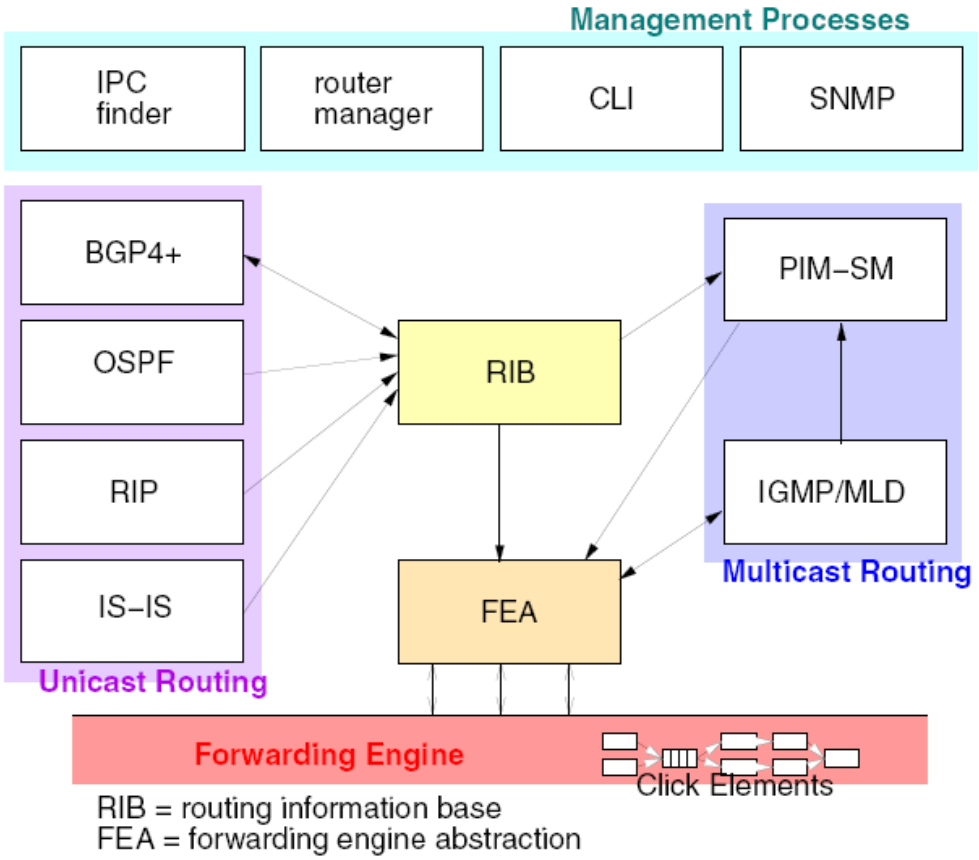
- Higher-level subsystem
  - Called “user-level”
  - Consist of the routing protocols themselves
  - Along with routing information bases and support processes
  - A multi-process architecture
    - With one process per routing protocol
    - Plus extra processes for management, configuration, and coordination
  - Communicate between these modules using XORP Resource Locators (XRLs)

XORP Overview

- Lower-level subsystem
  - Initially run inside an OS kernel
  - Manage the forwarding path
  - Provide APIs for the higher level to access
  - Use the Click modular router

XORP Overview

- XORP Architecture



XORP Overview

- Four core processes
  - Router manager process
  - Finder process
  - Routing information base process (RIB)
  - Forwarding engine abstraction process (FEA)

XORP Overview

- Router manager process
  - Manage the router as a whole
  - Maintain configuration information
  - Start other processes
  - Restart failed processes as necessary

XORP Overview

- Finder process
  - Store mappings between abstracted application requests and the particular IPC calls necessary to answer those requests
  - Regarded as an IPC redirector

XORP Overview

- Routing information base process (RIB)
  - Receive routes from the routing processes
  - Arbitrate which routes should be propagated into the forwarding path, or redistributed to other routing processes

XORP Overview

- Forwarding engine abstraction process (FEA)
  - Abstract the details of how the forwarding path of the router is implemented
  - Manage the networking interfaces and forwarding table in the router
  - Provide information to routing processes about the interface properties and events occurring on interfaces

Solving Design Challenges

- Traditional router features
  - The minimal list of routing and routing-support protocols
    - BGP4+, OSPF, RIPv2/RIPng, Integrated IS-IS, PIM-SM/SSM, IGMPv3/MLD, PPP
  - With the exception of IS-IS, all of these are currently being worked upon within XORP

Solving Design Challenges

- Extensibility
  - Open interfaces are the key to extensibility
  - XORP's design encourages the construction of useful interfaces through multi-process design
  - Open inter-process interfaces form the basic source of user-level XORP's extensibility
  - IPC within XORP takes place via XORP Resource Locators (XRLs)

Solving Design Challenges

- Performance
  - The forwarding path must touch every packet; performance is paramount
  - Click provides a high-performance forwarding path in the kernel

Solving Design Challenges

- Robustness
  - Routing processes are protected from each other
  - They can have their resources constrained according to administrative preference
  - If a routing protocol does crash
    - The RIB will remove its routes from the forwarding engine

XORP Summary

- Internet research is being frustrated by an inability to deploy experimental router software at points in the network
- XORP aims to be both a research tool and a stable deployment platform, thus easing the transition of new ideas from the lab to the real world

XEN AND THE ART OF VIRTUALIZATION

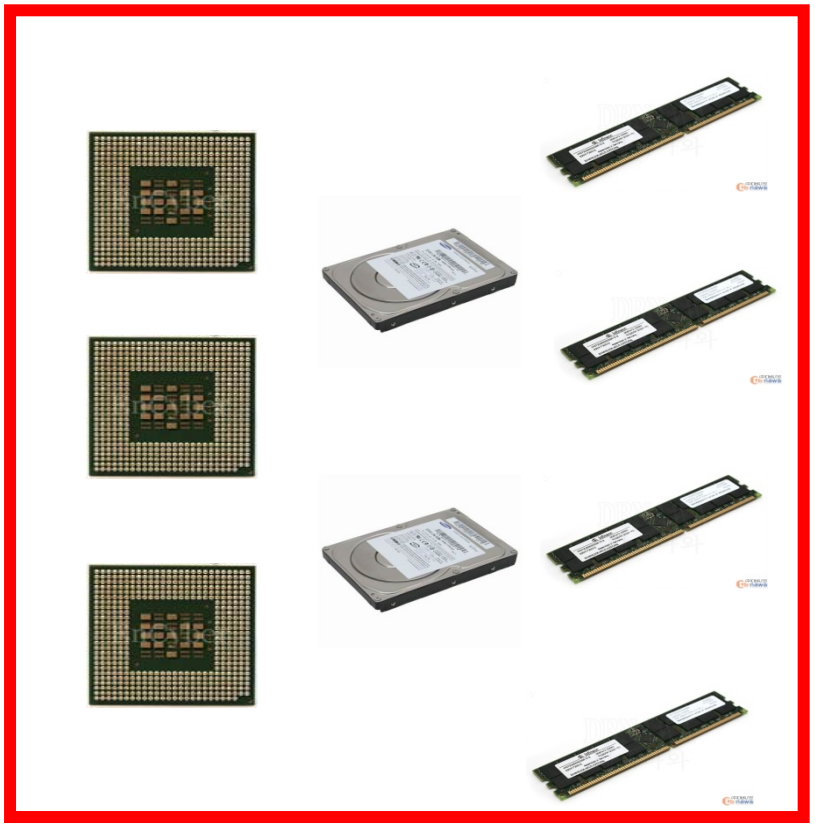
P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, A. Warfield

SOSP 2003

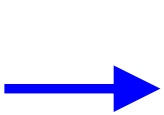
Courtesy of Keon Jang

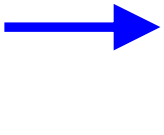
Partitioning a physical machine

Physical Machine



Virtual Machines





Why virtualize?

- Problems of process level parallelism
  - System administration a burden
  - Performance isolation not adequate
    - Scheduling priority, memory demand, network, disk access
- Low-level multiplexing can mitigate this problem

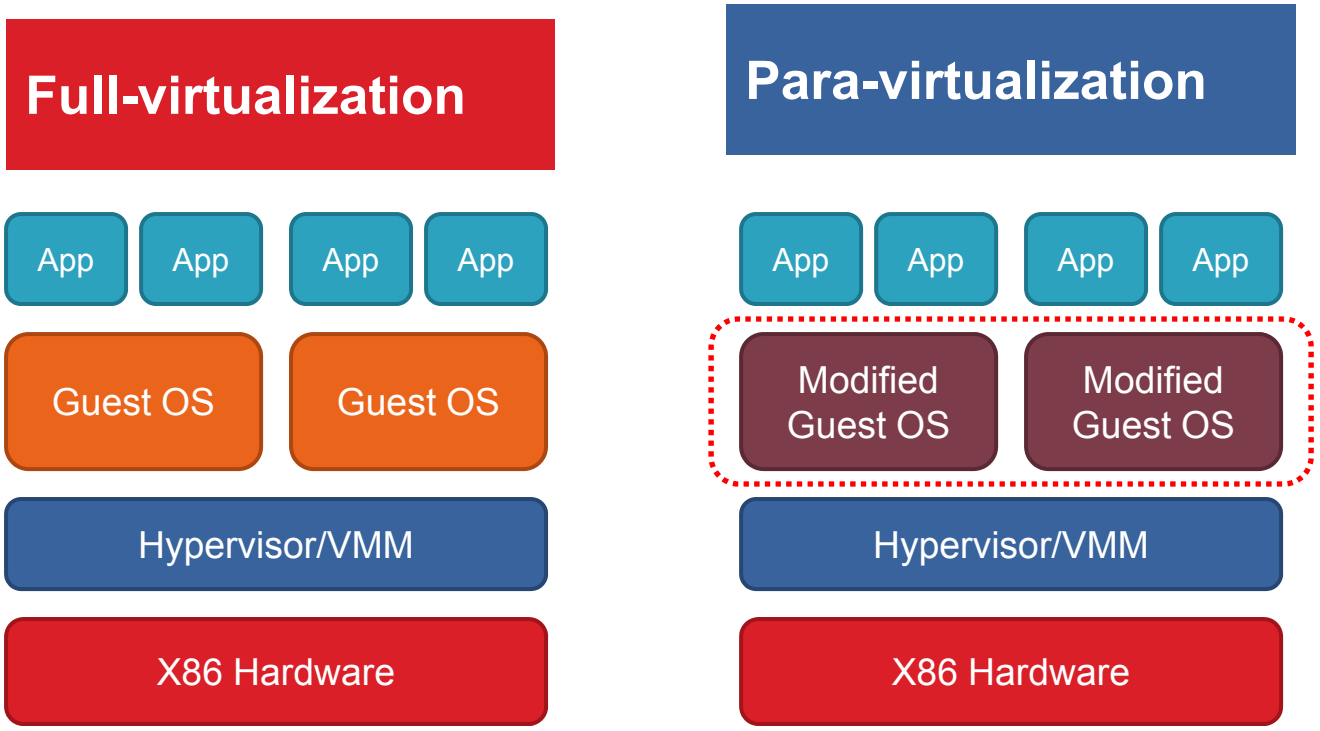
Why virtualize now?

- Modern computers are sufficiently powerful to use virtualization
- Virtualization makes web server farm management easy
- Virtualization has several challenges
  - Isolation between virtual machines
  - Support for various OSs
  - Small performance overhead

Xen: Approach & Overview

- Targeting x86 architecture
  - Virtualization was never part of a plan
  - Some privileged instructions fails silently when executed without privilege
- VMWare ESX Server
  - Dynamically rewrite non-trapping privileged instructions
- Xen
  - Modify OS codes to achieve performance

Full-virtualization vs. Para-virtualization



- ✓ **Guest OS** – One of OSes that VMM can host
- ✓ **Domain** – a running virtual machine within which a guest OS executes

Principles

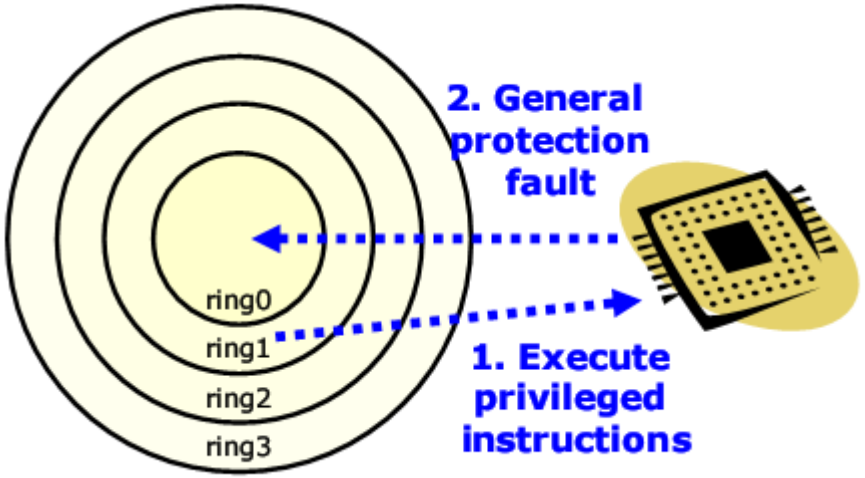
- Support applications without modification
- Support full multi-application operating system
- Paravirtualization is necessary to obtain high performance and strong resource isolation
- Completely hiding the effects of resource virtualization from guest OSes risk both correctness and performance

Virtual Machine Interface

- Memory management
  - Segmentation, Paging
- CPU
  - Protection, Exceptions, System Calls, Interrupts and Time
- Device I/O
  - Network, Disk, etc.

CPU

- x86 ring has 4 privilege levels
  - Ring 0: Xen VMM
  - Ring 1: Guest OS
  - Ring 3: User Software



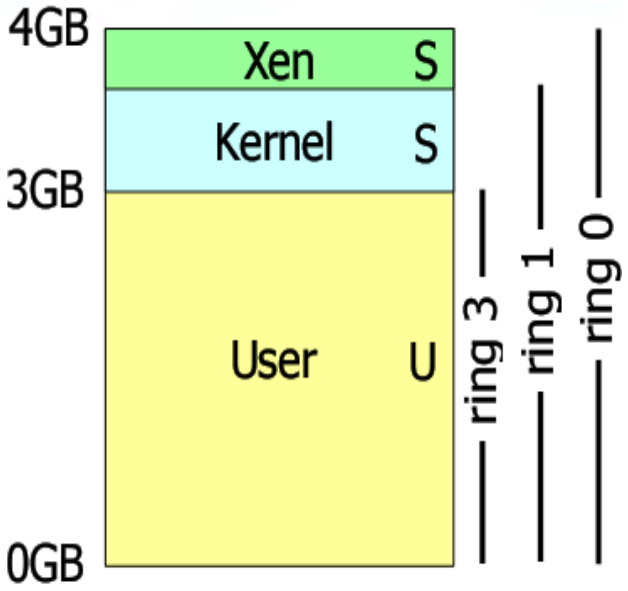
- Privileged instructions are required to be validated and executed within Xen

CPU(cont'd)

- Exceptions, page faults, software traps, are virtualized straightforwardly
  - Sole modification to page fault handler(due to CR2 register)
- Support "Fast" exception handler which can be directly called from ring 1

Memory Management

- Guest OSes are responsible for allocating and managing the hardware page tables
- Xen exists in a 64MB section at the top of every address space
  - Avoiding a TLB flush when entering and leaving



Device I/O

- Xen exposes a set of clean and simple device abstractions
- I/O data is transferred to and from each domain(guest OS) via Xen
  - Using shared memory: asynchronous buffer-descriptor rings
- Xen supports a lightweight event-delivery mechanism
  - Asynchronous notification to a domain

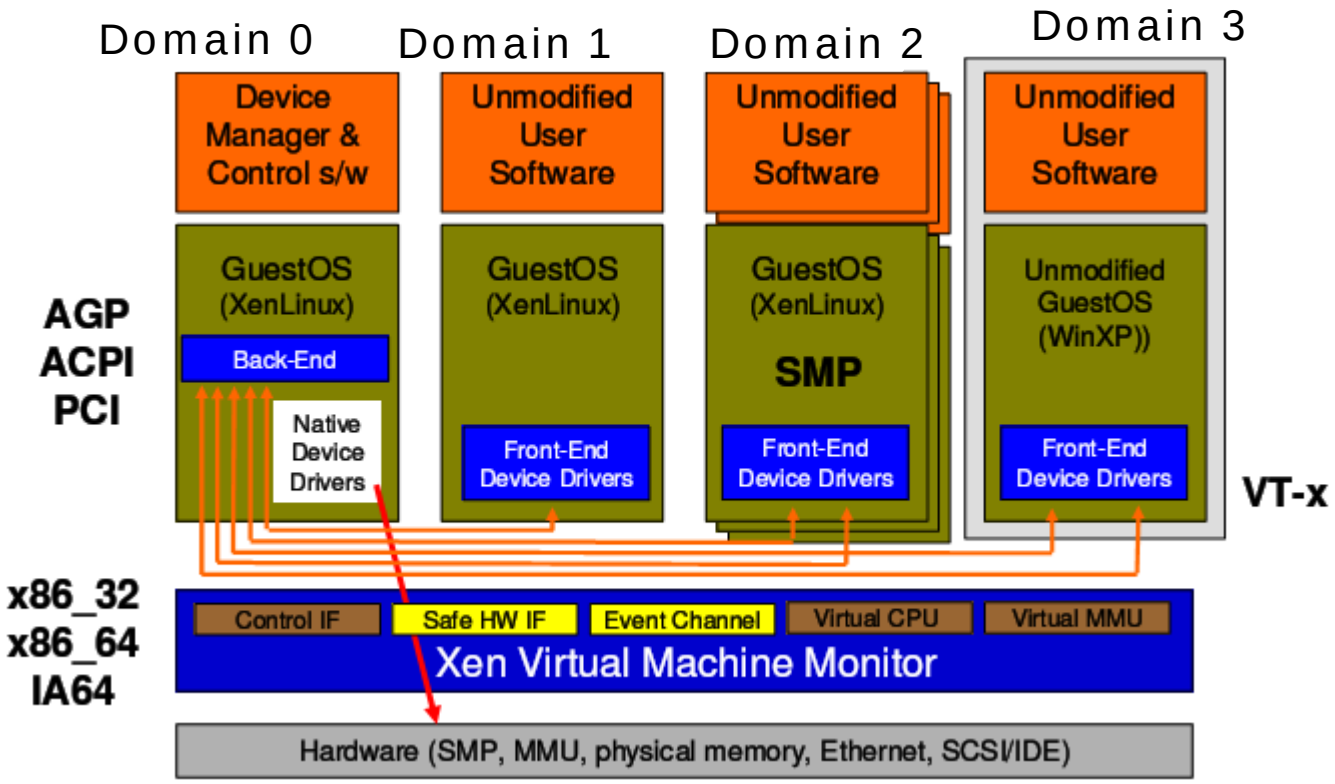
Control and Management

- Provide simple interface so that user can change policy such as CPU scheduling

- Domain 0
  - Permitted to use the control interface
  - Created at boot time
  - Responsible for hosting the application-level management software
    - Create and terminate other domains
    - Control scheduling parameters
    - Physical memory allocations
    - Access to physical disks and network devices

Xen Architecture

Domain 0



Detailed Design

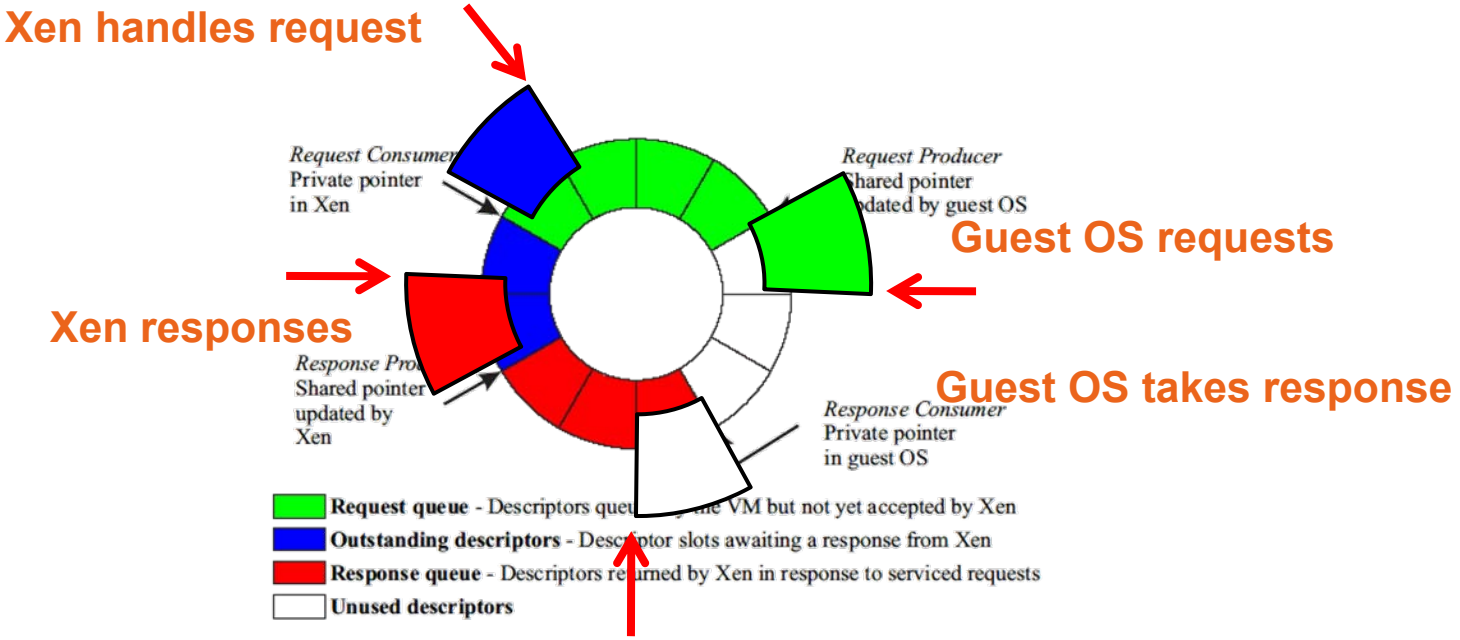
- Control transfer
  - To communicate between VMM and Guest OS
- Data transfer
  - To send I/O data between VMM and Guest OS
- Subsystem Virtualization
  - CPU, Memory, Network and Disk

Control Transfer: Hypercalls and Events

- Hypercalls
  - From domain to Xen
  - Synchronous calls
  - Similar to ‘System call’ between process and kernel
  - E.g. Page table updates

- Events
  - From Xen to domain
  - Asynchronous notifications
  - Replaces device interrupt delivery

Data Transfer: I/O Rings



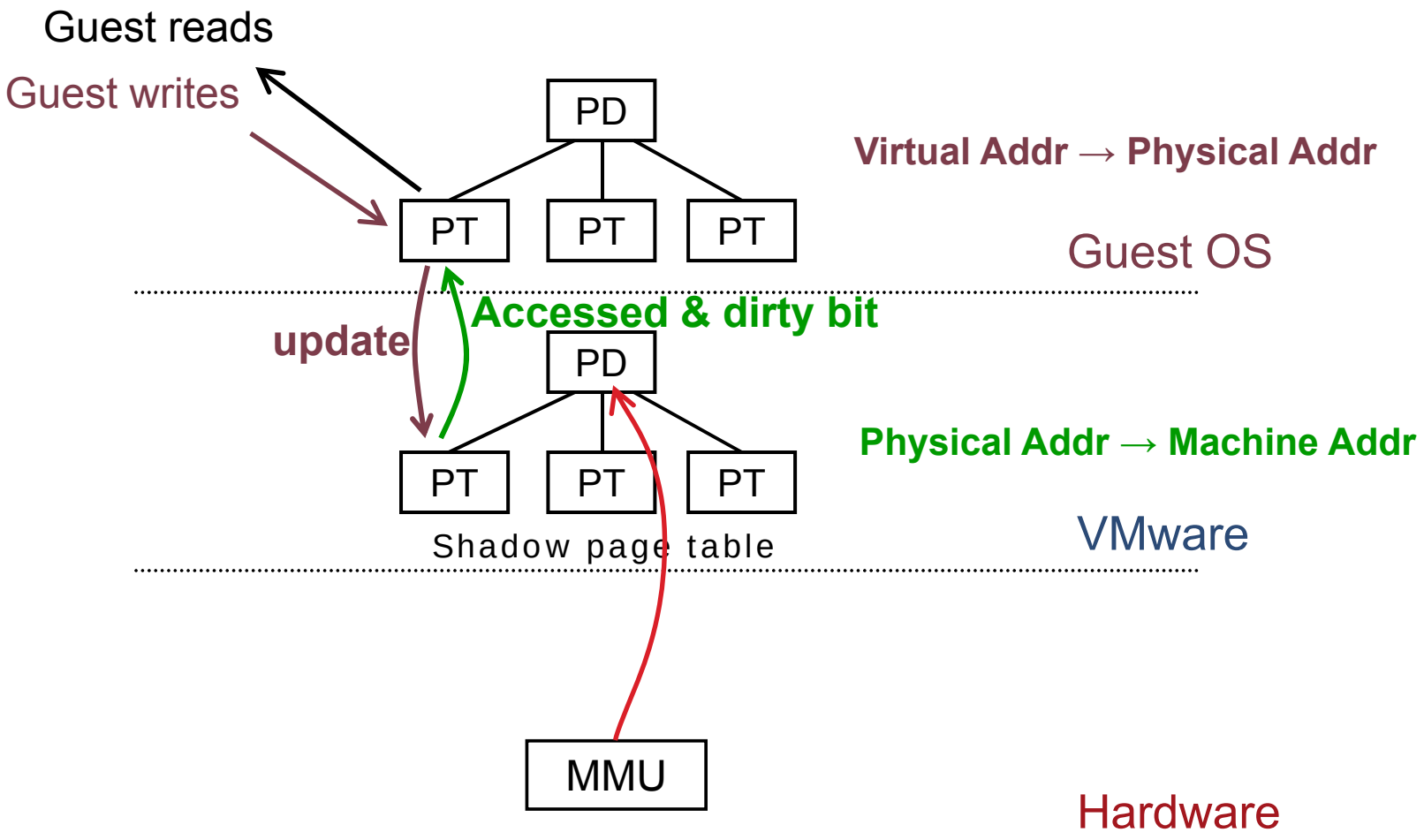
CPU Virtulization - Timer

- Real time
  - Nanoseconds passed since machine boot
  - TCP/IP timeout
- Virtual time
  - A domain's virtual time advances white it is executing
  - Used by guest OS scheduler
- Wall-clock time
  - Current real time + Offset
  - Time zone

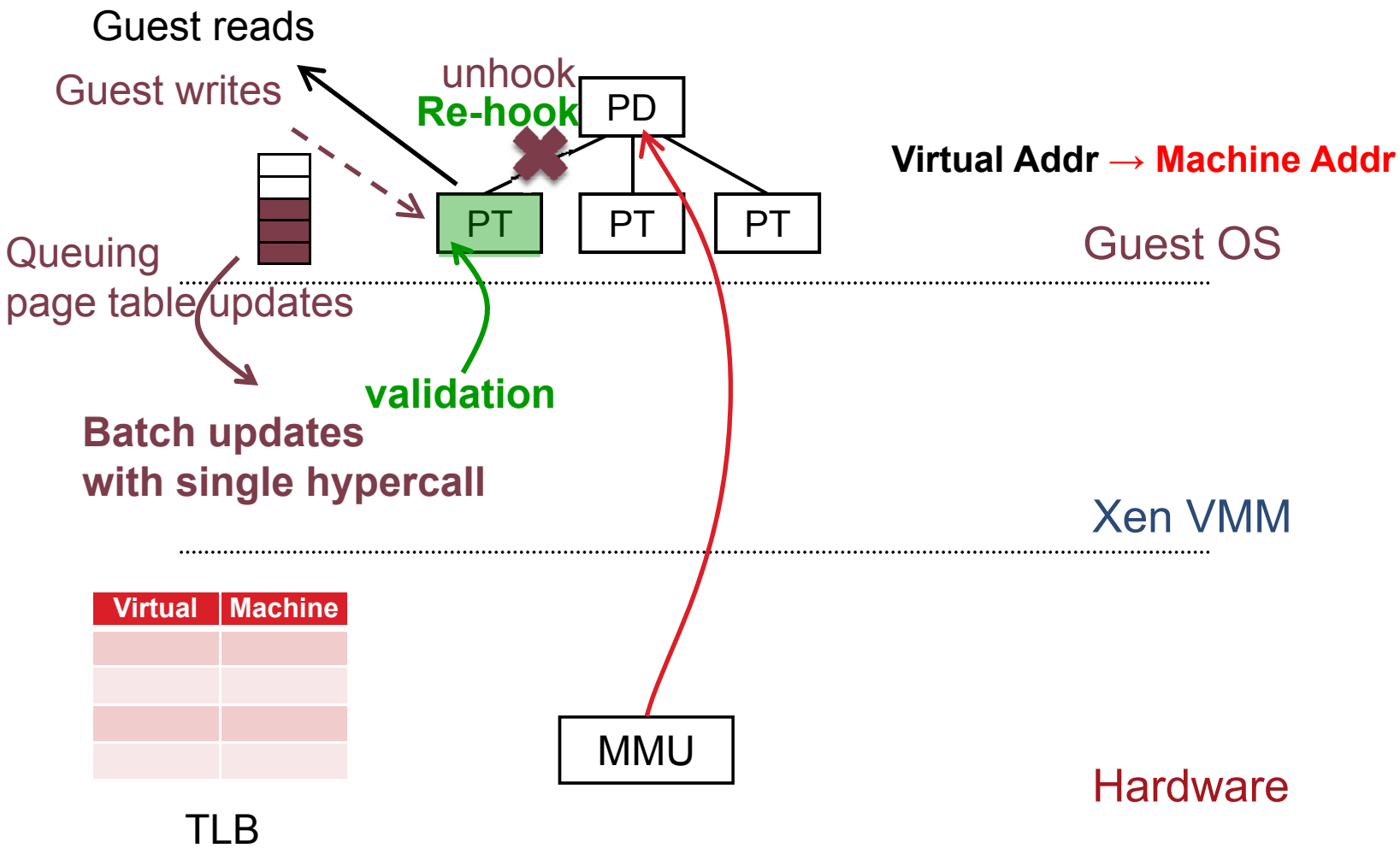
Virtual address translation

- VMware
  - Shadow page table
    - To translate physical → machine address
- Xen
  - Guest OS uses machine address directly
  - Guest OS updates own HW page table
  - Update to a page table must be validated by xen
  - Batched updates are possible

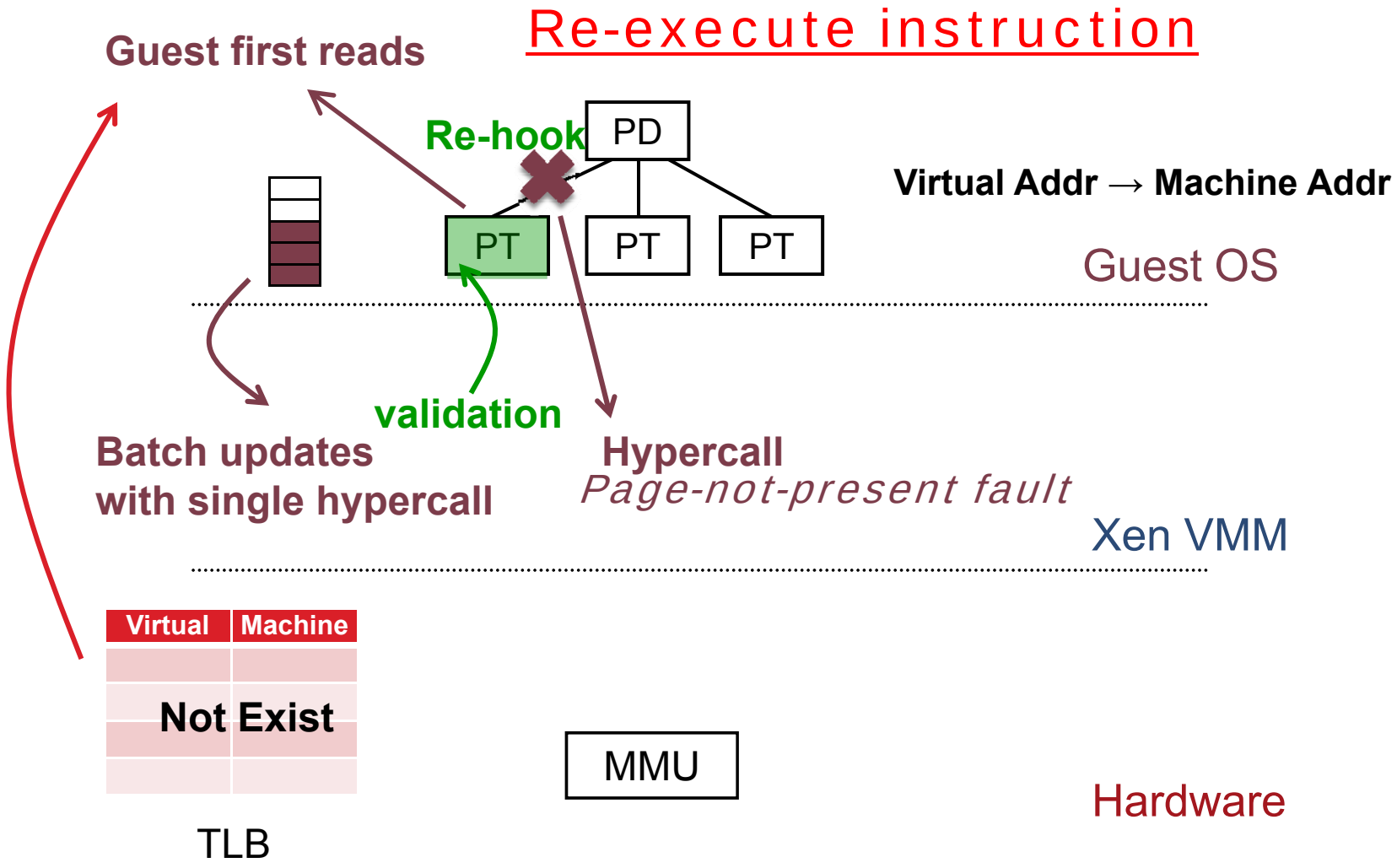
MMU Virtualization – Shadow Mode



MMU Virtualization – Direct Mode



MMU Virtualization – Direct Mode



Physical memory

- Memory allocation for each domain is specified at the time of its creation
- Statically partitioned between domains
  - Strong isolation
  - No sharing
- Balloon driver
  - Add or reduce a domain's memory
- OS may use an additional table to give the illusion of physical memory

Network / Disk

- VIFs/VFR
  - Looks like a modern network interface card
  - Two I/O rings of buffer descriptor for receiving/sending
  - Zero-copy
- Virtual Block Devices (VBDs)
  - Domain0 has direct access to physical disks
  - Other domains access through VBDs

Evaluation

The cost of porting an OS to Xen

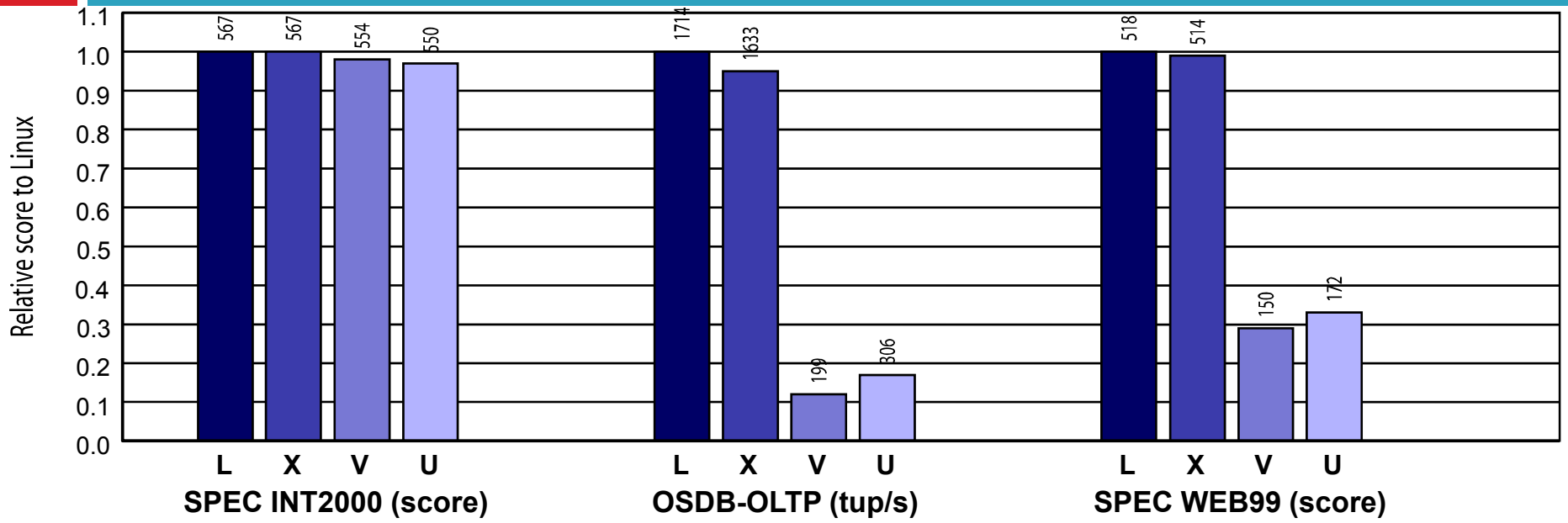
OS Subsection	Linux	XP
Architecture-independent	78	1299
Virtual Network Driver	484	-
Virtual Block Device Driver	1070	-
Xen-specific (non-driver)	1363	3321
Total (Portion of total x86 code base)	2995 (1.36 %)	4620 (0.04%)

단위: Line

Test Environments

- Test Machines
  - Dell 2650 dual processor 2.4GHz Xeon Server
  - 2GB RAM
  - Linux 2.4.21
- Compared VMMs
  - Native Linux
  - XenoLinux
  - VMWare workstation 3.2
  - User-mode Linux

Relative Performance



SPEC INT2000

- CPU-intensive
- CPU time spent in user-space

Open Source Database Benchmark

- PostgreSQL native API over Unix domain socket
- Synchronous disk operations, protection domain transitions

SPEC WEB99

- Evaluating web servers
- Dynamic content generation
- HTTP POST
- CGI script

Linux (L), Xen (X), VMware Workstation (V), and UserModeLinux (U)

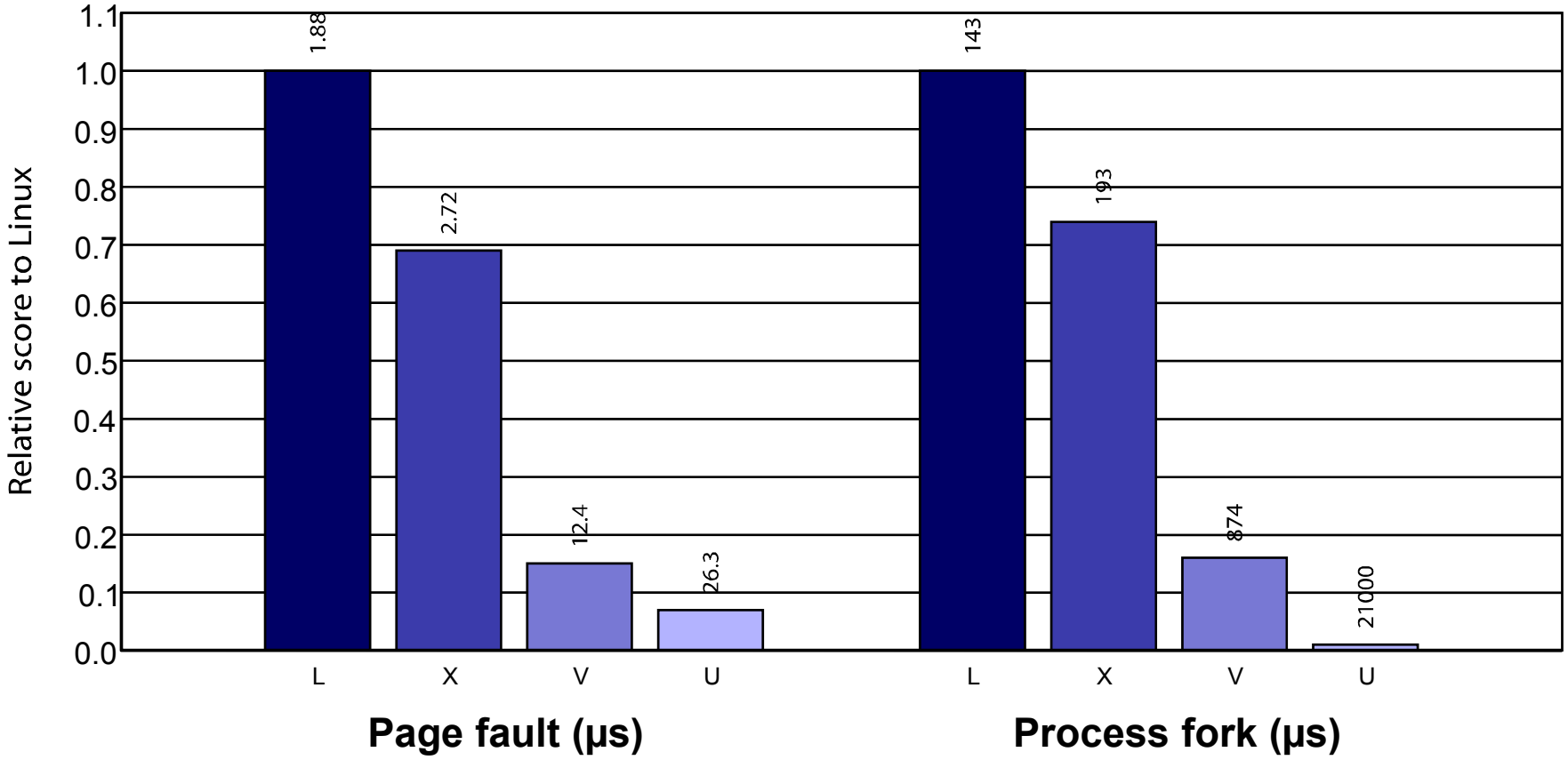
Operating System Microbenchmarks

Config	2p 0K	2p 16K	2p 64K	8p 16K	8p 64K	16p 16K	16p 64K
L-SMP	1.69	1.88	2.03	2.36	26.8	4.79	38.4
L-UP	0.77	0.91	1.06	1.03	24.3	3.61	37.6
Xen	**1.97**	**2.22**	**2.67**	**3.07**	**28.7**	**7.08**	39.4
VMW	18.1	17.6	21.3	22.4	51.6	41.7	72.2
UML	15.5	14.6	14.4	16.3	36.8	23.6	52.0

Table 4: `lmbench`: Context switching times in μs

Hypercall to change the page table base

Operating System Microbenchmarks

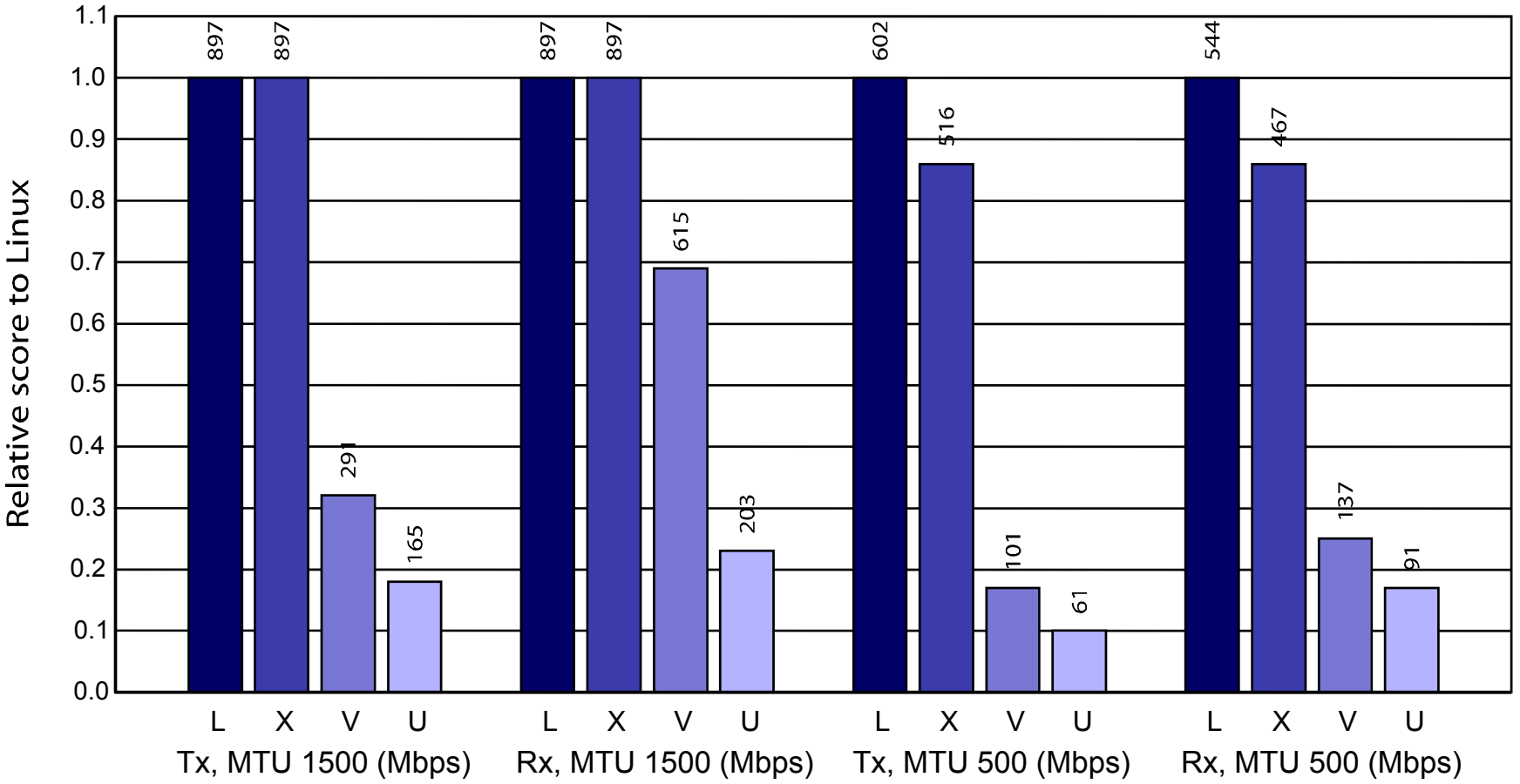


2 transitions into Xen per page

Many page table updates

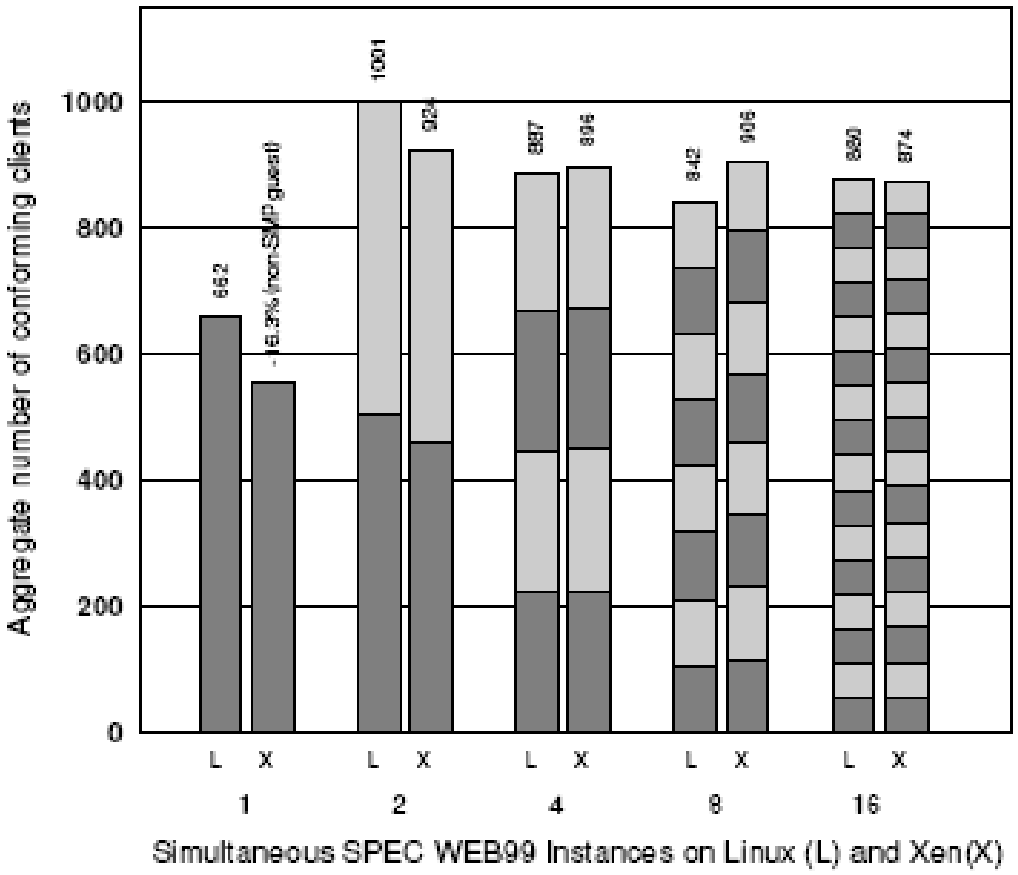
Imbench results on Linux (L), Xen (X), VMWare Workstation (V), and UML (U)

Network performance



TCP bandwidth on Linux (L), Xen (X), VMWare Workstation (V), and UML (U)

Concurrent Virtual Machines



- 2 CPU
- 1, 2, 4, 8 and 16 copies
- SPEC WEB99
  - Evaluating web servers
  - Throughput
  - Bounded latency
- Interrupt load balancer

Scalability

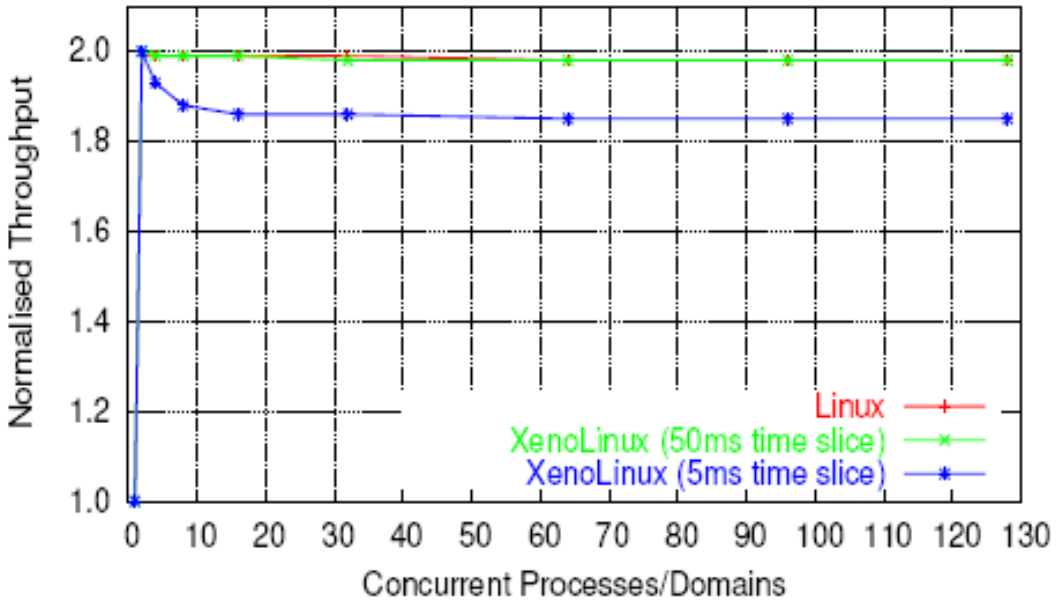


Figure 6: Normalized aggregate performance of a subset of SPEC CINT2000 running concurrently on 1-128 domains

- 2 CPU
- 1 ~ 128 domains or processes
- SPEC CINT2000
  - CPU intensive
- Context switching between large numbers of domains
- Native Linux:
  - No loss of performance
  - Identified as CPU bound
  - Long time slices of 50ms or more

Discussion and Conclusions

- Fast virtualization solution
  - Xen is practically equivalent to the performance of the native Linux
- But Guest OS must be modified
- Xen execution
- http://www.youtube.com/watch?v=m7XO3Gf9yGE

PLANETLAB

http://www.planet-lab.org

- PlanetLab is..
  - A global-scale overlay network.
  - Just a set of Linux PCs geographically distributed.

- How is it different from typical Linux PCs?
  - If you have an account on one machine, you can access all of them.

History (I)

- March 2002:
  - Peterson and Culler organize an "underground" meeting on planetary-scale network services; David Tennenhouse (Intel Research) agrees to seed-fund the project with 100 machines.
- June 2002:
  - Chun, Roscoe, Fraser Mike Wawrzoniak bring up first PlanetLab nodes at Intel. The initial system (dubbed Version 0.5) leverages the Ganglia monitoring service and the RootStock installation mechanism from the Millennium cluster project.
- October 2002:
  - Initial deployment of 100 nodes at 42 sites is complete. Operational support (including www.planet-lab.org) moves to Intel's Distributed Systems Lab (DSL) in Oregon, under the direction of Mic Bowman.
  - V1.0 with vserver-based virtual machines and safe raw sockets
  - Blueprint paper presented at HotNets workshop

History (II)

- December 2002:
  - Meet after OSDI to report on early experiences
- February 2003:
  - PlanetLab nodes come on-line at 3 PoPs of Internet2's Abilene backbone.
  - All 11 Abilene PoPs host PL by end of 2003.
- June 2003:
  - HP joins PlanetLab
- September 2003:
  - NSF announces a $4.5M award
- September 2003:
  - PlanetLab passes the 200 node mark.

History (III)

- October 2003:
  - KAIST joins PlanetLab
  - SOSP meet
- December 2003:
  - PlanetLab passes the 300 node mark.
- January 2004:
  - PlanetLab Consortium created
  - V2.0 with support for dynamic slices, is deployed.
- February 2004:
  - First PlanetLab nodes come up on CANARIE.
- April 2004:
  - NSDI meet

History (IV)

- May 2004:
  - European PlanetLab meeting hosted by Cambridge University (UK)
- August 2004:
  - First PlanetLab nodes come up on the Rede Nacional de Ensino e Pesquisa (RNP), the Brazilian National Education and Research Network. RNP nodes are donated by HP and Intel.
- September 2004:
  - PlanetLab is featured in Intel CTO Pat Gelsinger's keynote address at the Intel Developers Forum (IDF).
  - Asian PlanetLab meeting is hosted by KAIST in Seoul, South Korea.
- December 2004:
  - CERNET, the Chinese Education and Research Network, joins PlanetLab. CERNET nodes are donated by Intel and HP.

History (V)

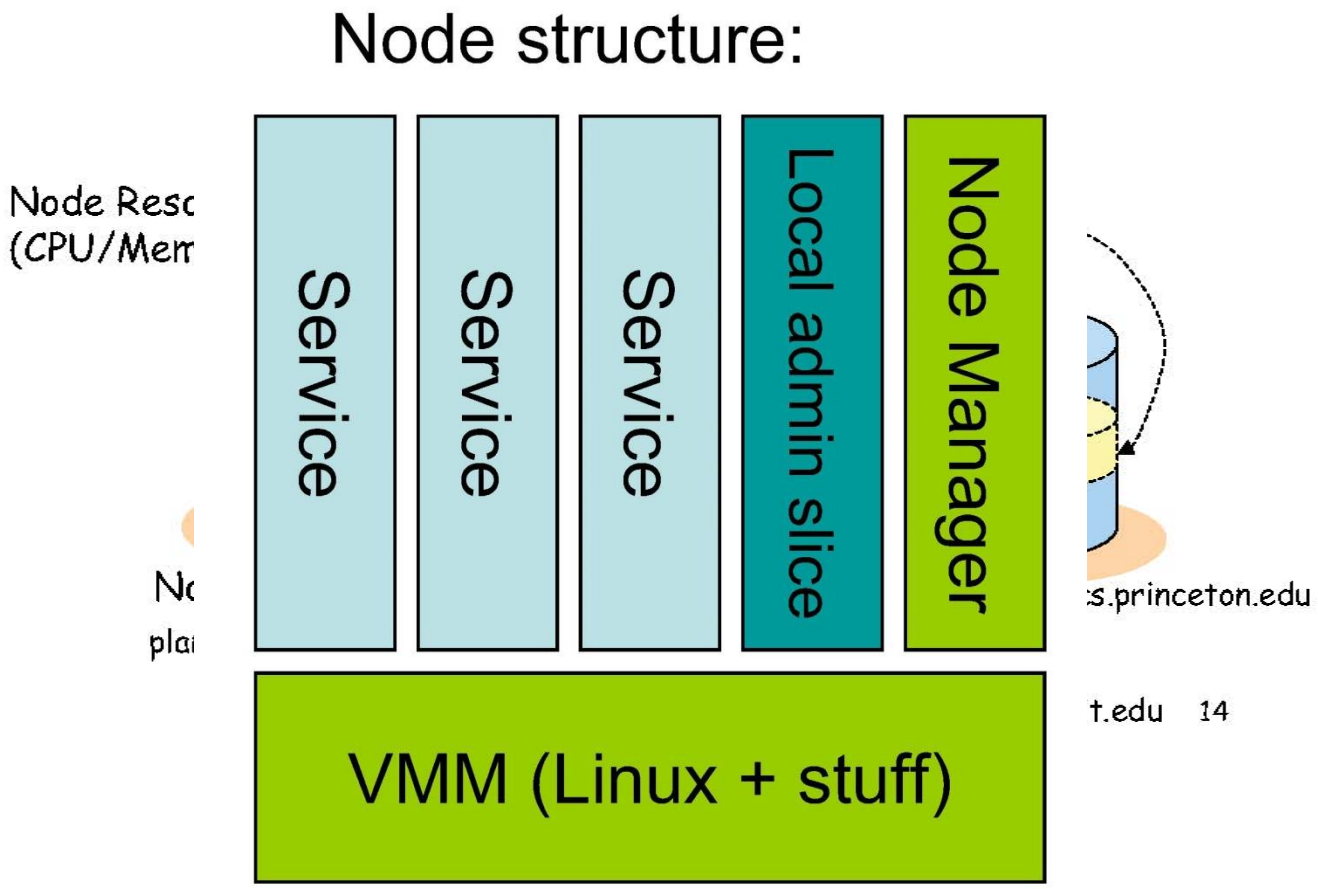
- October 2005:
  - EPFL in Lausanne Switzerland hosts a "PlanetLab Everywhere" meeting. The attendees discuss "private PlanetLabs" and how to federate them.
- May 2006:
  - HP Labs hosts a PlanetLab meeting focused on federation.
  - Mark Huang (Princeton) demonstrates MyPLC, a packaging of the PlanetLab software for the purpose of instantiating private PlanetLabs.
- April 2007:
  - V4.0 based on MyPLC deployed
- June 2007:
  - PlanetLab passes the 800 node mark.
- July 2007:
  - PL federates with the OneLab “PlanetLab – Europe”

Design Principles

- **Slice-ability**
  - “slices” as fundamental resource unit
  - Distributed set of (virtual machine) resources
  - Each service runs in a slice of PlanetLab node

- Unbundled management
  - Partition management into orthogonal services
- Application-centric interfaces
  - Evolve from what people actually use

Nodes, Slices, and Slivers



tesy of Aki Nakao

Node Virtualization Options

- Full virtualization (VMware, etc.)
  - Inadequate scaling for PlanetLab
  - Long-term: investigate h/w support
- Paravirtualization (Xen, Denali)
  - Not yet mature, but multiple OSes
  - Very attractive medium-term
- Syscall-level virtualization (Vservers)
  - Works today, fairly stable
  - Only one operating system (Linux)

What is PlanetLab Good for?

- Planetary-Scale applications:
  - Low latency to widely spread users
  - Span boundaries: jurisdictional and administrative
  - Simultaneous viewpoints: on the network or sensors
  - Hardware deployment is undesirable
  - Long-running services, not only experiments / exercises

PlanetLab is not ...

- A distributed supercomputer
- A simulation platform
- An Internet emulator
- An arena for repeatable experiments
- Completely representative of the current Internet

PlanetLab is ....

- An opportunity to qualitatively validate distributed systems in a real deployment environment
- An opportunity to gain valuable experience about what works and what doesn't in the wide area at scale

What is PlanetLab Doing to Internet?

- PlanetLab appears to function as a disruptive technology
  - Applications use the network differently
  - The network sometimes reacts in a hostile manner
  - Leads to new requirements on infrastructure

Long-Term Aims

- PlanetLab incubates the next Internet
- New networks deployed as overlays over existing Internet
- Service-oriented network becomes the norm
- Computation as a localizable network resource
- Students can enjoy the opportunity to participate in the process

Lessons from PlanetLab

- Nothing works as expected at scale!
  - Many unintended and unexpected consequences of algorithmic choices
  - Simulation results do not carry over well
  - Simulate, deploy, measure, edit cycle
- Evaluating competing approaches “in the wild” refines techniques
- The ability to try things out “for real” really stimulates ideas

Research using PlanetLab

- Network measurement
- Wide-area distributed storage
- Distributed query processing
- Overlay Networks
- Testbed Federation
- ...

Example #1: OpenHash

- Brad Karp, Sylvia Ratnasamy, Sean Rhea
- Sharable, stable DHT service
  - “Turn on, put in, get out”
  - Accessible from any machine
  - “Redir” allows consistent hashing over
  - arbitrary node sets
- Implemented over Bamboo
  - Extremely robust & churn-resilient DHT implementation

Example #2: PIER

- Joe Hellerstein, Ryan Huebsch, Boon Thau Long, Timothy Roscoe, Scott Shenker, Ion Stoica, etc.
- Pure P2P Relational Query Engine
  - Join, Selection, Projection, Aggregation
- DHT (Bamboo) used for:
  - Joins (rehashing / rendezvous)
  - Aggregation (building agg. trees)
- Used for PlanetLab monitoring & resource discovery

Joining PlanetLab

- PlanetLab Consortium
  - Princeton U. / U. of Washington / Intel Research
- Apply for PlanetLab membership
  - Intellectual Property belongs to the member, not the PlanetLab consortium.
  - Academic institutions and non-profit organizations join with no fee
- Connect machines to PlanetLab
  - Just buy 2 PCs and provide power & network link.
  - You're in!!
- http://www.planet-lab.org/

OneLab

- European Project (2006.9 ~ 2008.8)
  - To organize PlanetLab regionally
  - UPMC NPA
  - http://one-lab.org
  - Funded by FP6-IST programme
  - 10 member organizations

ROADS

- International Workshop on Real Overlays And Distributed Systems
  - From a technical perspective, to share new ideas, experiences, and work in progress.
  - From a social point of view, to nurture important face-to-face interaction among researchers working on Real Overlays And Distributed Systems builders.
  - To bring together researcher from the United States who are building novel new ROADS with those in outside of the USA
- June 2007 in Brazil; July in Warsaw

Real in ROADS

- Systems
  - that are designed to run on a real platform for a period of time.
  - Research projects, teaching exercises, or more permanent services, but they should address technical issues of actual overlays and distributed systems.

What Will We Build?

- Federated PlanetLab
  - A starter for
    - Hands-on experience of managing PlanetLab nodes
- Design & run simple overlay services
  - Who:
  - What:

Acknowledgements

[1] Timothy Roscoe's presentation at 18th APAN

[2] Aki Nakao's presentation at 21st APAN

Validating Internet Research

- Routing toolkits
  - Implement a subset of IP routing functionality
  - Rarely used in production environments
- Open source-based testbed networks
  - Provide valuable tools for the researcher
  - Rarely provide a realistic test environment