ClickOS and the Art of Network Function Virtualization

(NSDI 2014 Paper)

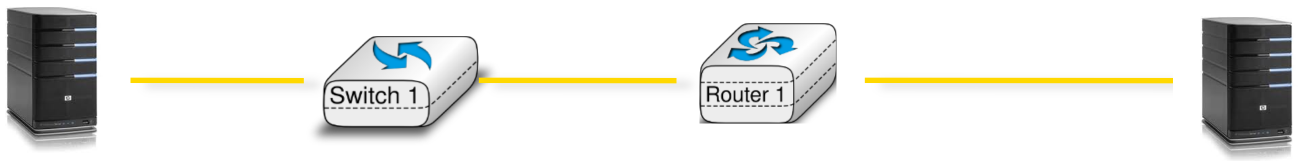
Joao Martins*, Mohamed Ahmed*, Costin Raiciu§, Roberto Bifulco*,
Vladimir Olteanu§, Michio Honda*, **Felipe Huici***

* NEC Labs Europe, Heidelberg, Germany

§ University Politehnica of Bucharest

firstname.lastname@neclab.eu, firstname.lastname@cs.pub.ro

The Idealized Network



Application			Application
Transport			Transport
Network		Network	Network
Datalink	Datalink	Datalink	Datalink
Physical	Physical	Physical	Physical

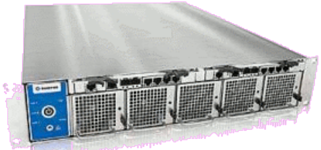
A Middlebox World

ad insertion

WAN accelerator

BRAS

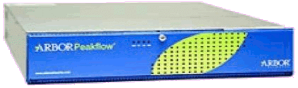
carrier-grade NAT


transcoder

session border controller

IDS

load balancer



DDoS protection

firewall

QoE monitor

DPI

Hardware Middleboxes - Drawbacks

- **Expensive equipment/power costs**

- **Difficult to add new features (vendor lock-in)**

- **Difficult to manage**

- **Cannot be scaled on demand (peak planning)**

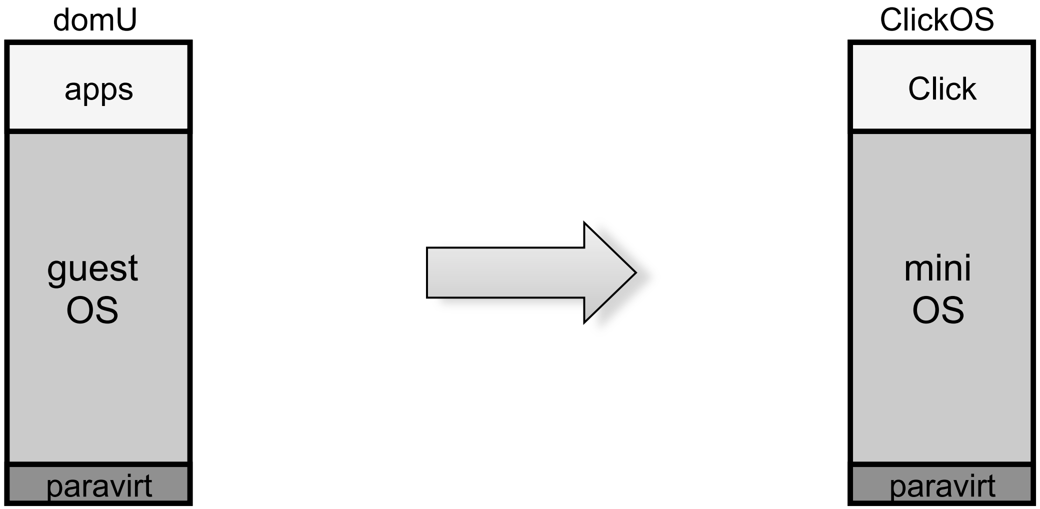
Shifting Middlebox Processing to Software

- **Can share the same hardware across multiple users/tenants**
- **Reduced equipment/power costs through consolidation**
- **Safe to try new features on a operational network/platform**
- But can it be built using commodity hardware while still achieving high performance?
- ClickOS: **tiny Xen-based virtual machine that runs Click**

From Thought to Reality - Requirements

	ClickOS
Fast Instantiation	✓ 30 msec boot times
Small footprint	✓ 5MB when running
Isolation	✓ provided by Xen
Performance	✓ 10Gb/s line rate* 45 µsec delay
Flexibility	✓ provided by Click

What's ClickOS ?

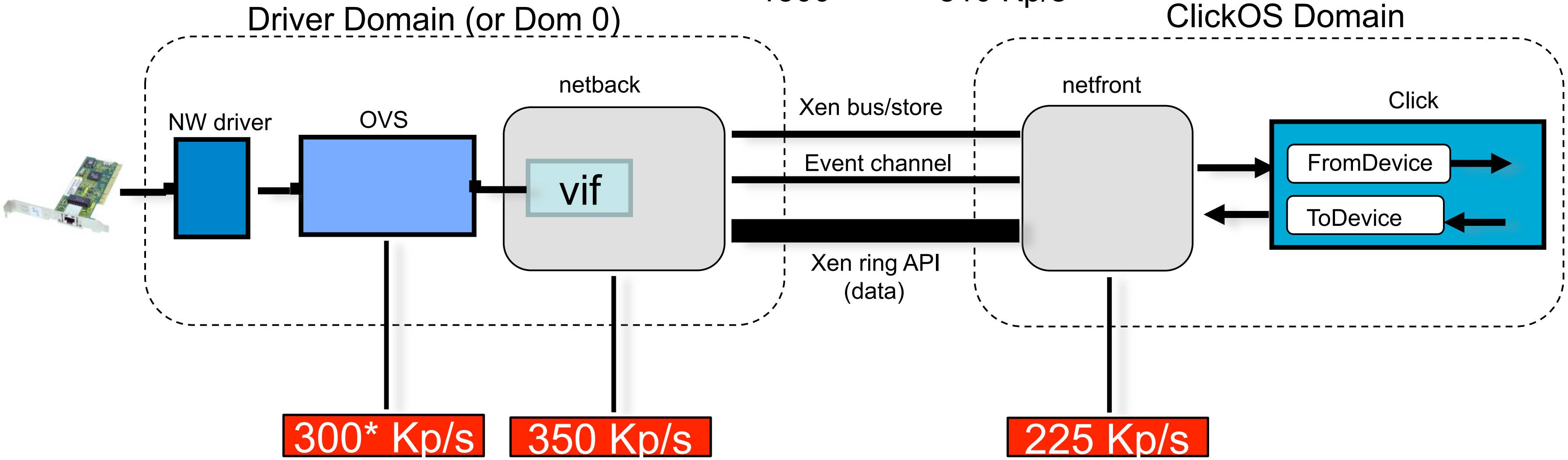


Work consisted of:

- **Build system to create ClickOS images (5 MB in size)**
- **Emulating a Click control plane over MiniOS/Xen**
- **Reducing boot times (roughly 30 milliseconds)**
- **Optimizations to the data plane (10 Gb/s for almost all pkt sizes)**
- **Implementation of a wide range of middleboxes**

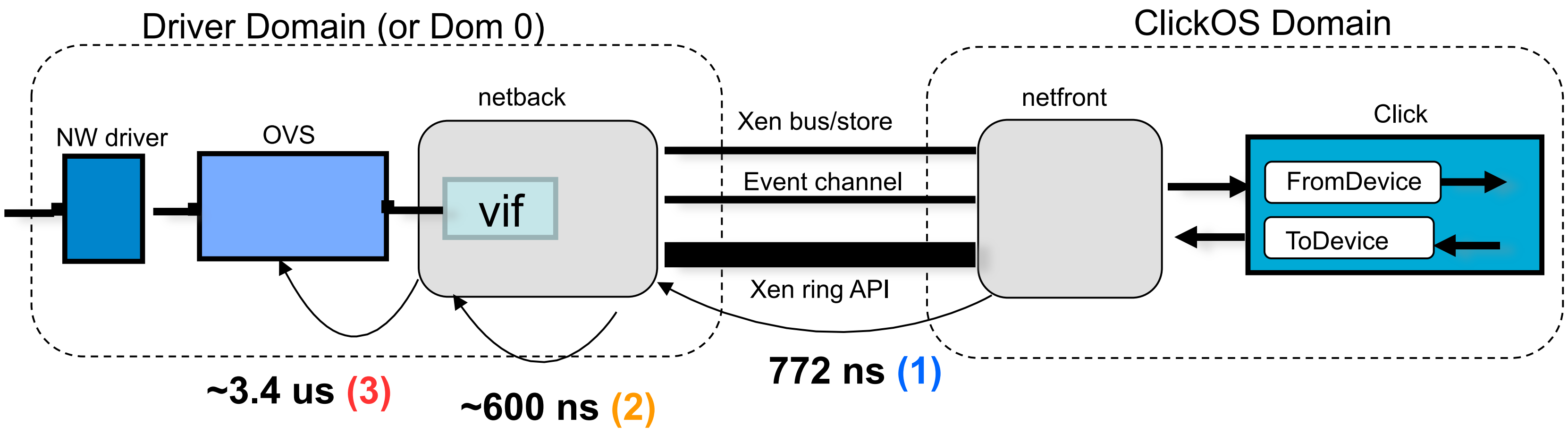
Performance analysis

packet size (bytes)	10 Gbit/s rate
64	14.88 Mp/s
128	8.4 Mp/s
256	4.5 Mp/s
512	2.3 Mp/s
1024	1.2 Mp/s
1500	810 Kp/s



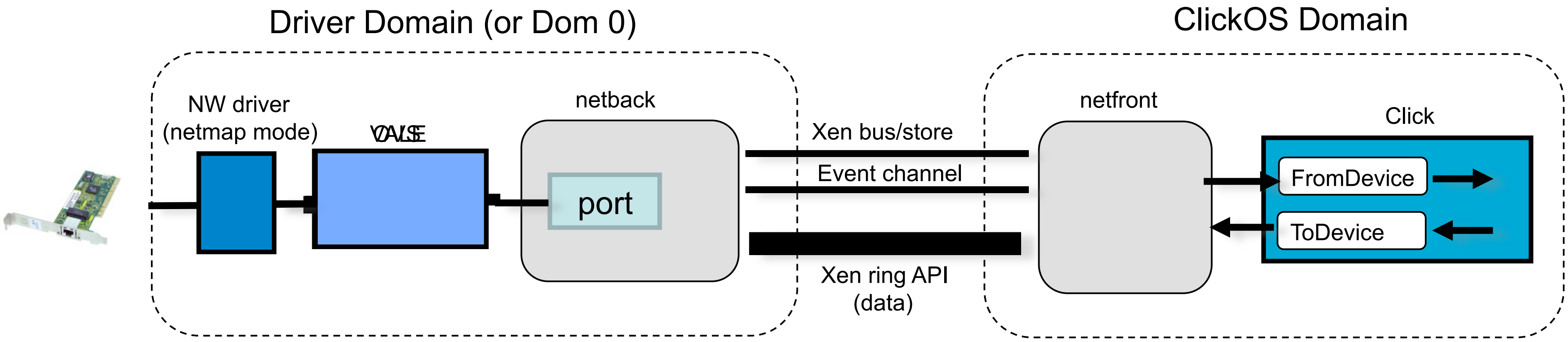
* - maximum-sized packets

Performance analysis

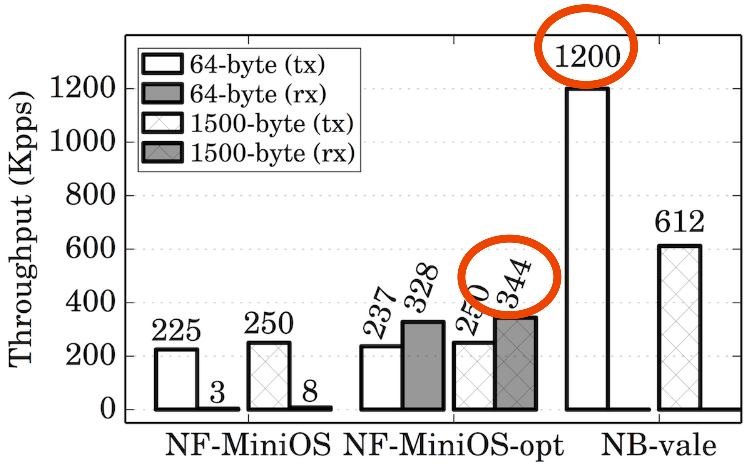


- **Copying packets between guests greatly affects packet I/O (1)**
- **Packet metadata allocations (2)**
- **Backend switch is slow (3)**
- **MiniOS netfront not as good as Linux**

Optimizing Network I/O – Backend Switch

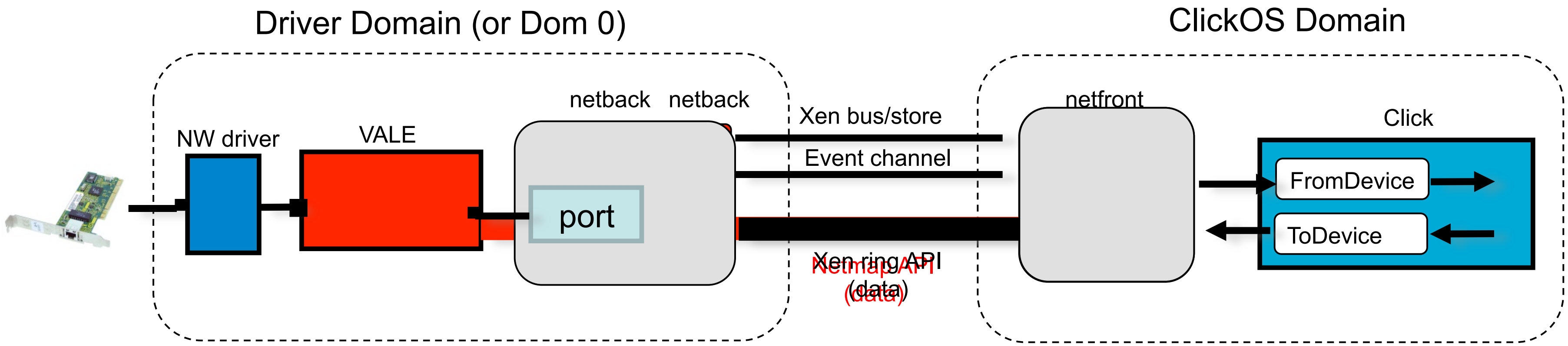


- Reuse Xen page permissions (frontend)
- Introduce VALE[1] as the backend switch
- Increase I/O requests batch size



[1] VALE, a switched ethernet for virtual machines, ACM CoNEXT'2012
Luigi Rizzo, Giuseppe Lettieri
Universita di Pisa

Optimizing Network I/O



Minimal memory requirements

- For max. throughput a guest only needs 2 MB of memory

Breaks other (non-MiniOS) guests

- **But we have implemented Linux netfront driver**

slots	KB (per ring)	# grants (per ring)
64	135	33
128	266	65
256	528	130
512	1056	259
1024	2117	516
2048	4231	1033

ClickOS Prototype Overview

- **Click changes are minimal ~600 LoC**
- **New toolstack for fast boot times**
- **Cross compile toolchain for MiniOS-based apps**
- **`netback` changes comprise ~500 LoC**
- **`netfront` (Linux/MiniOS) around ~600 LoC**
- **VALE switch extended to:**
  - **Connect NIC ports and modular switching**

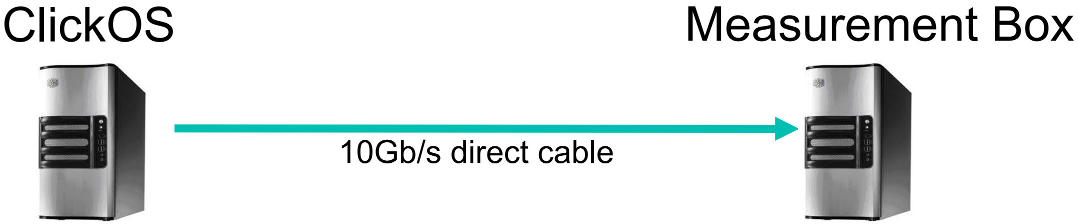
EVALUATION

Experiments

- ClickOS Instantiation
- State reading/insertion performance
- Delay compared with other systems
- Memory footprint

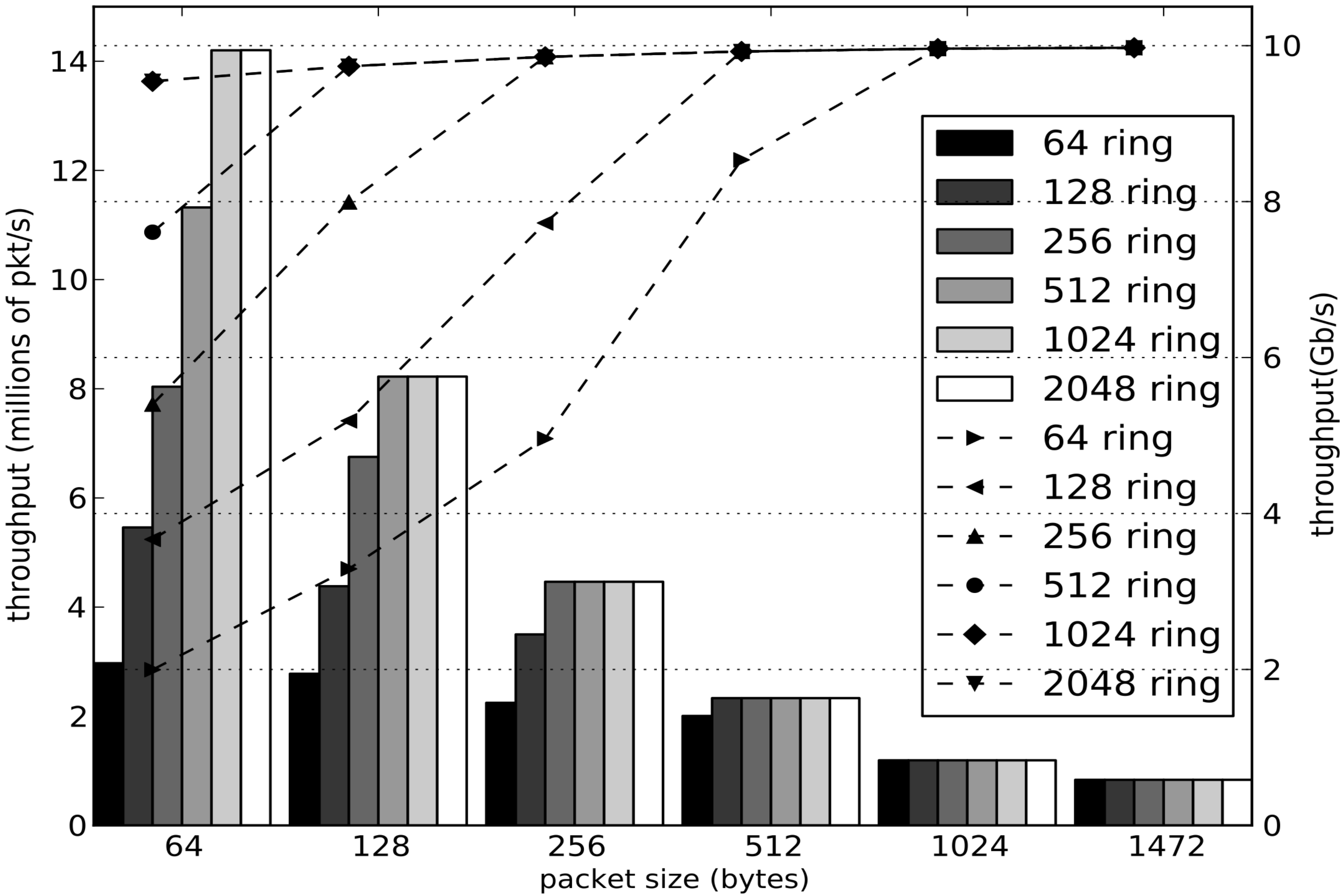
- Switch performance for 1+ NICs
- **ClickOS/MiniOS performance**
- Chaining experiments
- Scalability over multiple guests
- Scalability over multiple NICs
- **Implementation and evaluation of middleboxes**
- **Linux Performance**

ClickOS Base Performance

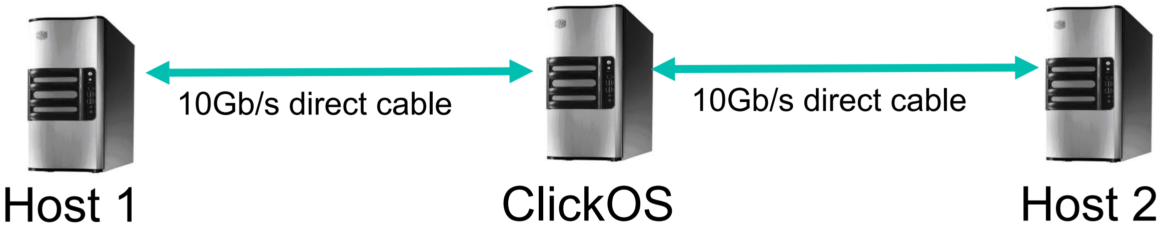


Intel Xeon E1220 4-core 3.2GHz (Sandy bridge)
16GB RAM, 1x Intel x520 10Gb/s NIC.
One CPU core assigned to VMs, the rest to the Domain-0
Linux 3.6.10

ClickOS Base TX Performance

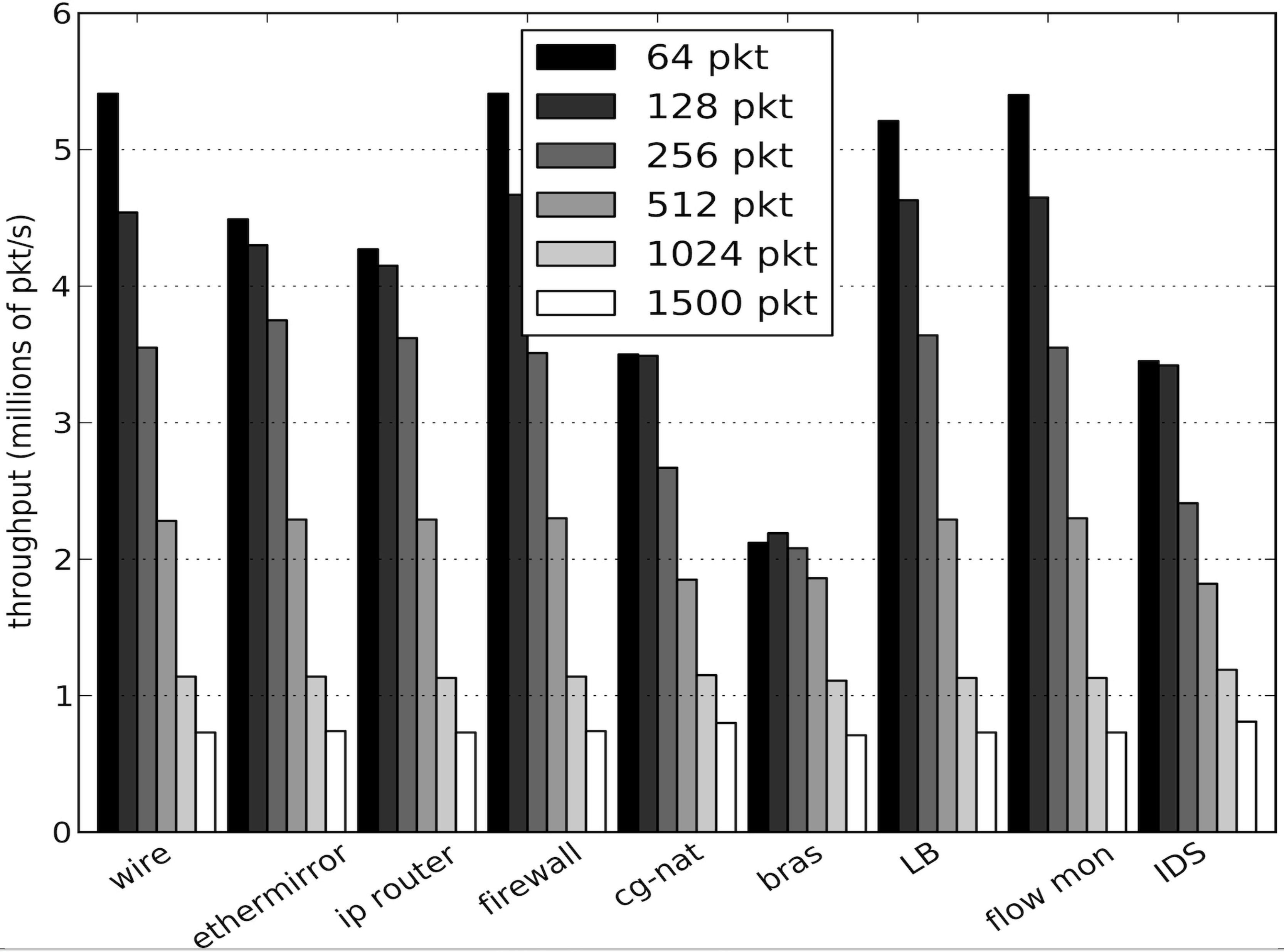


ClickOS (virtualized) Middlebox Performance

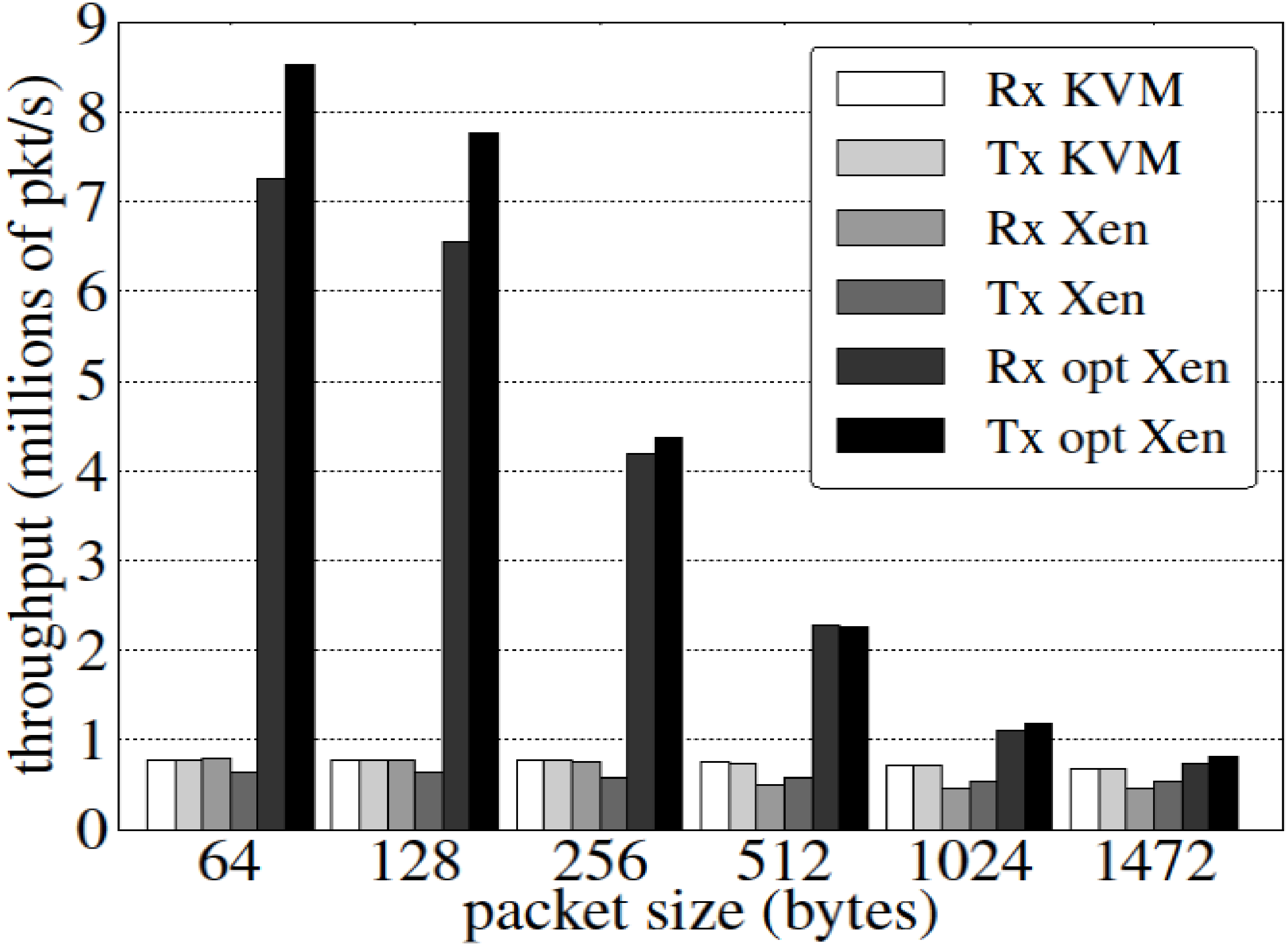


Intel Xeon E1220 4-core 3.2GHz (Sandy bridge)
16GB RAM, 2x Intel x520 10Gb/s NIC.
One CPU core assigned to Vms, 3 CPU cores Domain-0
Linux 3.6.10

ClickOS (virtualized) Middlebox Performance



Linux Guest Performance



- Note that our Linux optimizations apply only to netmap-based applications

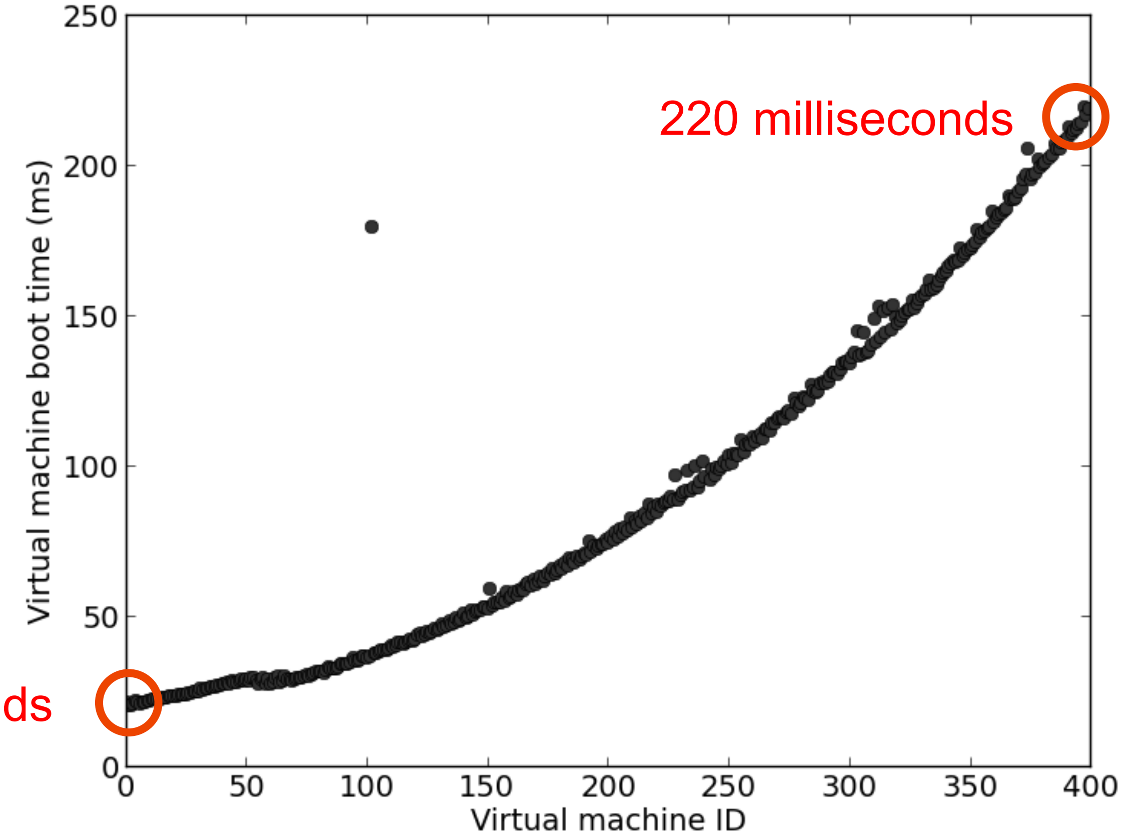
Conclusions

- Virtual machines can do flexible high speed networking

- ClickOS: **<u>Tailor-made operating system</u> for network processing**
  - Smaller is better: **Low footprint is the key to heavy consolidation**
  - Memory footprint: **5MB**
  - Boot time: **30ms**

- Future work:
  - **Massive consolidation of VMs (thousands)**
  - **Improved Inter-VM communication for service chaining**
  - **Reactive VMs (e.g., per-flow)**

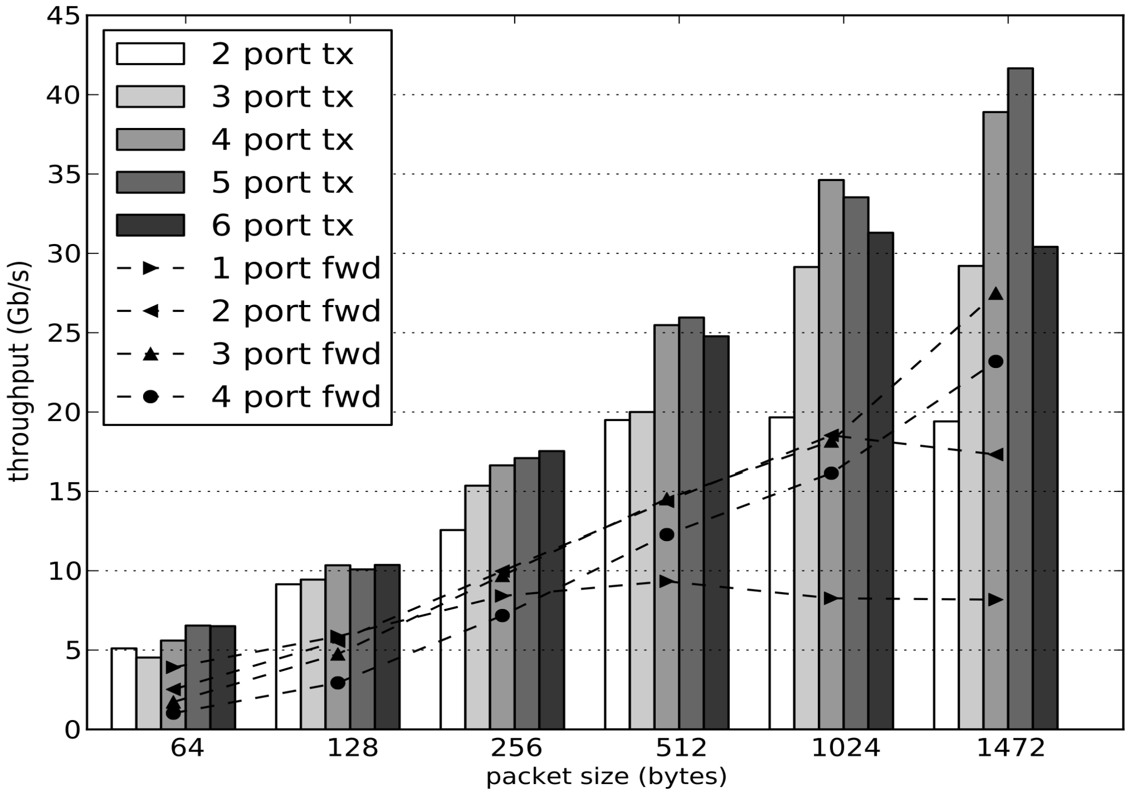
ClickOS Boot times

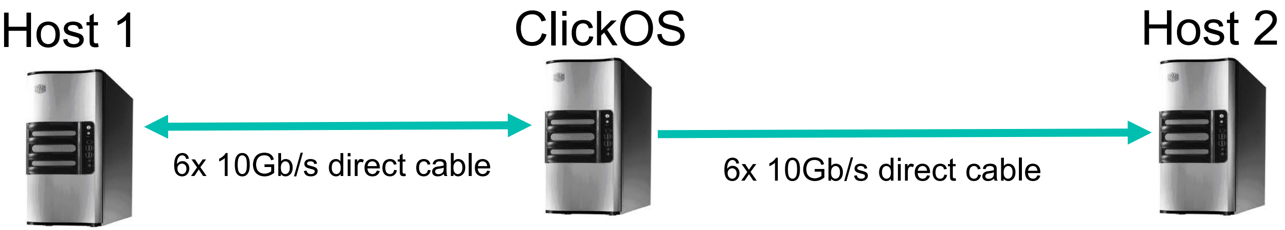


220 milliseconds

30 milliseconds

Scaling out – Multiple NICs/VMs





Host 1

ClickOS

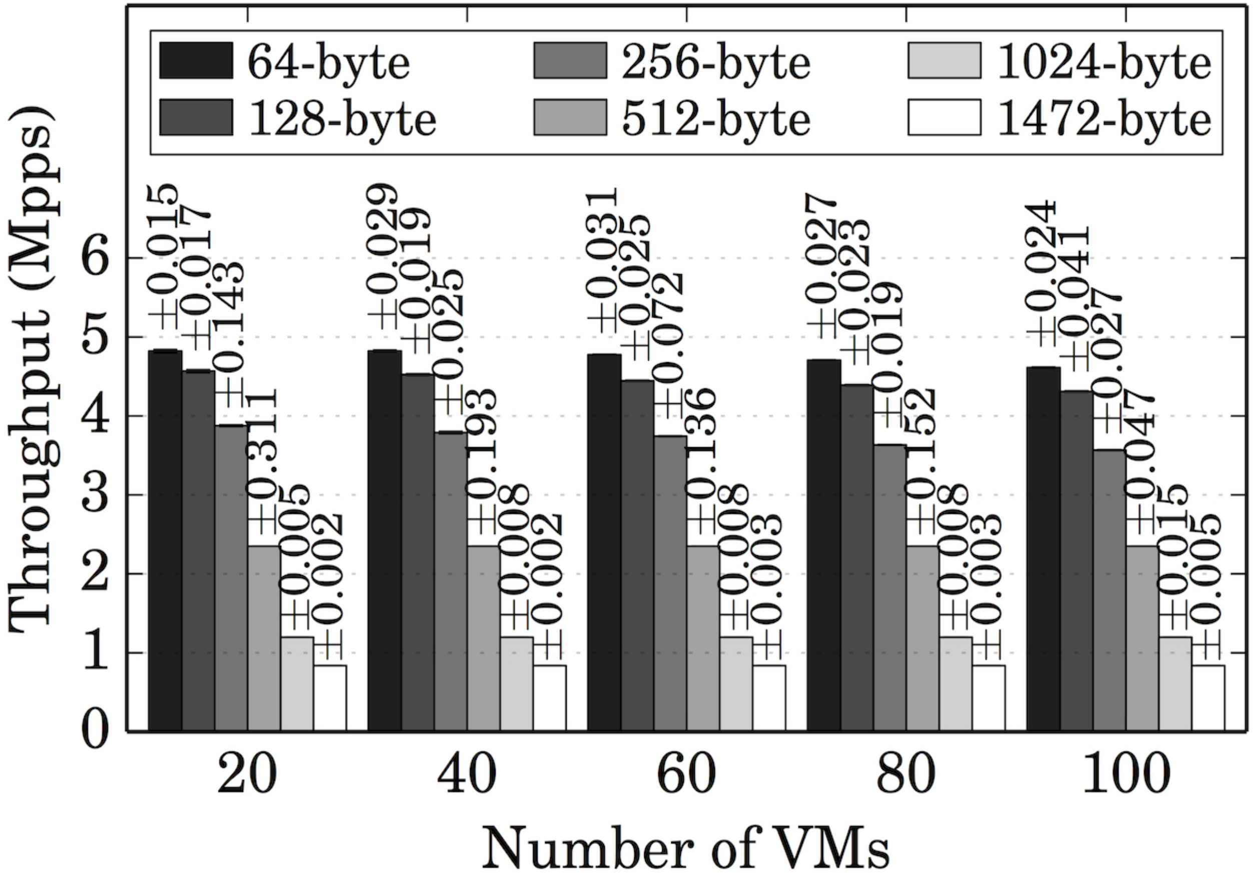
Host 2

6x 10Gb/s direct cable

6x 10Gb/s direct cable

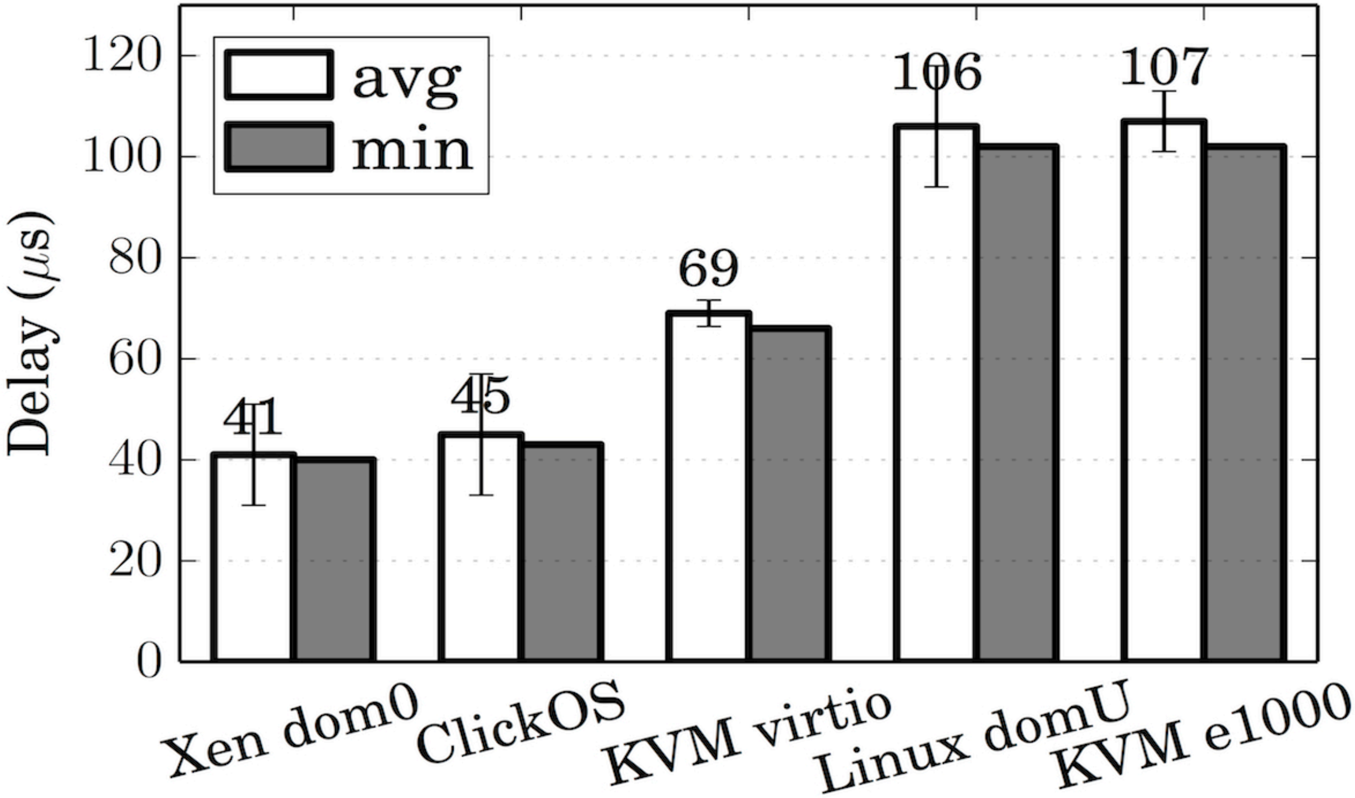
Intel Xeon E1650 6-core 3.2GHz, 16GB RAM, dual-port Intel x520 10Gb/s NIC.
3 cores assigned to VMs, 3 cores for dom0

Scaling out – 100 VMs Aggregate Throughput



Intel Xeon E1650 6-core 3.2GHz, 16GB RAM, dual-port Intel x520 10Gb/s NIC.
3 cores assigned to VMs, 3 cores for dom0

ClickOS Delay vs. Other Systems



Towards Massive Server Consolidation

Filipe Manco, João Martins, **Felipe Huici**

{filipe.manco,joao.martins,felipe.huici}@neclab.eu

NEC Europe Ltd.

Xen Developer Summit 2014

The Super Fluid Cloud

- Target: remove barriers in current cloud deployments
  - Extremely flexible infrastructure
  - **Milliseconds** instantiation and migration of resources
  - **Thousands** of concurrent units running
- This would allow new use cases
  - On the fly deployment of middleboxes
  - Flash crowds
  - Energy consumption reduction
  - Your use case here...

Recent trend: specialized guests

- ClickOS, OSv, Mirage, Erlang on Xen, etc
  - Small memory footprints
  - Relatively fast boot times
  - Provide the basic functionality to make use cases a reality

- Our work focuses on ClickOS
  - Targets network processing using the Click modular router software

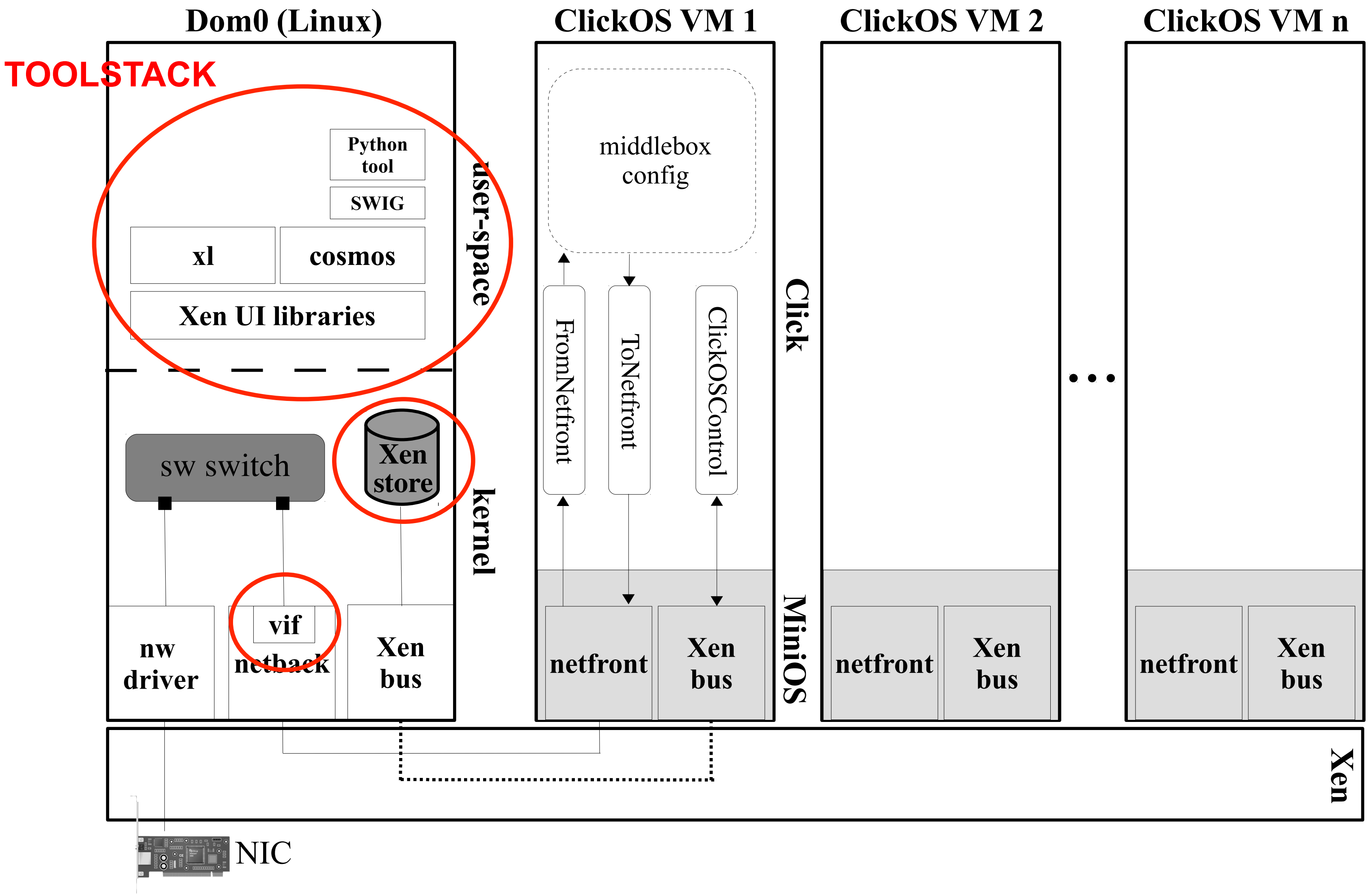
Wouldn't it be Nice if...

- Thousands of guests on a **single server**
  - Short-term target: **10K**
  - Medium-term target: **100K**
- Extremely fast domain creation, destruction and migration
  - **Tens of milliseconds**
  - Constant as number of guests increases

Experiment Setup

- Freshly installed Xen/Debian system
  - Xen 4.2
  - Linux 3.6.10
  - Debian squeeze
- Commodity server
  - 64 Cores @ 2.1GHz [4 x AMD Opteron 6376]
  - 128GB RAM DDR3 @ 1333MHz

Xen and ClickOS Architecture



Baseline Test

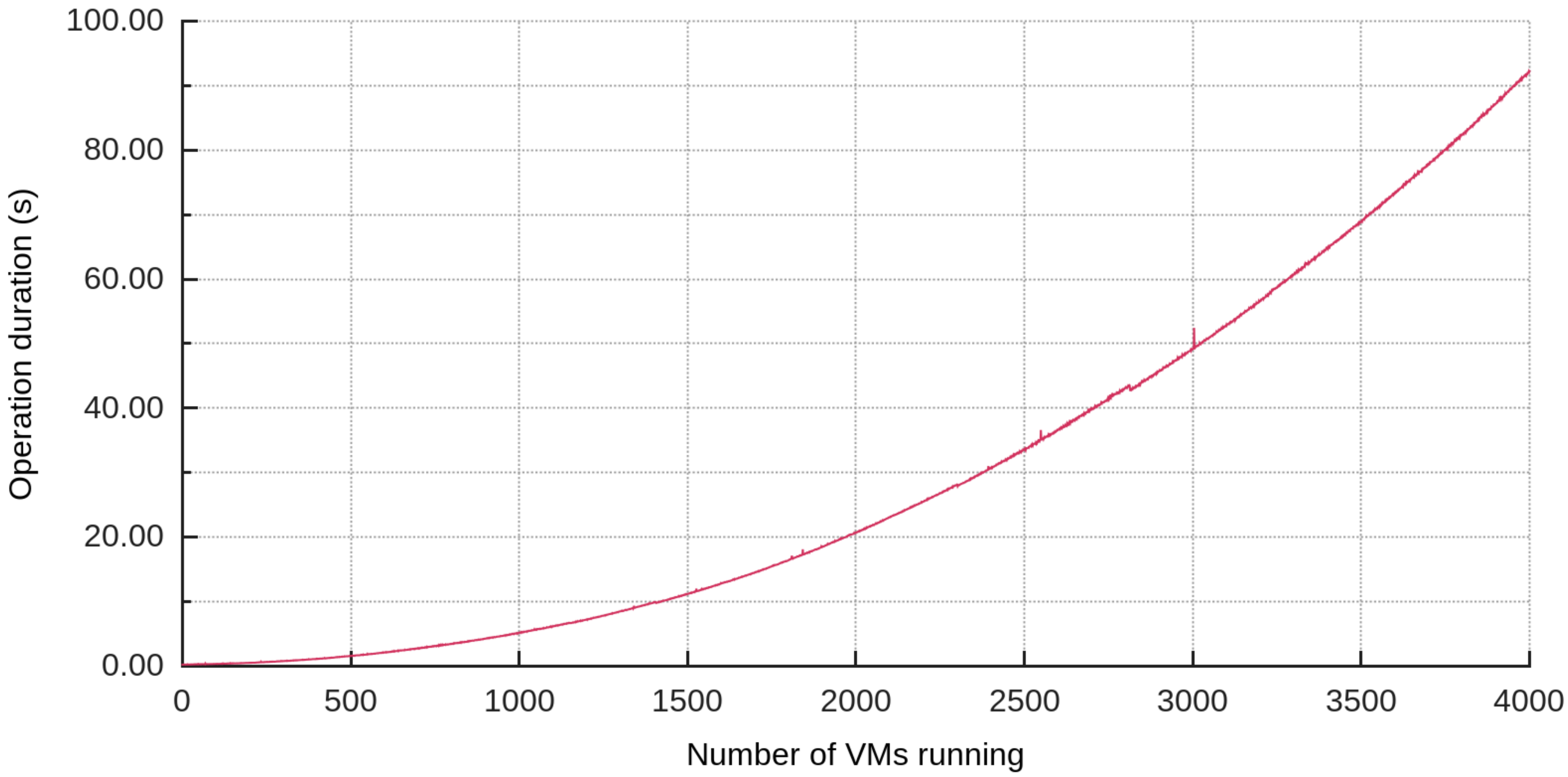
Boot as many guests as possible before system breaks

- Using ClickOS guests
  - 8 MB of RAM
  - 1 VIF
- Guests are mostly idle
  - Running arp responder configuration
  - Only *arping* guests to check they're working

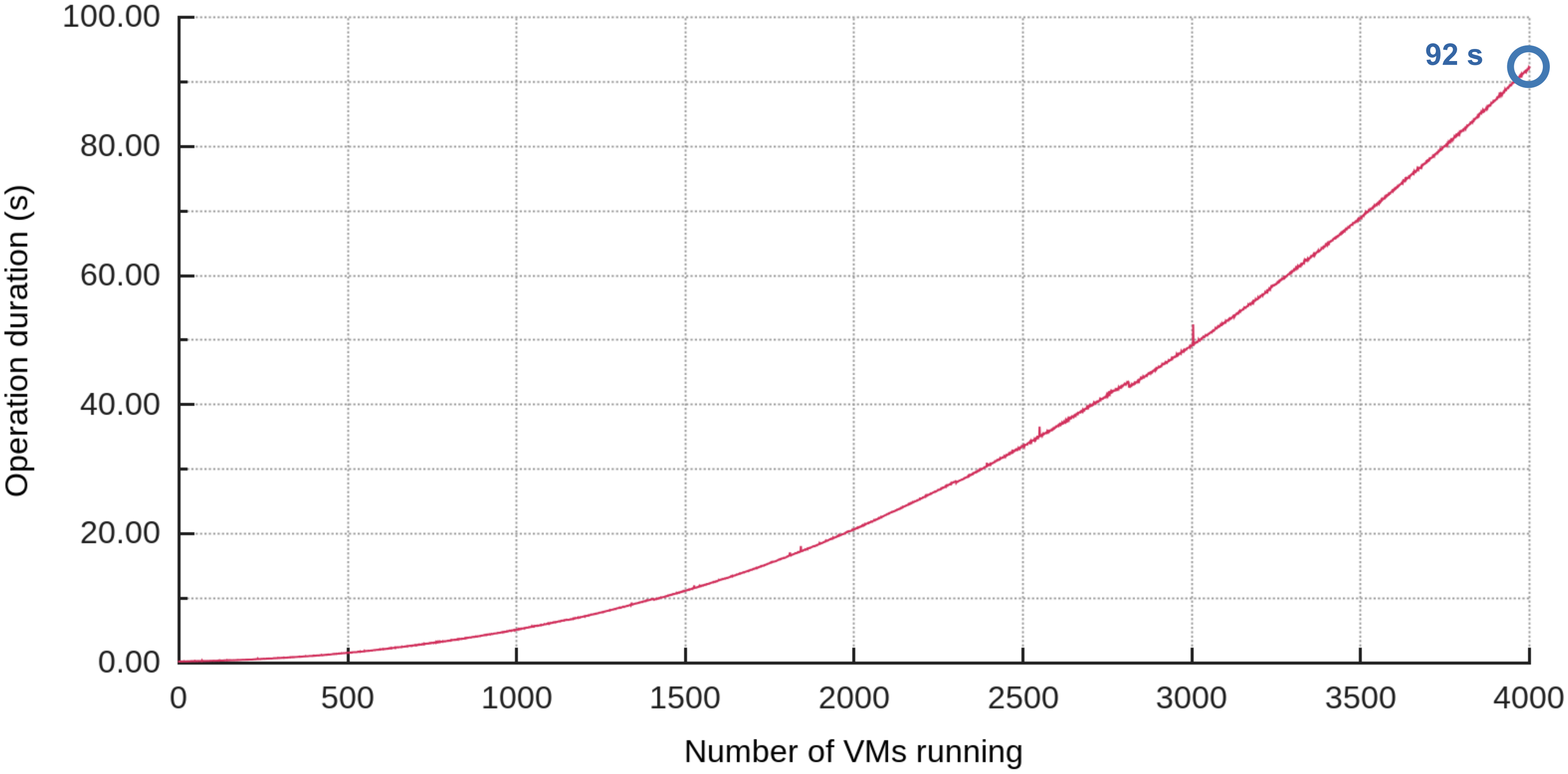
Didn't Work Quite Well...

- Stopped test after 4K guests
  - Took ~ 5 days
  - Up to ~ 100 seconds for creation of last guest (normally ClickOS boots in ~30 milliseconds)
- All the domains were running, but:
  - Only first ~300 guests fully functional
- System got extremely slow
  - Dom0 unusable

Domain Creation Time



Domain Creation Time



Two Types of Problems

- Hard limitations
  - Prevent guests from booting correctly
  - Only ~300 guests fully usable
- Performance limitations
  - Decreasing system performance
  - System unusable after just a few hundred guests

HARD LIMITATIONS

Issues

- Cannot access guests' console
  - Only first ~300 guests have accessible console
- Guests' VIF is not created
  - Only first ~1300 guests have usable VIF
- Guests cannot access the Xenstore
  - Only first ~1300 guests have access to it
- The back-end switch doesn't provide enough ports
  - Only 1024 available

Number of File Descriptors

- `xenconsoled` opens 3 FD per guest
  - `/dev/xenbus`; `/dev/ptmx`; `/dev/pts/<id>`;
- Fix
  - Linux can easily handle > 300K FD
  - Tune `fs.file-max; nofile ulimit;`

Number of PTYs

- `xenconsoled` opens 1 PTY per guest
- Fix
  - Linux can easily handle > 100K PTY
  - Tune `kernel.tty.max`
- Future
  - Only create PTY when user connects to console
  - This also reduces number of FD to 1 per guest

Number of Event Channels

- 3 Interdomain evtchn per guest
  - xenstore; console; VIF
  - 64bit Dom0: `NR_EVTCHNS == 4096`
  - Dom0 runs out after ~1300 guests
- Fix
  - Upgrade to Xen 4.4 + Linux 3.14:
    - `NR_EVTCHNS == 128K`
  - Split services into stub domains

Number of IRQs

- Linux runs out of IRQs to map evtchn
  - Limited by `NR_CPUS`
- Fix
  - `Build with: MAXSMP=y; NR_CPUS=4096`
  - `NR_IRQS == 256K`

vSwitch Ports

- Currently back-end switch supports up to few thousand ports
  - Linux bridge: 1K
  - Open vSwitch: 64K
- Workaround
  - Create multiple bridges
- Longer-term fix
  - Develop a purpose-built back-end switch

Summarizing

- Xen 4.4; Linux 3.14
- `fs.file-max; nofile ulimit`
- `kernel.tty.max`
- `MAXSMP=y; NR_CPUS=4096`

- Not yet fixed:
  - Back-end switch ports

PERFORMANCE LIMITATIONS

Issues

- Overall system becomes too slow
  - `oxenstored`
    - CPU fully utilized after a few dozen guests
  - `Xenconsoled`
    - CPU limited after ~ 2K guests
- Domain creation takes too long
  - Affects migration too

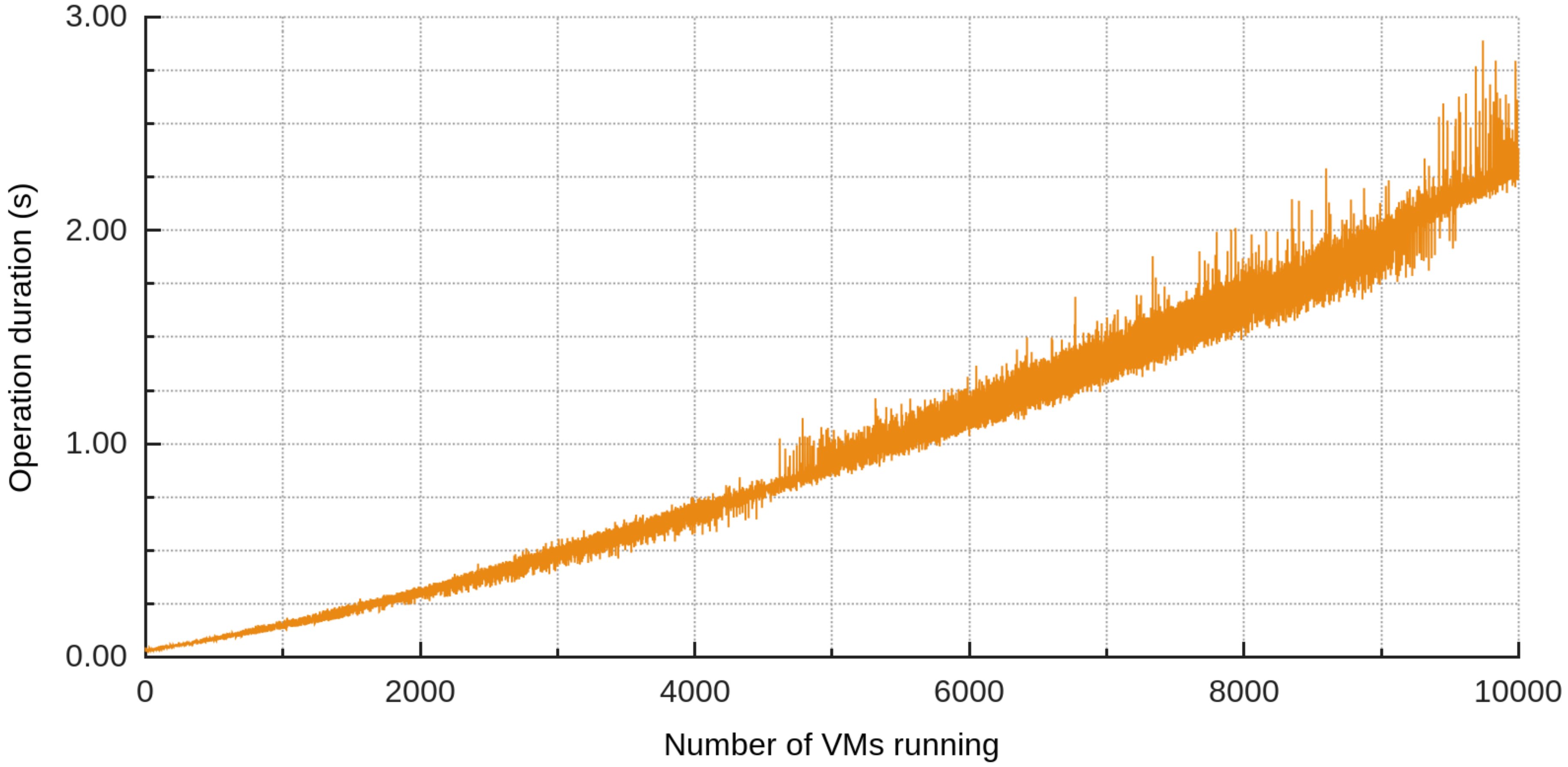
“Blind” optimizations

- 4 Core Dom0
  - 1 core for `oxenstored`
  - 1 core for `xenconsoled`
  - 2 cores for remaining processes
- Pin all vCPUs to pCPUs
- Round robin remaining 60 cores for guests
- Put everything in a ramfs

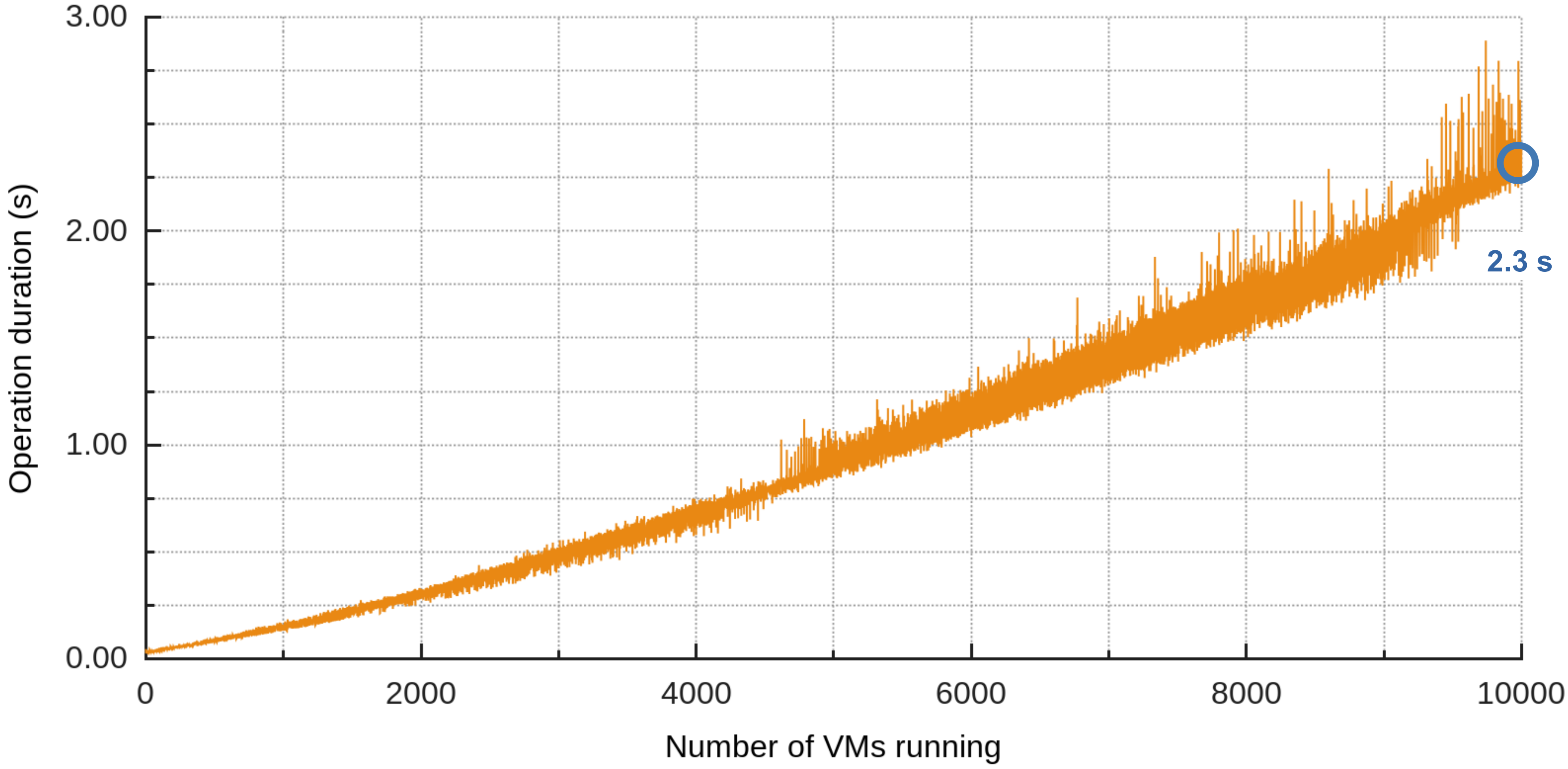
Tools' Optimizations

- xl toolstack
  - Disable xl background process (`xl create -e`)
  - Disable memory ballooning on Dom0
  - Never use domain name
    - This causes xl to retrieve all guest names from the Xenstore
  - Use specialized VIF hotplug script
  - Don't retrieve domain list on creation [PATCH]
- oxenstored
  - Use more recent version of Xenstore from:
    - https://github.com/mirage/ocaml-xenstore

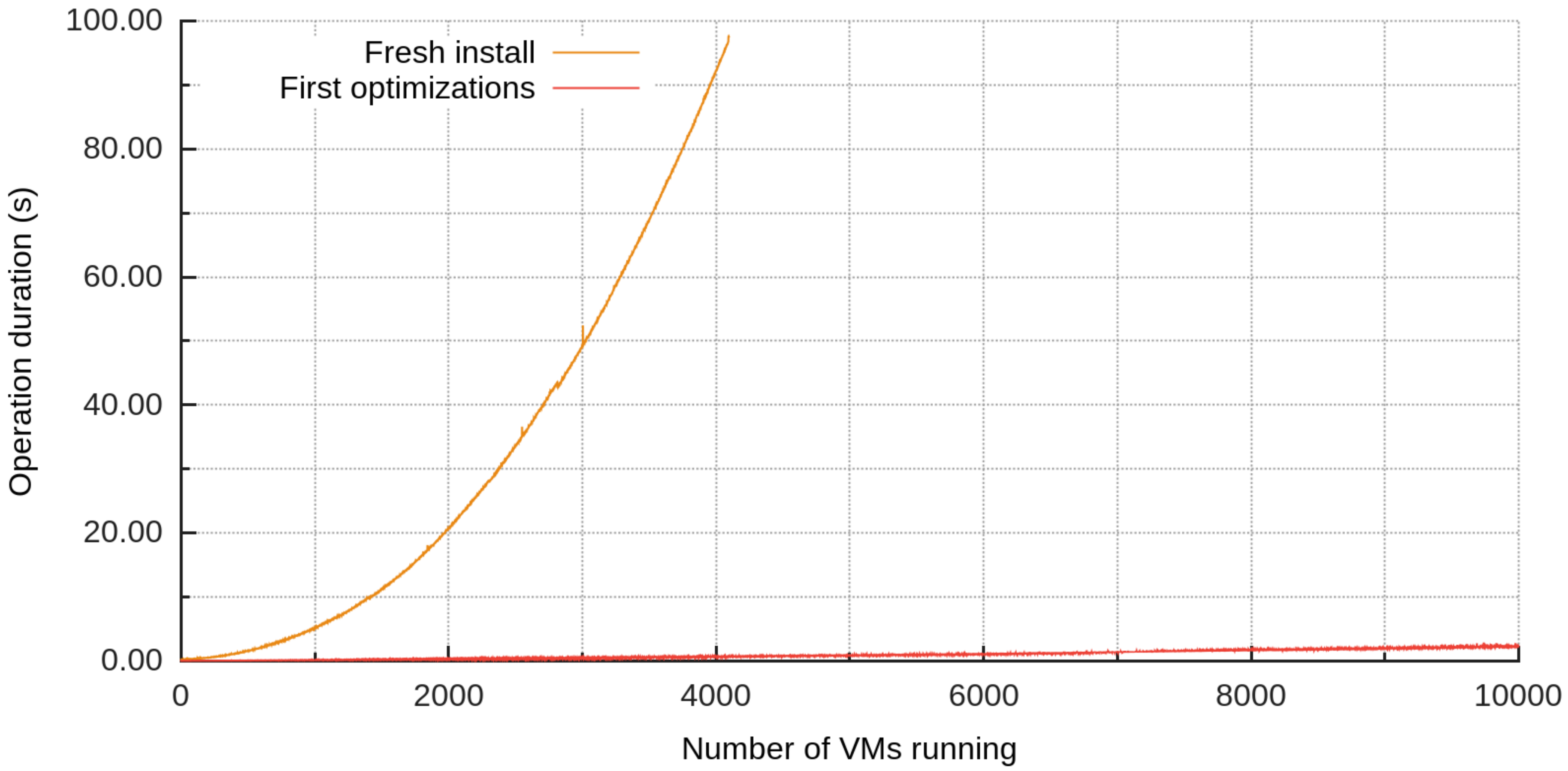
Creation Times with Optimizations



Creation Times with Optimizations



How much better is it?



With Optimizations

- Improvement: system is still usable after 10K guests
  - Although domain creation time is far from ideal

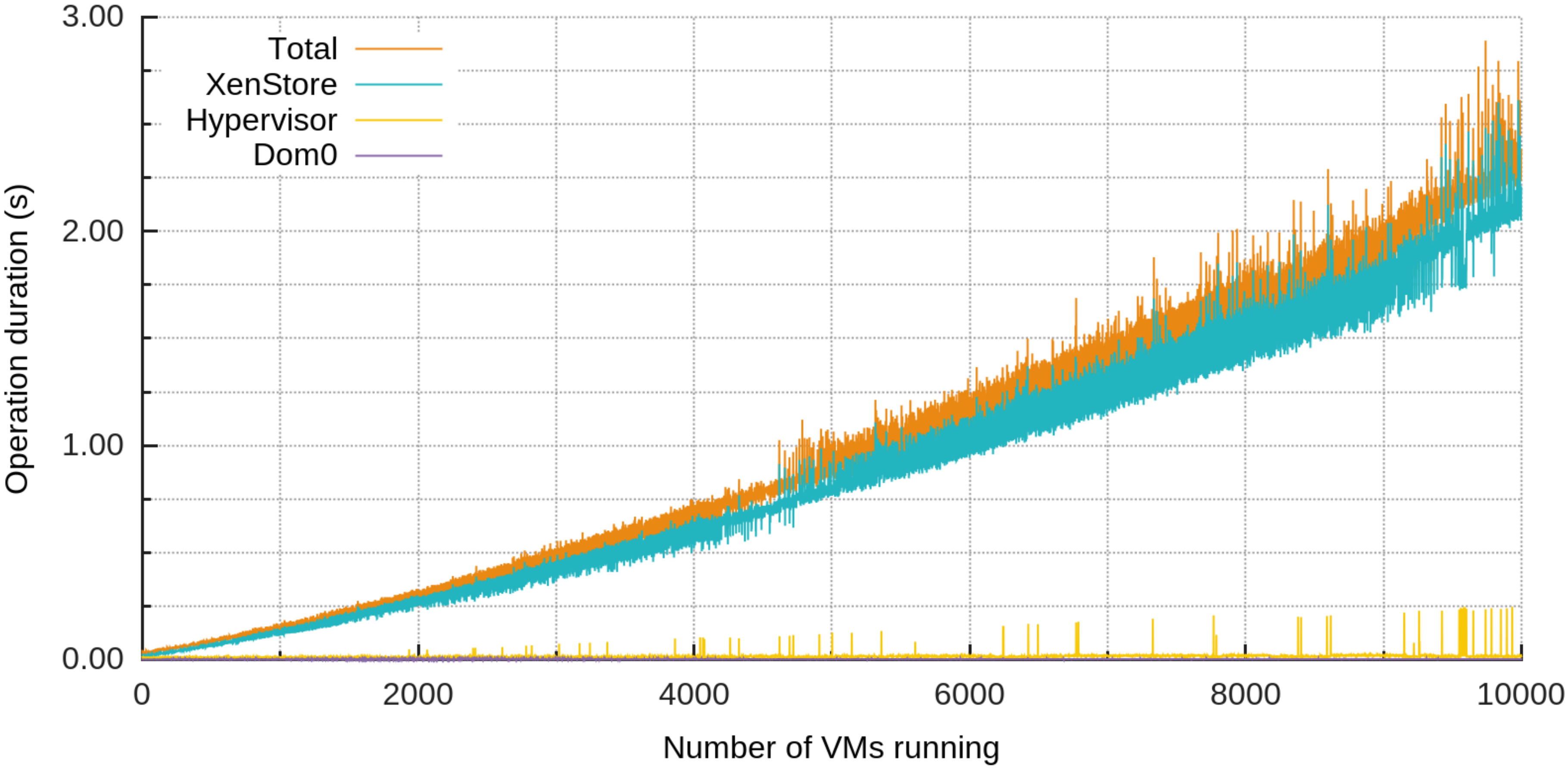
- However...
  - xenstored still CPU heavy
  - xenconsoled still CPU heavy

xenconsoled

- Two major optimizations
  - Move from `poll` to `epoll`
  - On INTRODUCE_DOMAIN, search from last domid
    - Avoid listing all existing domains
- CPU usage down to ~ 10% max.

What Bottlenecks Remain?

Domain Creation Breakdown



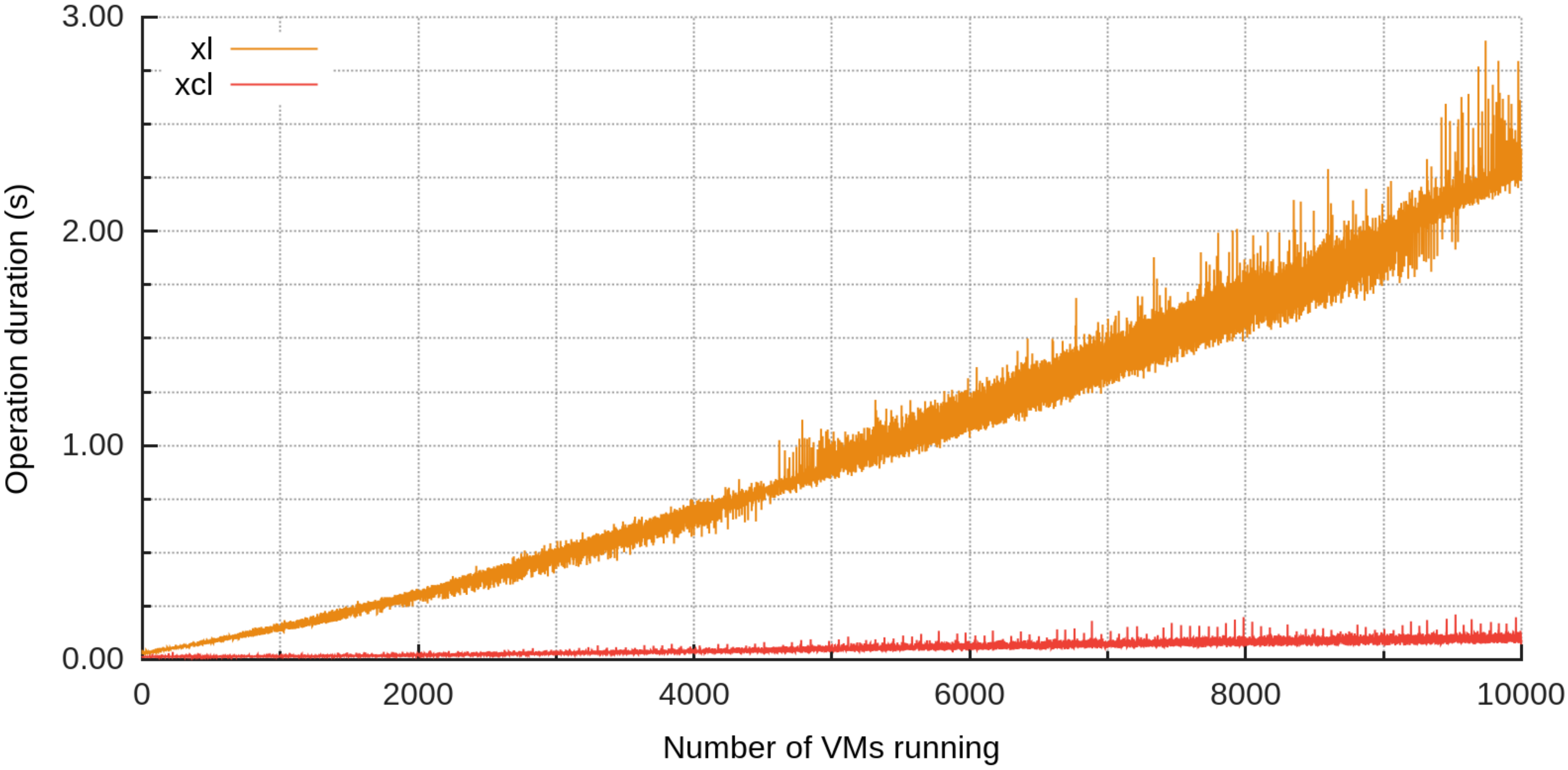
Let's Look at the Toolstack Again

- The domain creation process is too complex for our specialized VMs
  - Also makes the profiling really difficult and inaccurate
  - A lot of unnecessary Xenstore entries
- Some checks take a lot of time
  - Mainly checking for duplicate names

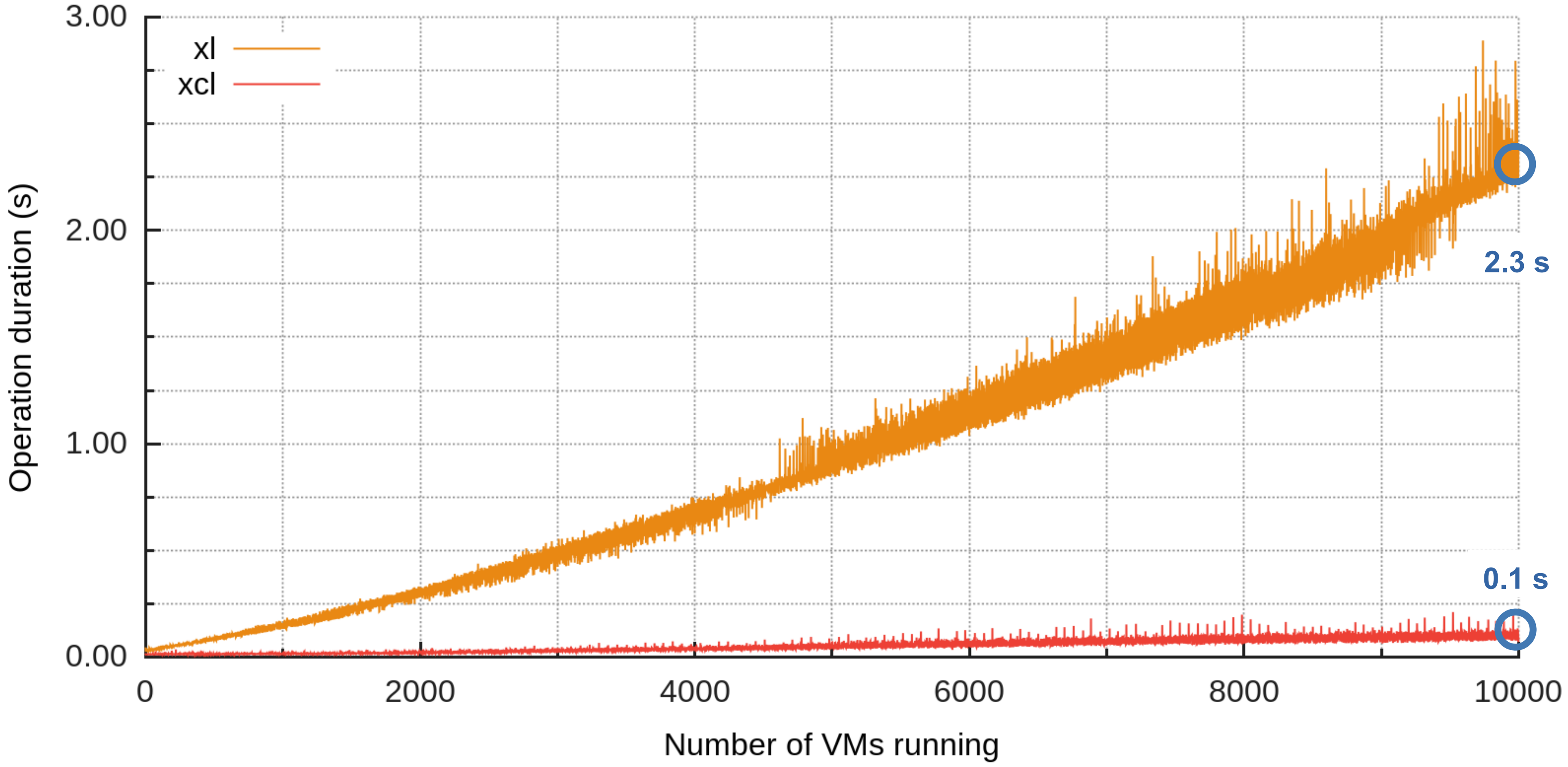
xcl: XenCtrl Light

- A very simplified toolstack
- Small abstraction on top of libxc (~600 LOC)
  - Optimized for our use case
    - Only boots PV and PVH domains
    - Only supports VIFs
  - **Reduced Xenstore usage**
    - From 37 to 17 entries per guest
    - Less Xenstore operations
  - **Doesn't check domain name**

xl vs xcl

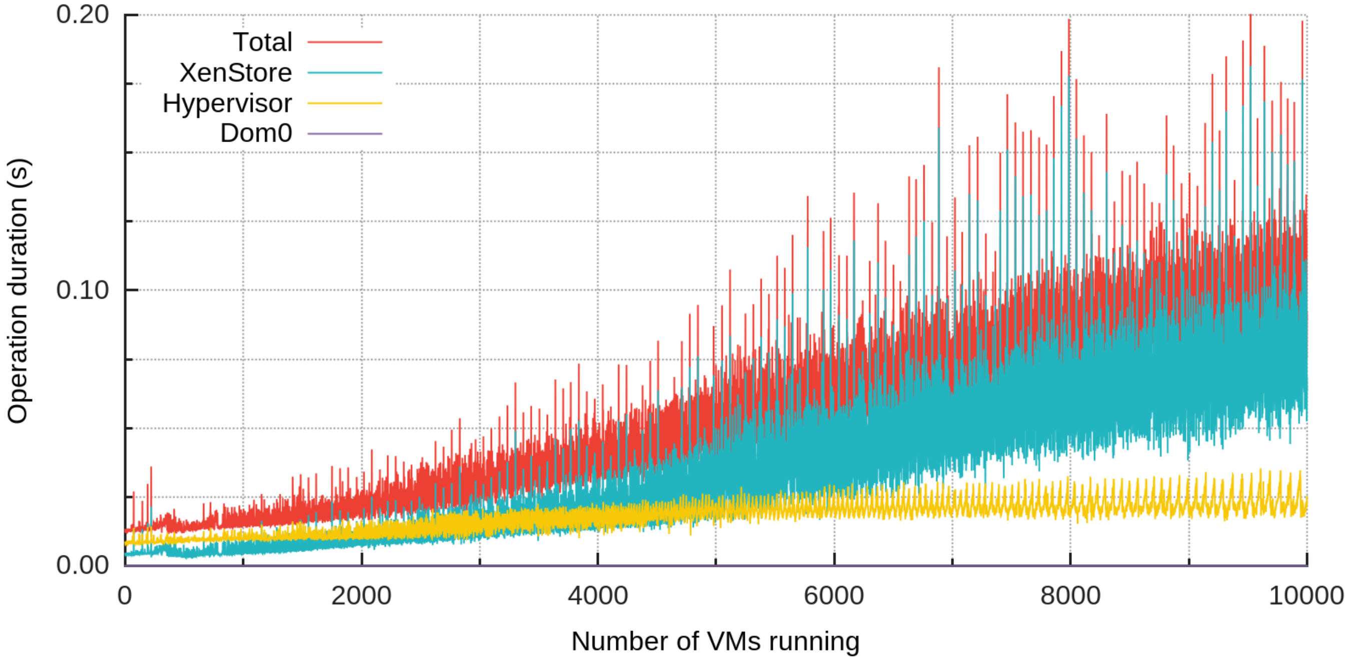

xl
xcl
3.00
2.00
1.00
0.00
Operation duration (s)
0
2000
4000
6000
8000
10000
Number of VMs running

xl vs xcl


3.00
2.00
1.00
0.00
Operation duration (s)
0
2000
4000
6000
8000
10000
Number of VMs running
xl
xcl
2.3 s
0.1 s

With xcl

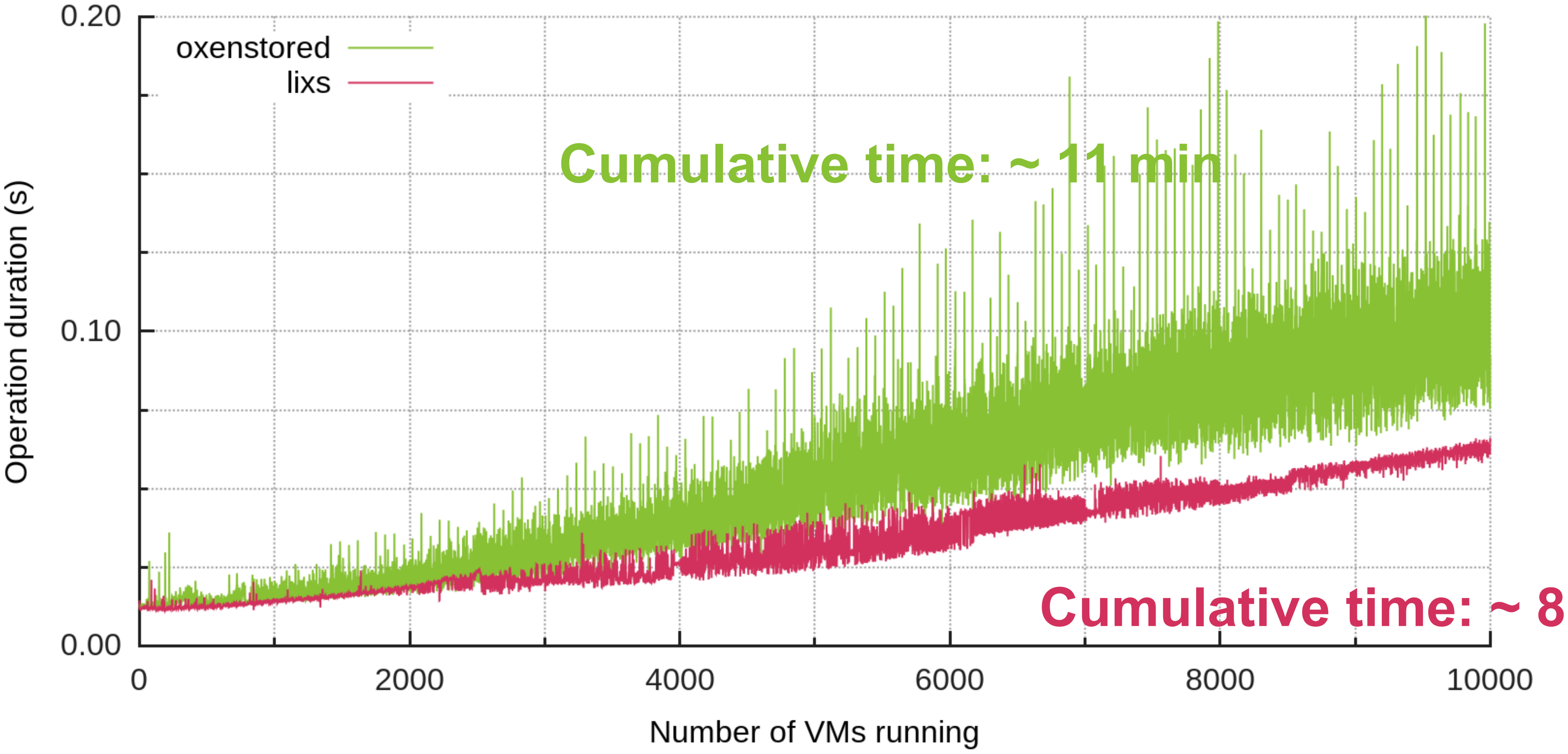
- Much better
- But reducing the number of Xenstore entries is only a palliative
  - Eventually the issue will come back as we increase the number of guests
    - Xenstore remains a major bottleneck



lixs: LIghtweight XenStore

- Work in progress (about 1 month)
- Written from scratch but compatible with the Xenstore protocol
- Currently ~1800 LOC
- C++

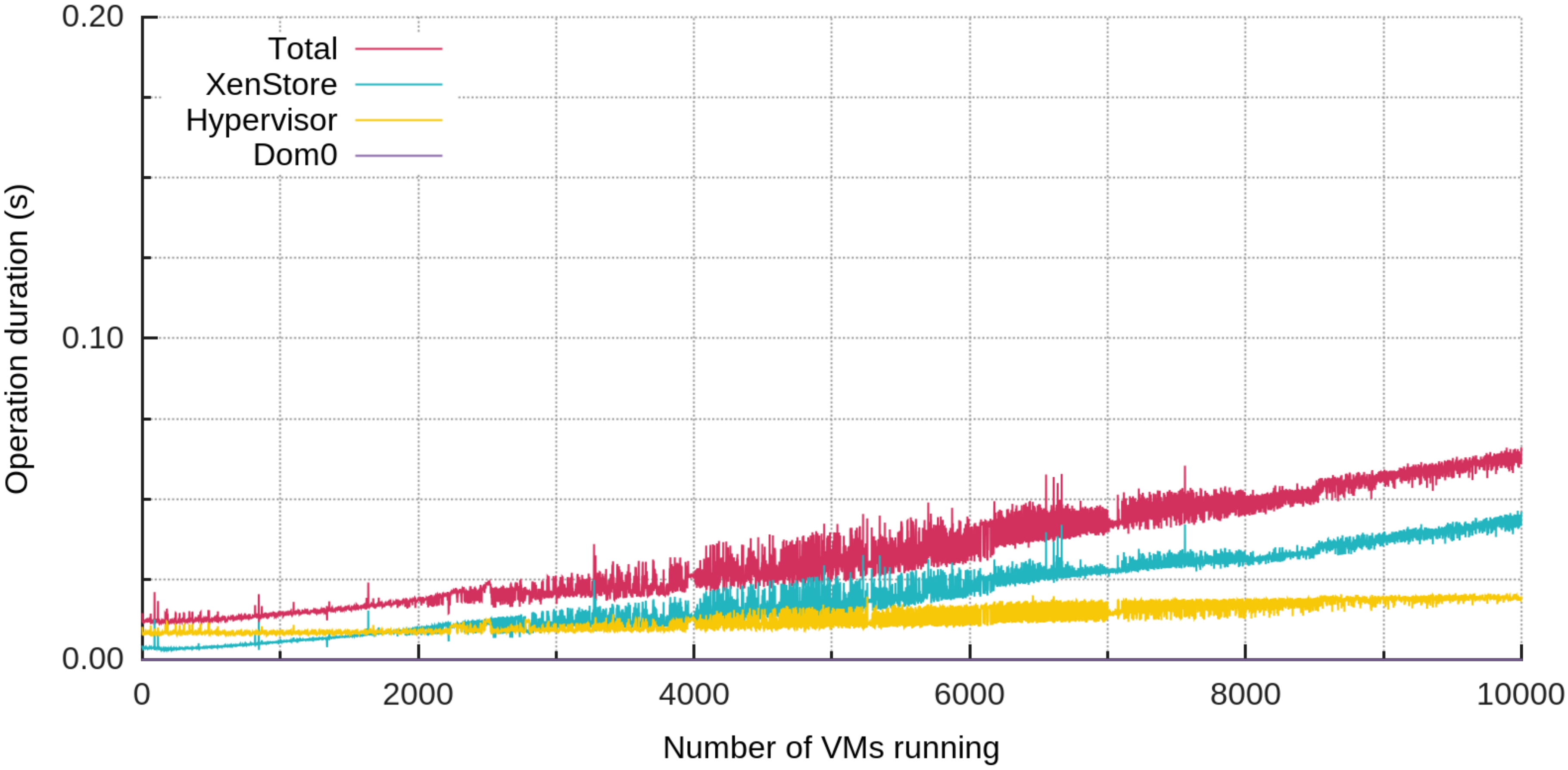
lixs vs oxenstored



Cumulative time: ~ 11 min

Cumulative time: ~ 8

Breakdown with lixs



lixs: Future Work

- Optimize protocol
  - Make Xenstore more specialized
  - Avoid all possible listing operations
- Optimize implementation
  - Remove unix sockets
  - Generic storage backend
    - std::map; noSQL DB; <your backend here>;
    - 10K guests with `std::map` took **10m 3s**
    - 10K guests with `boost::unordered_map` took **7m 54s**

Where are we?

- Usable system running **10K** guests
- 10K guests actually working
  - Although idle most of the time
- Lower domain creation times
  - First domain: **< 10ms**
  - With 10K domains: **< 100ms**

- Recent test: 1,000 VMs running ICMP responder configuration, plus one running content cache (Minicache)
  - **All 1,001 VMs work as expected!**

Will it work? Can we reach 100K?

- There are no fundamental issues with Xen
  - But we only tested it up to 10K guests
- Xenstore protocol needs work
  - Make Xenstore more specialized
  - With 10K+ guests we need to avoid listings

Future work

- Improve lixs and Xenstore protocol
- Multi thousand-port vSwitch
- Have guests doing useful work
- Scheduling
  - Number of guests much bigger than number of cores
  - With that many guests we'll have scheduling issues
- Reducing Memory Usage
  - Smaller image sizes
  - Share memory between guests booting same image

Xenstore Entries: xl vs xcl

XL

```
1 = ""
 vm = "/vm/2baefa82-612c-4e5b-a52d-396a91d5ad7b"
 name = "proxy"
 cpu = ""
  0 = ""
   availability = "online"
 memory = ""
  static-max = "8192"
  target = "8193"
  videoram = "-1"
 device = ""
  suspend = ""
   event-channel = ""
  vif = ""
   0 = ""
    backend = "/local/domain/0/backend/vif/46/0"
    backend-id = "0"
    state = "1"
    handle = "0"
    mac = "00:16:3e:32:ca:23"
   1 = ""
    backend = "/local/domain/0/backend/vif/46/1"
    backend-id = "0"
    state = "1"
    handle = "1"
    mac = "00:16:3e:2e:22:7c"
 control = ""
  shutdown = ""
  platform-feature-multiprocessor-suspend = "1"
  platform-feature-xs_reset_watches = "1"
 data = ""
 domid = "46"
 store = ""
  port = "1"
  ring-ref = "3188551"
 console = ""
  backend = "/local/domain/0/backend/console/46/0"
  backend-id = "0"
  limit = "1048576"
  type = "xenconsoled"
  output = "pty"
  tty = "/dev/pts/1"
  port = "2"
  ring-ref = "3188550"
```

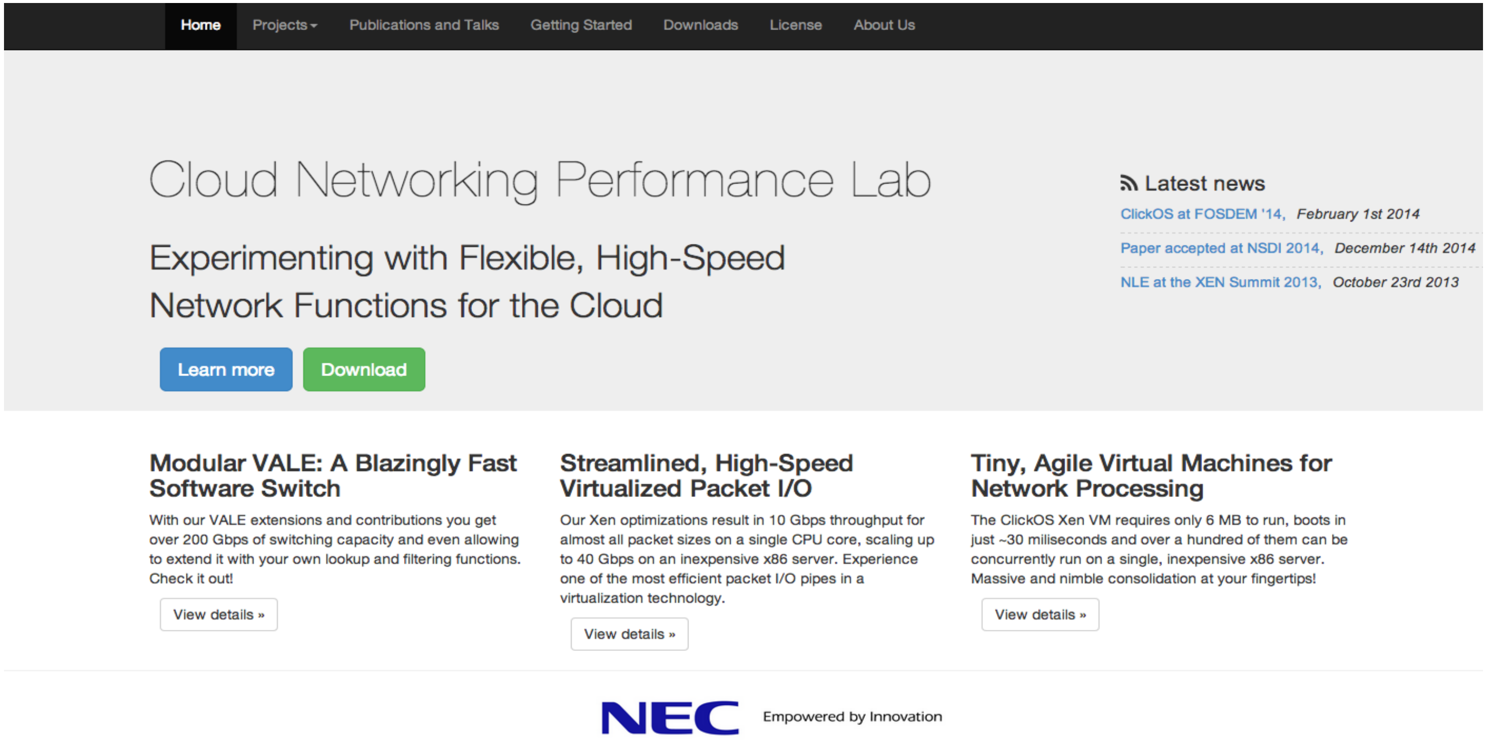
XCL

```
1 = ""
 control = ""
  shutdown = ""
 vm = "/vm/4c3f2a04-e39f-4ad8-9d7f-1b5556f02b34"
 name = "proxy"
 domid = "48"
 console = ""
  port = "2"
  ring-ref = "3157830"
  type = "xenconsoled"
  tty = "/dev/pts/1"
 device = ""
  vif = ""
   0 = ""
    backend = "/local/domain/0/backend/vif/48/0"
    backend-id = "0"
    state = "1"
    handle = "0"
    mac = "00:00:00:00:00:00"
   1 = ""
    backend = "/local/domain/0/backend/vif/48/1"
    backend-id = "0"
    state = "1"
    handle = "1"
    mac = "00:00:00:00:00:00"
```

Number of grants

- 2 grants per domain
  - xenstore; xenconsole;
  - With v1: 512 grants per frame
  - `DEFAULT_MAX_NR_GRANT_FRAMES == 32`
    - Maximum of (512 * 32) / 2 == 8K
- Fix
  - Boot xen with `max_nr_grant_frames=512`
    - Up to 128K domains

It's Open Source!



Cloud Networking Performance Lab

Experimenting with Flexible, High-Speed Network Functions for the Cloud

Latest news

ClickOS at FOSDEM '14, *February 1st 2014*

Paper accepted at NSDI 2014, *December 14th 2014*

NLE at the XEN Summit 2013, *October 23rd 2013*

Modular VALE: A Blazingly Fast Software Switch

With our VALE extensions and contributions you get over 200 Gbps of switching capacity and even allowing to extend it with your own lookup and filtering functions. Check it out!

Streamlined, High-Speed Virtualized Packet I/O

Our Xen optimizations result in 10 Gbps throughput for almost all packet sizes on a single CPU core, scaling up to 40 Gbps on an inexpensive x86 server. Experience one of the most efficient packet I/O pipes in a virtualization technology.

Tiny, Agile Virtual Machines for Network Processing

The ClickOS Xen VM requires only 6 MB to run, boots in just ~30 miliseconds and over a hundred of them can be concurrently run on a single, inexpensive x86 server. Massive and nimble consolidation at your fingertips!

NEC Empowered by Innovation

Checkout

- ClickOS, Backend Switch, Xen optimizations and more!
- Tutorials
- Better performance than listed in the papers!

We are always looking for…

Interns
Visiting researchers
Collaborations

(and often full-time researchers)

Interested? felipe.huici@neclab.eu

© NEC Corporation 2009

Empowered by Innovation

NEC