Improving Scalability of Xen: the 3,000 domains experiment

Wei Liu <wei.liu2@citrix.com>

Xen: the gears of the cloud

- large user base

  estimated more than 10 million individuals users

- power the largest clouds in production

- not just for servers

rackspace
the open cloud company

amazon web services™

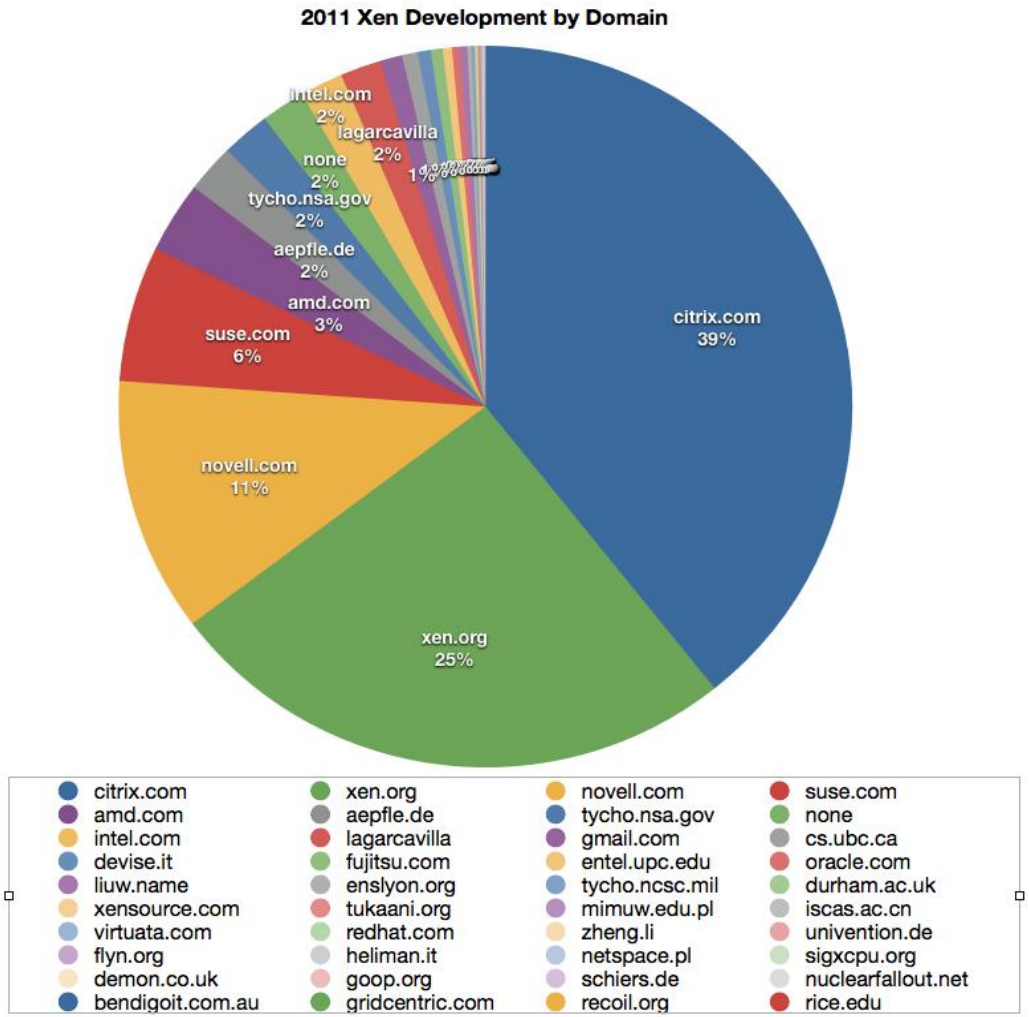
linode.com

QUBESOS

Xen: Open Source

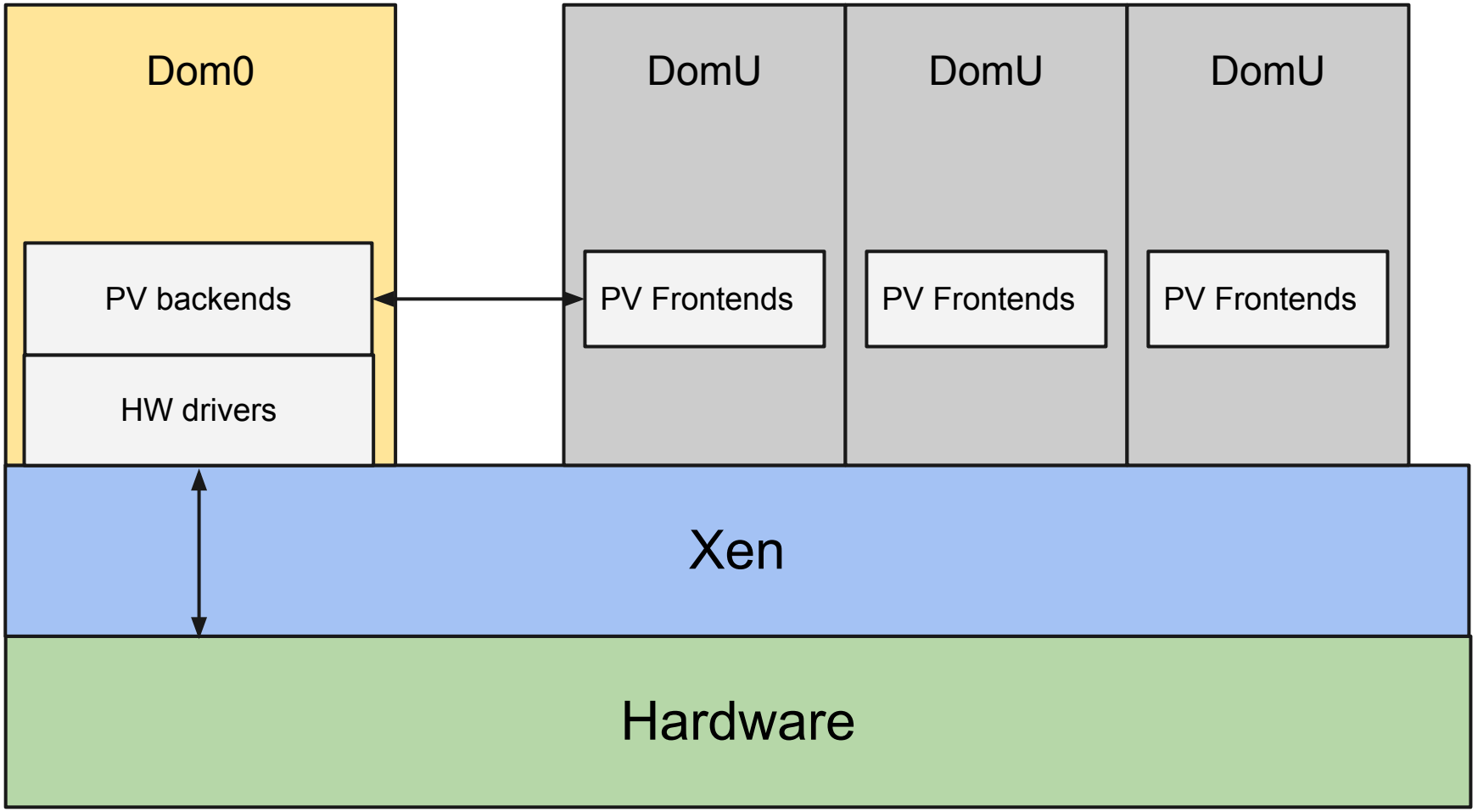
GPLv2 with DCO (like Linux)

Diverse contributor community

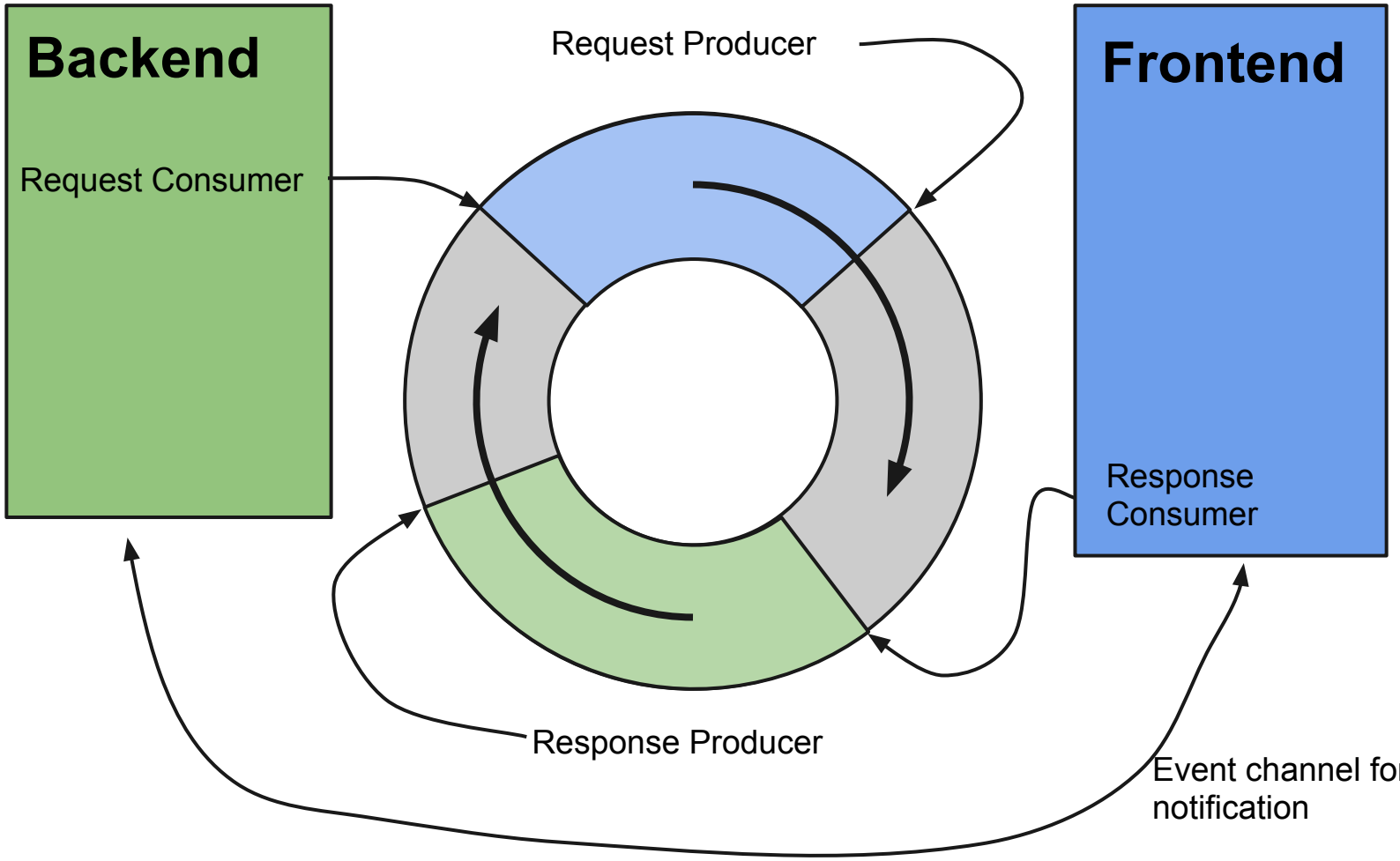


source:
Mike Day
http://code.ncultra.org

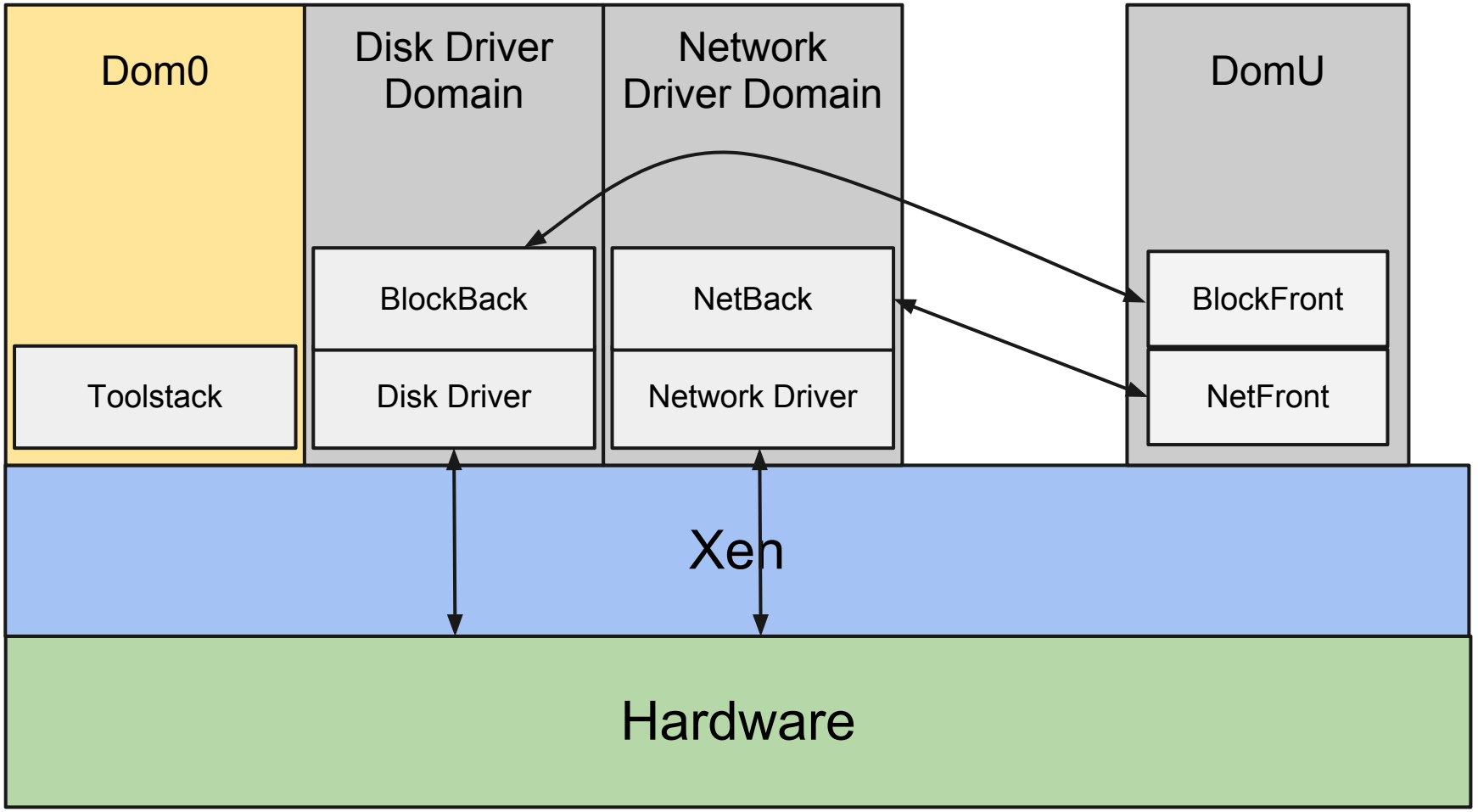
Xen architecture: PV guests



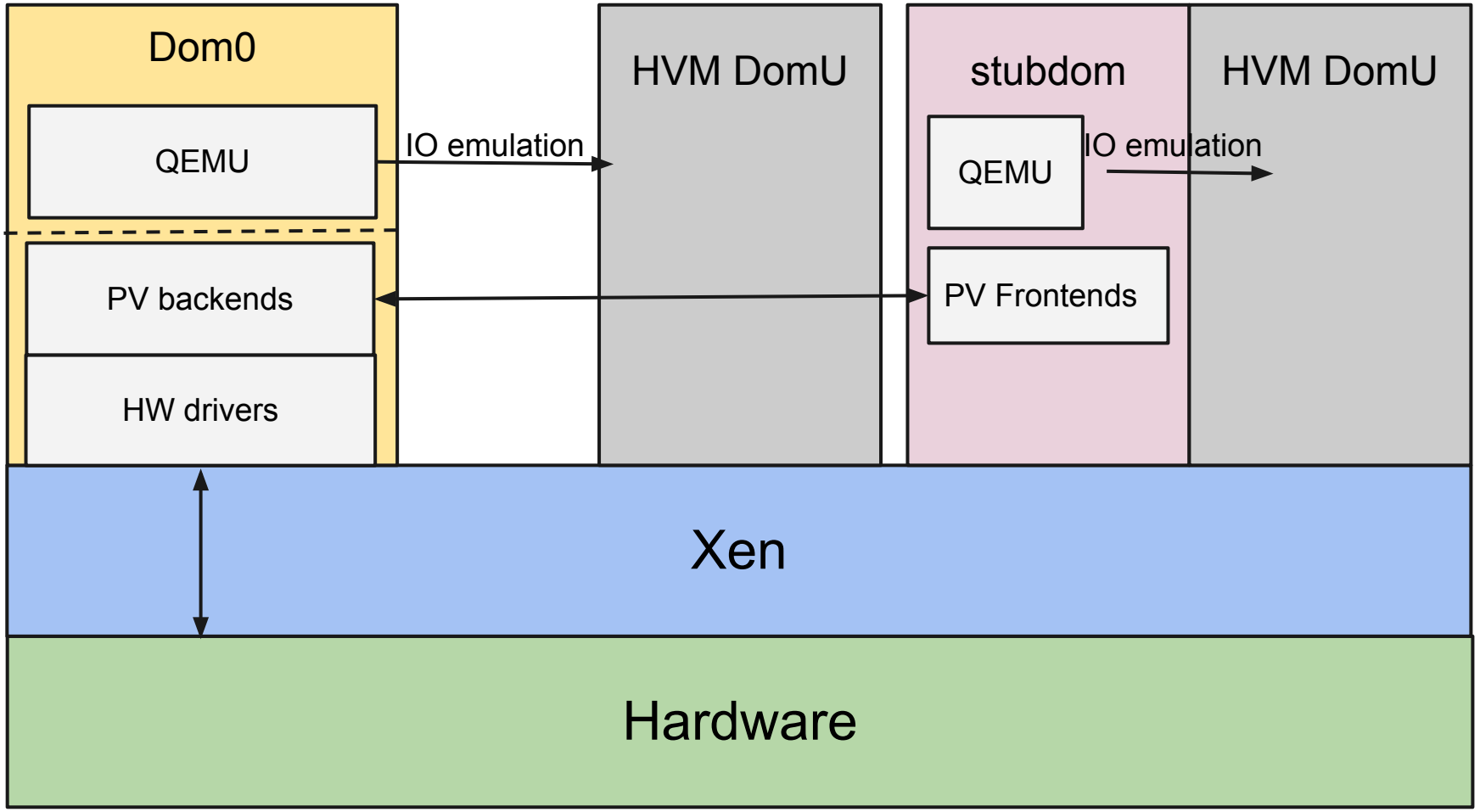
Xen architecture: PV protocol



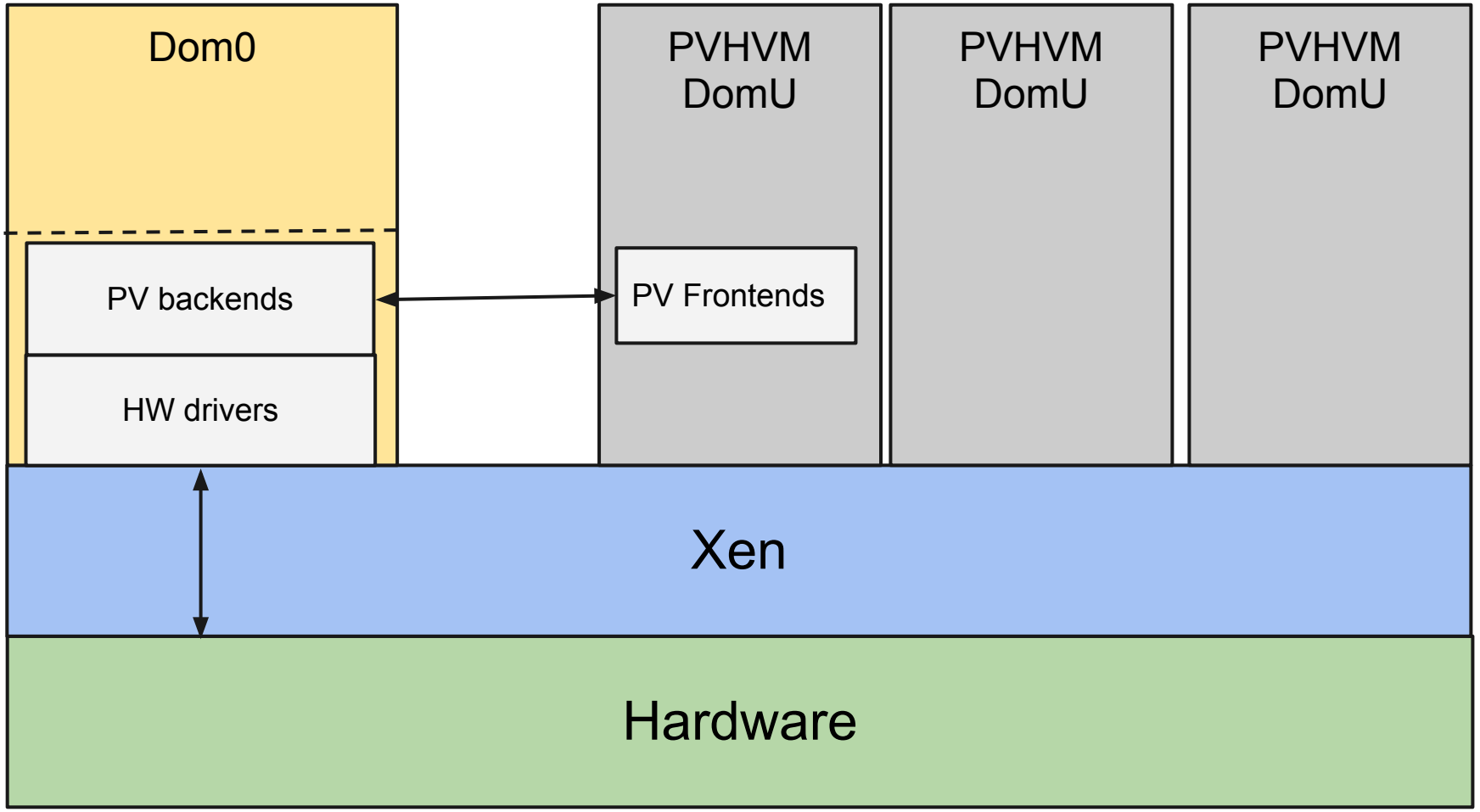
Xen architecture: driver domains



Xen architecture: HVM guests



Xen architecture: PVHVM guests



Xen scalability: current status

Xen 4.2

- Up to 5TB host memory (64bit)
- Up to 4095 host CPUs (64bit)
- Up to 512 VCPUs for PV VM
- Up to 256 VCPUs for HVM VM
- Event channels
  - 1024 for 32-bit domains
  - 4096 for 64-bit domains

Xen scalability: current status

Typical PV / PVHVM DomU

- 256MB to 240GB of RAM
- 1 to 16 virtual CPUs
- at least 4 inter-domain event channels:
  - xenstore
  - console
  - virtual network interface (vif)
  - virtual block device (vbd)

Xen scalability: current status

- From a backend domain's (Dom0 / driver domain) PoV:
  - IPI, PIRQ, VIRQ: related to number of cpus and devices, typical Dom0 has 20 to ~200
  - yielding less than 1024 guests supported for 64-bit backend domains and even less for 32-bit backend domains
- 1K still sounds a lot, right?
  - enough for normal use case
  - not ideal for OpenMirage (OCaml on Xen) and other similar projects

Start of the story

- Effort to run 1,000 DomUs (modified Mini-OS) on a single host *
- Want more? How about 3,000 DomUs?
  - definitely hit event channel limit
  - toolstack limit
  - backend limit
  - open-ended question: is it practical to do so?

* http://lists.xen.org/archives/html/xen-users/2012-12/msg00069.html

Toolstack limit

xenconsoled and cxenstored both use select(2)

- xenconsoled: not very critical and can be restarted
- cxenstored: critical to Xen and cannot be shutdown otherwise lost information
- oxenstored: use libev so there is no problem

switch from select(2) to poll(2)

implement poll(2) for Mini-OS

Event channel limit

Identified as key feature for 4.3 release. Two designs came up by far:

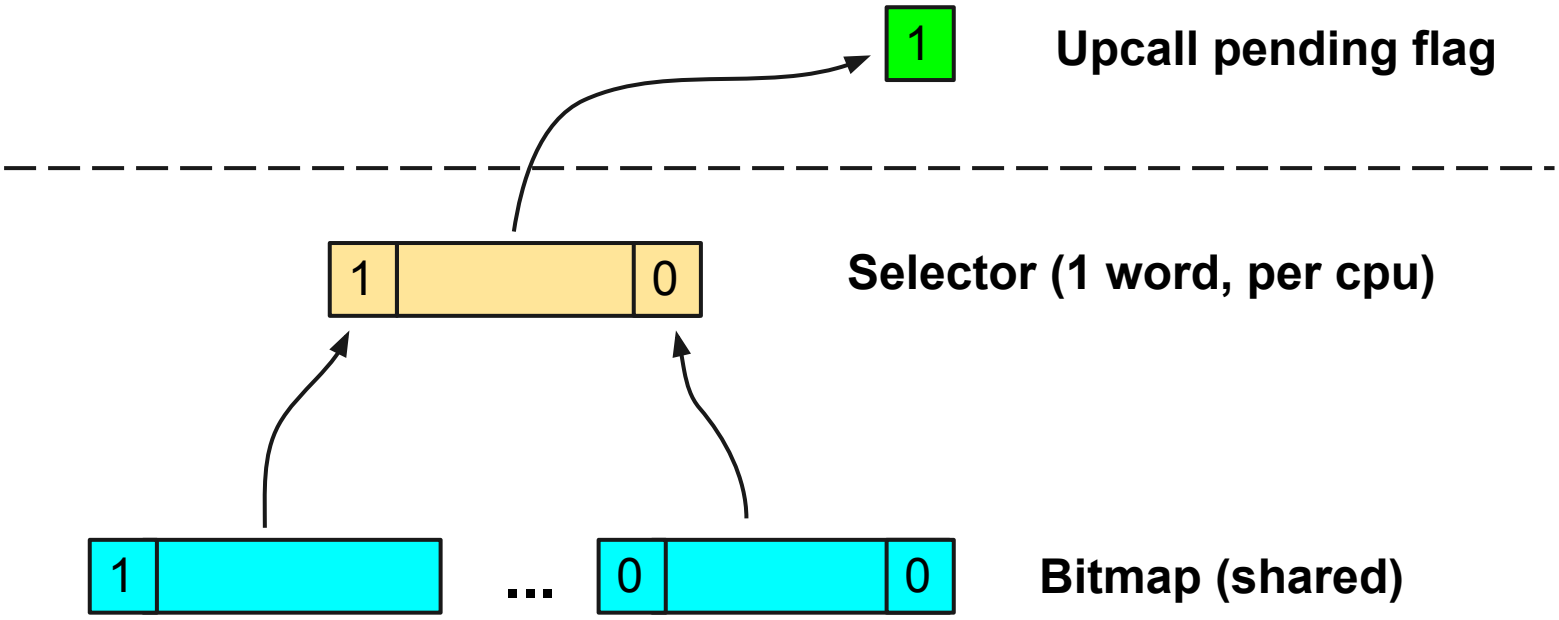
- 3-level event channel ABI
- FIFO event channel ABI

3-level ABI

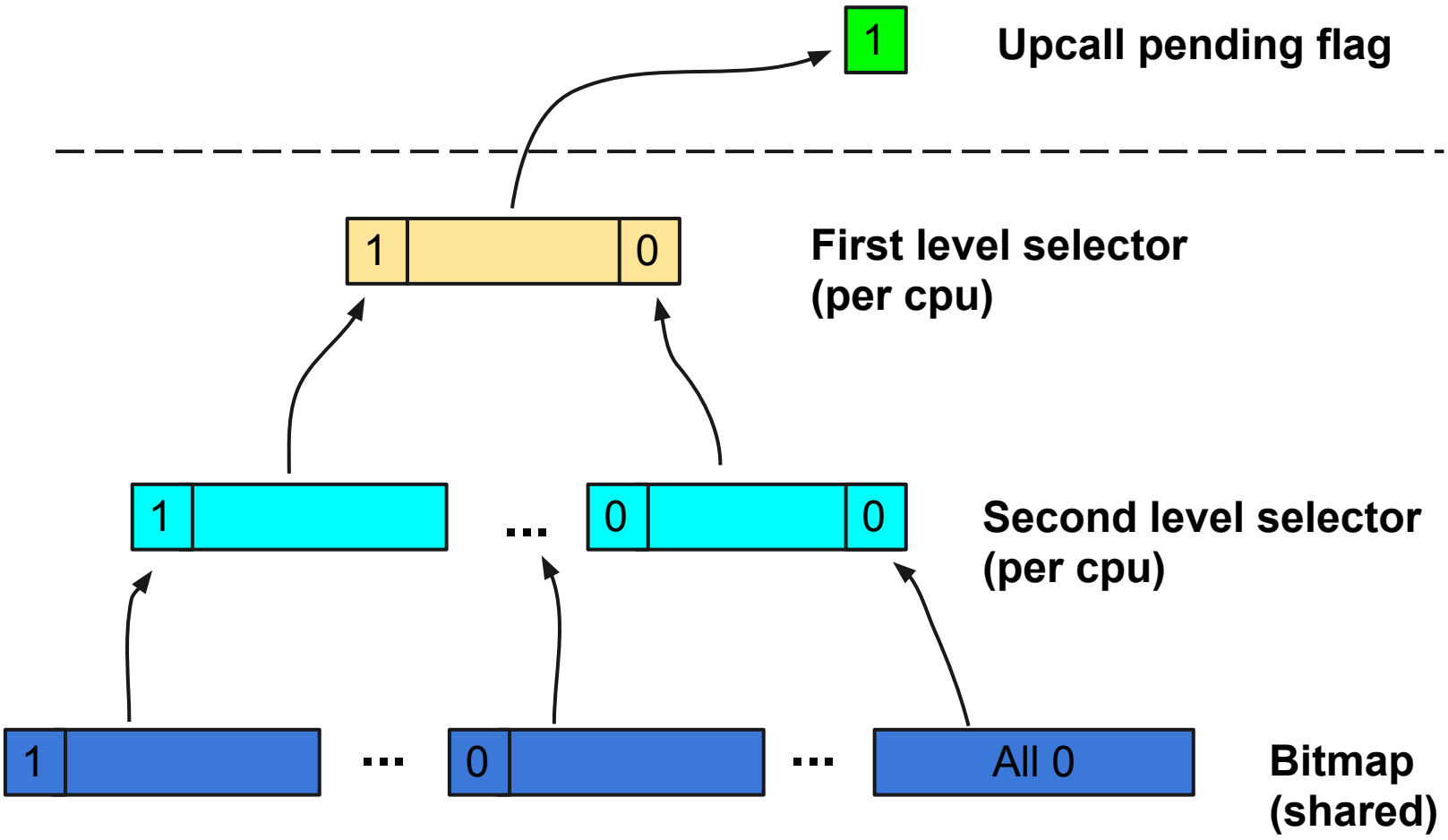
Motivation: aimed for 4.3 timeframe

- an extension to default 2-level ABI, hence the name
- started in Dec 2012
- V5 draft posted Mar 2013
- almost ready

Default (2-level) ABI



3-level ABI


1
Upcall pending flag
1
0
First level selector
(per cpu)
1
...
0
0
Second level selector
(per cpu)
1
...
0
...
All 0
Bitmap
(shared)

3-level ABI

Number of event channels:

- 32K for 32 bit guests
- 256K for 64 bit guests

Memory footprint:

- 2 bits per event (pending and mask)
- 2 / 16 pages for 32 / 64 bit guests
- NR_VCPUS pages for controlling structure

Limited to Dom0 and driver domains

3-level ABI

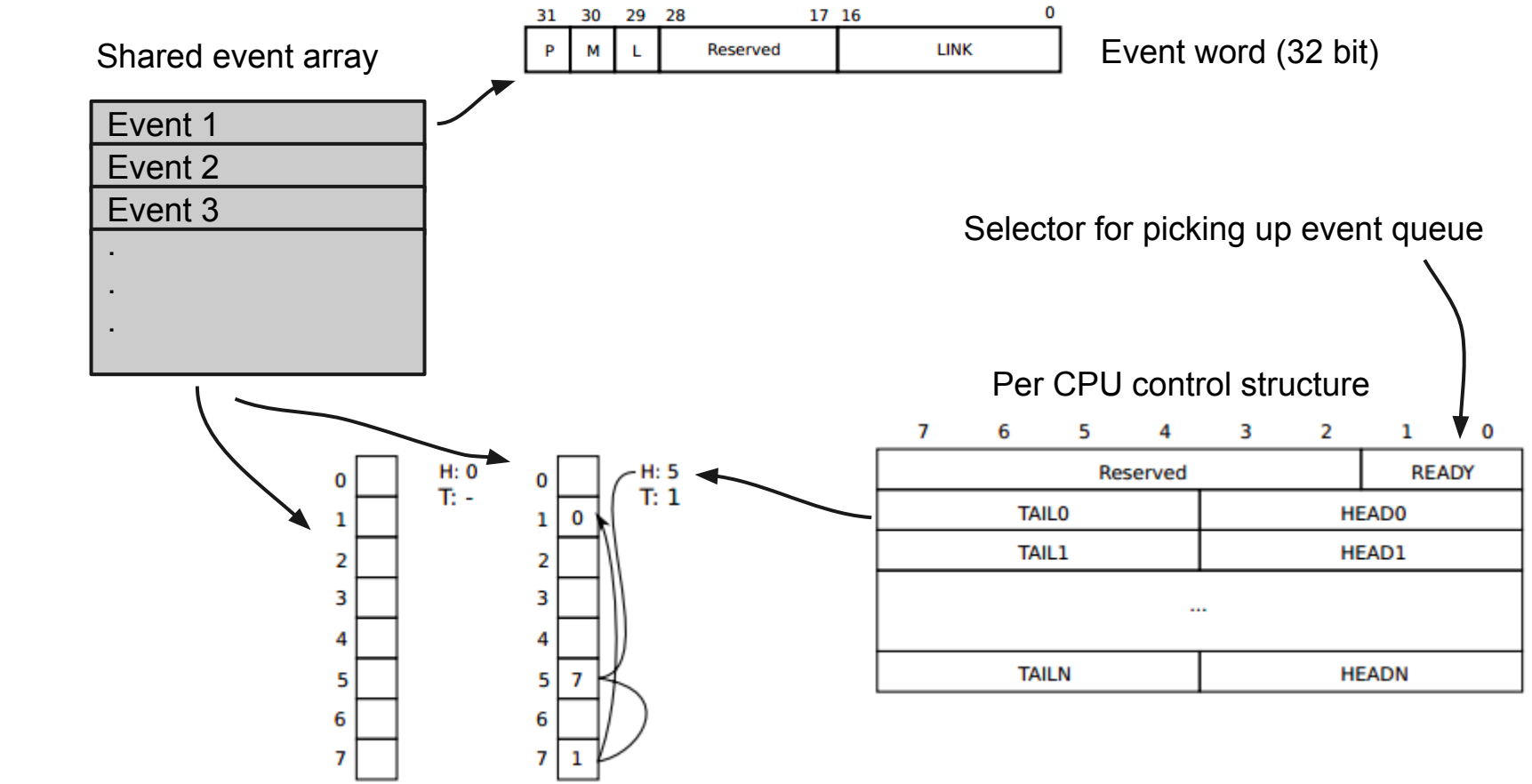
- Pros
  - general concepts and race conditions are fairly well understood and tested
  - envisioned for Dom0 and driver domains only, small memory footprint
- Cons
  - lack of priority (inherited from 2-level design)

FIFO ABI

Motivation: designed ground-up with gravy features

- design posted in Feb 2013
- first prototype posted in Mar 2013
- under development, close at hand

FIFO ABI



Empty queue and non-empty queue (only showing the LINK field)

FIFO ABI

Number of event channels:

- 128K (2^17) by design

Memory footprint:

- one 32-bit word per event
- up to 128 pages per guest
- NR_VCPUS pages for controlling structure

Use toolstack to limit maximum number of event channels a DomU can have

FIFO ABI

- Pros
  - event priority
  - FIFO ordering
- Cons
  - relatively large memory footprint

Community decision

- scalability issue not as urgent as we thought
  - only OpenMirage expressed interest on extra event channels
- delayed until 4.4 release
  - better to maintain one more ABI than two
  - measure both and take one
- leave time to test both designs
  - event handling is complex by nature

Back to the story

3,000 DomUs experiment

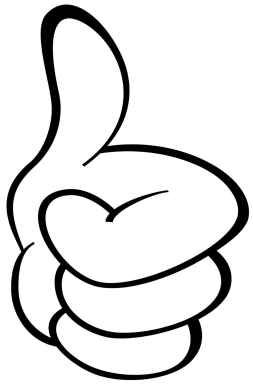
3,000 Mini-OS

Hardware spec:

- 2 sockets, 4 cores, 16 threads
- 24GB RAM

Software config:

- Dom0 16 VCPUs
- Dom0 4G RAM
- Mini-OS 1 VCPU
- Mini-OS 4MB RAM
- Mini-OS 2 event channels



DEMO

3,000 Linux

Hydramonster hardware spec:

- 8 sockets, 80 cores, 160 threads
- 512GB RAM

Software config:

- Dom0 4 VCPUs (pinned)
- Dom0 32GB RAM
- DomU 1 VCPU
- DomU 64MB RAM
- DomU 3 event channels (2 + 1 VIF)

Observation

Domain creation time:

- < 500 acceptable
- > 800 slow
- took hours to create 3,000 DomUs

Observation

Backend bottleneck:

- network bridge limit in Linux
- PV backend drivers buffer starvation
- I/O speed not acceptable
- Linux with 4G RAM can allocate ~45k event channels due to memory limitation

Observation

CPU starvation:

- density too high:1 PCPU vs ~20 VCPUs
- backend domain starvation
- should dedicate PCPUs to critical service domain

Summary

Thousands of domains, doable but not very practical at the moment

- hypervisor and toolstack
  - speed up creation
- hardware bottleneck
  - VCPU density
  - network / disk I/O
- Linux PV backend drivers
  - buffer size
  - processing model

Beyond?

Possible practical way to run thousands of domains:

Disaggregation

offload services to dedicated domains and trust Xen scheduler.

Happy hacking and have fun!

Q&A

Acknowledgement

Pictures used in slides:

thumbsup:

http://primary3.tv/blog/uncategorized/cal-state-university-northridge-thumbs-up/

hydra: http://www.pantheon.org/areas/gallery/mythology/europe/greek_people/hydra.html