CTSRD

CRASH-worthy Trustworthy Systems Research and Development

CheriBSD: a research fork of FreeBSD

Brooks Davis
SRI International

BSDCan, Ottawa, Canada
June 12, 2015

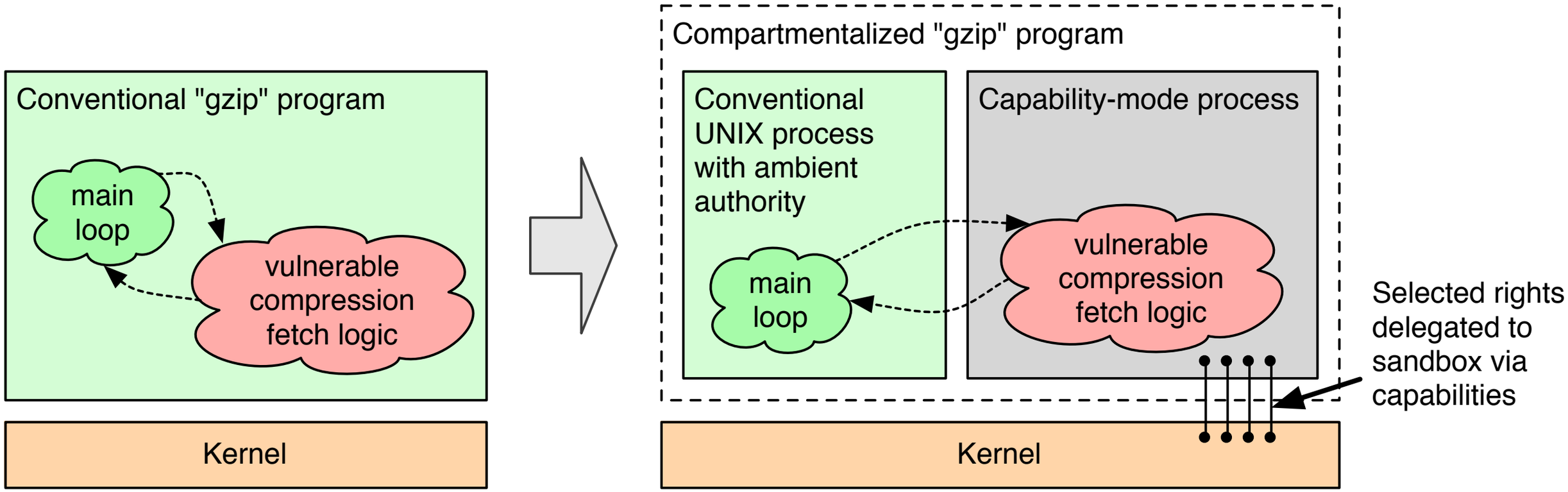
Approved for public release; distribution is unlimited. This research is sponsored by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL), under contract FA8750-10-C-0237. The views, opinions, and/or findings contained in this article/presentation are those of the author(s)/ presenter(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

Banks lose over $300m

Office of Personnel Management hacked

80 million customer records

Application Compartmentalization

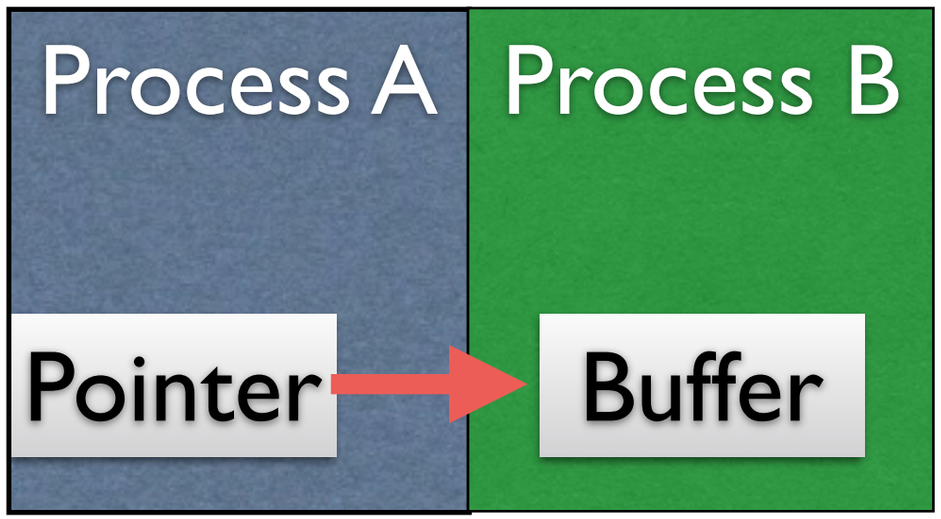


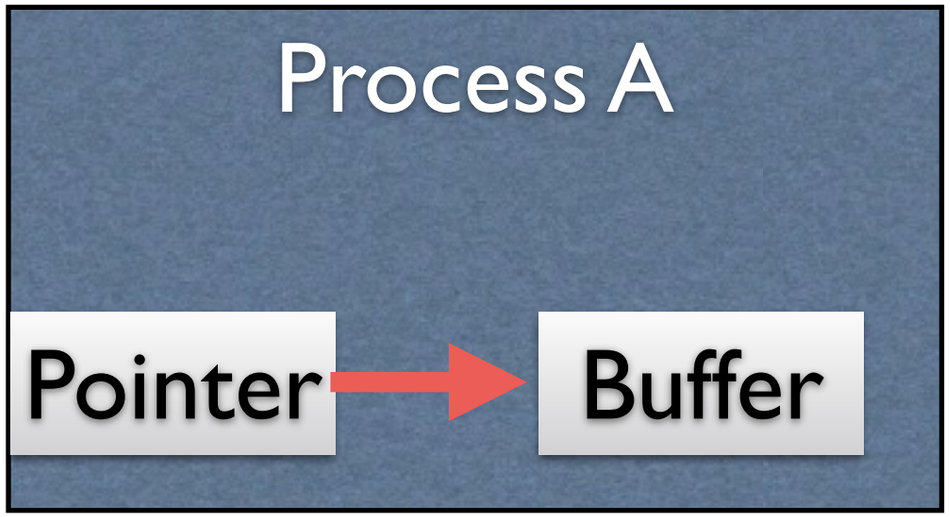
- Compartmentalization decomposes software into isolated components.
- Each sandbox runs with only the rights required to perform its function.
- This model implements the principle of least privilege.

Capsicum

- Hybrid capability model: OS APIs for application compartmentalization
- Out-of-the box in FreeBSD10.0
- Growing number of FreeBSD programs are using Capsicum out-of-the-box: tcpdump, auditdistd, hastd, etc.
- Casper framework offers services to sandboxes (e.g., DNS, socket server)
- Google has published a Linux port

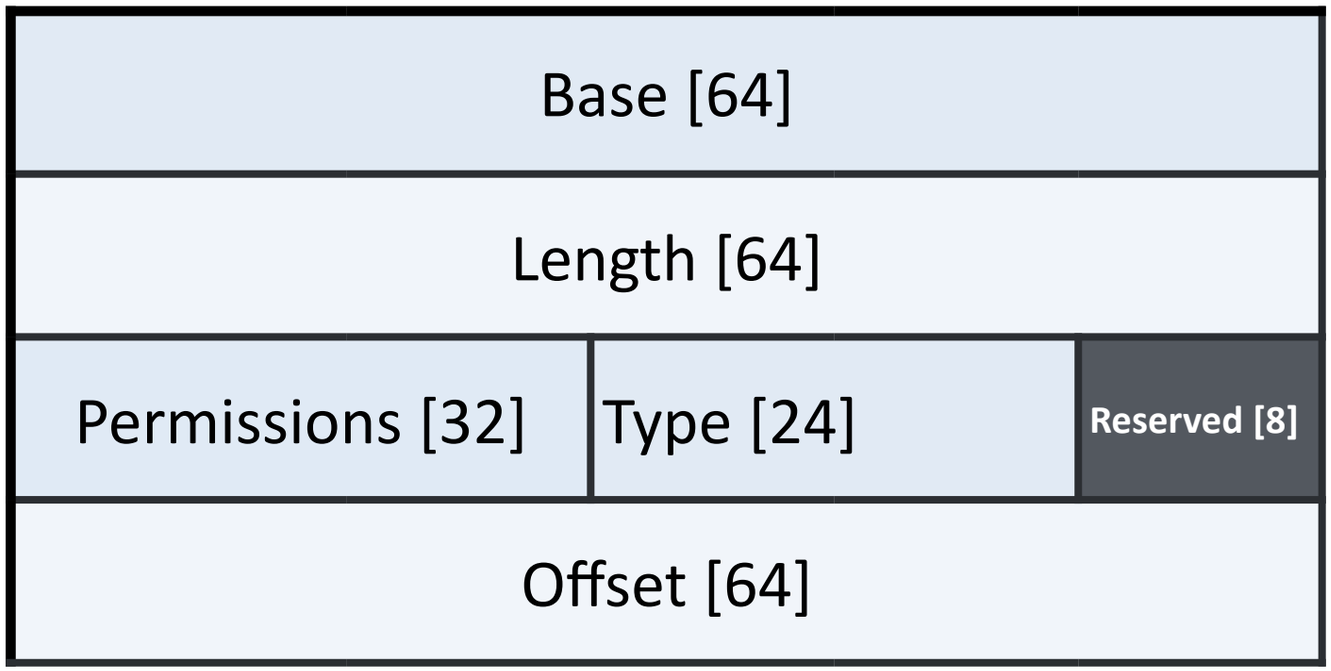
From compartments to objects





- Sharing requires pointers with enforced bounds and permissions
- Can we use this mechanism for every pointer?

CHERI capabilities



- Unforgeable
- Monotonic length and permissions
- Tagged memory protects capabilities
- Checks apply only on dereference

C language support

Hybrid:

- `__capability` annotations on pointers
- Small changes in the C runtime

Pure:

- Compiler compiles code with all pointers are capabilities
- Small application changes to maximize memory safety

Binary compatibility

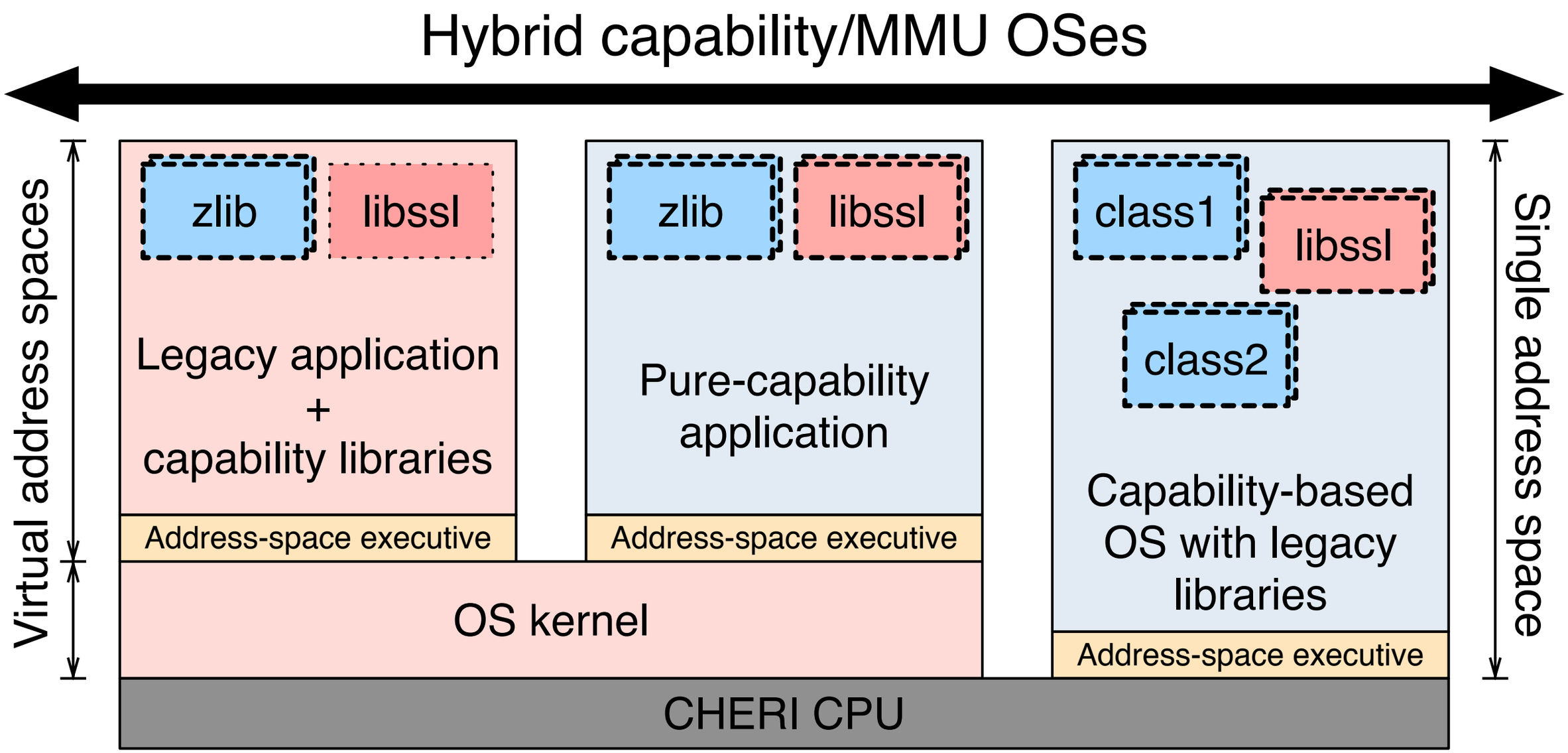
More compatible

More safe

n64
Pure MIPS

Hybrid
Some pointers are capabilities

Pure-capability
All pointers are capabilities


Hybrid capability/MMU OSes
Virtual address spaces
zlib
libssl
Legacy application
+
capability libraries
Address-space executive
zlib
libssl
Pure-capability
application
Address-space executive
OS kernel
class1
libssl
class2
Capability-based
OS with legacy
libraries
Address-space executive
Single address space
CHERI CPU

The prototype CPU

- 64-bit MIPS-compatible ISA (≈R4000)
- CHERI ISA extensions
- Runs at 100MHz on FPGA
- Full software stack

CheriBSD supports CHERI

- Platform support (BERI CPU)
- Support for new ISA features
- Infrastructure for compartmentalization
- Custom and adapted applications
- Build system improvements

Lots of deltas!

CTSRD-CHERI / **cheribsd**
forked from freebsd/freebsd

Unwatch 17

FreeBSD adapted for the CHERI CPU. WARNING: some programs contain deliberate vulnerabilities
http://www.cl.cam.ac.uk/research/security/ctsrd/cheri.html — Edit

215,467 commits **403** branches **21** releases **176** contributors

branch: **master** **cheribsd / +**

This branch is 5988 commits ahead, 620 commits behind freebsd:master Pull Request Compare

When CPU_CHERI_CSETBOUNDS is used, force use of the CSetBounds even on ···

rwatson authored a day ago latest commit `a23a8b6b32`

Kernel changes

Component	Files	Lines +	Lines -
Headers	19	1424	11
CHERI initalization	2	49	4
Context managment	2	392	10
Exception handling	3	574	90
Memory copying	2	122	0
Virtual memory	5	398	27
Object capailities	2	883	0
System calls	2	76	0
Signal delivery	3	327	71
Process monitoring/debugging	3	298	0
Kernel debugger	2	264	0

libc changes

- Capability aware memcpy, memmove, etc
- Explicit capability forms of mem* and and str* functions (memcpy_c, memcpy_c_fromcap, memcpy_c_tocap)
- Fixing optimizations based on assumptions about pages
- Split of syscalls and libc (coming soon!)

libcheri

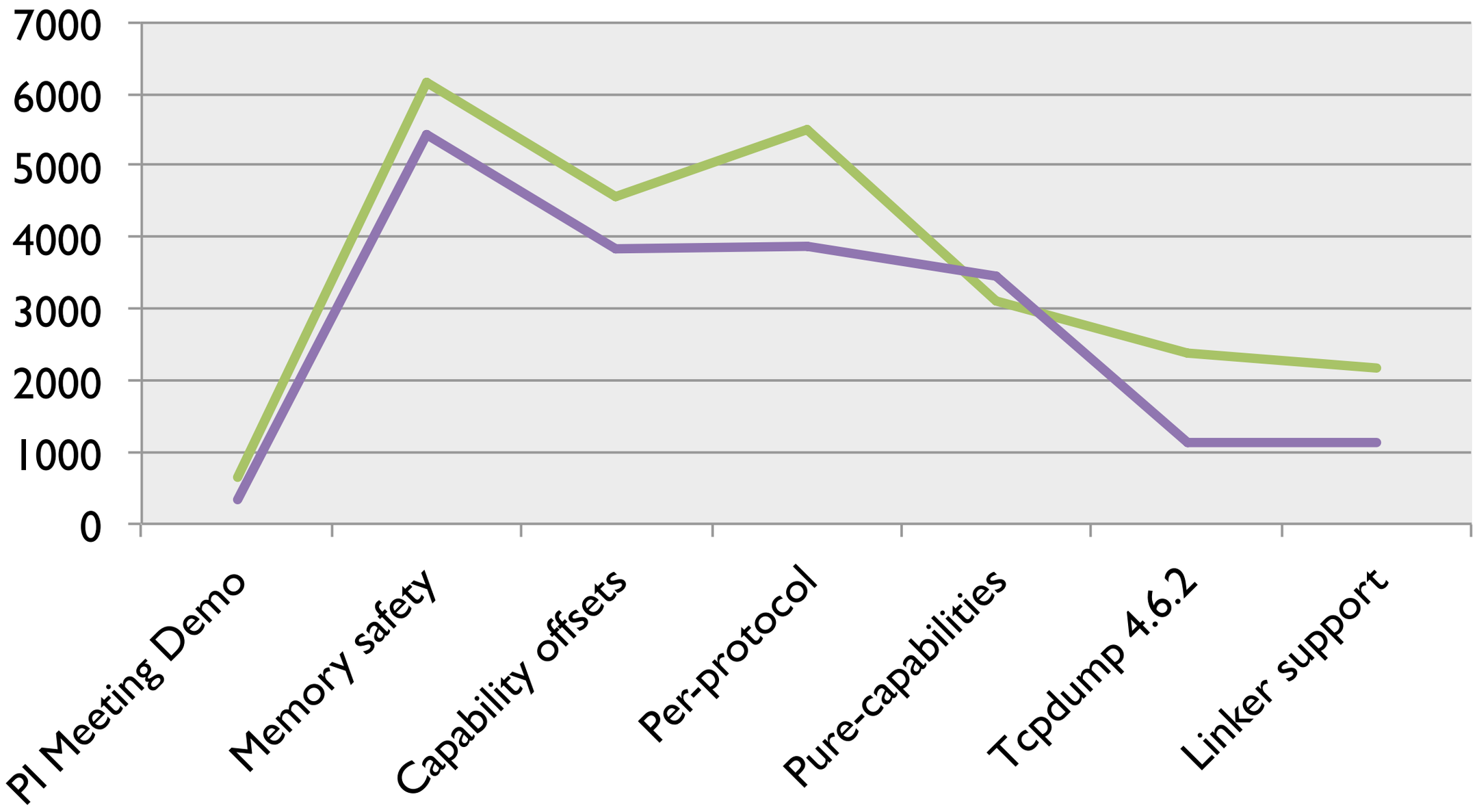
- Compartment object management
- Type allocator
- Loader and runtime linker
- System call implementation for compartments

/usr/libcheri

- Similar to /usr/lib32
- Builds all libraries in pure-capability mode
- Allows for pure-capability programs on a MIPS64 system

Demo Applications

Tcpdump changes



Infrastructure

- Build system improvements
  - Unprivileged builds
  - Per-program (and per-file) compiler replacement
  - Strip during build, not at install
  - …

bringing MIPS

Porting FreeBSD

FreeBSD Journal
http://freebsdjournal.org

to a new CPU, even within a previously supported family, is a significant undertaking.

Early days: Perforce

Pros

- FreeBSD infrastructure
- Good merging
- Easy to maintain stacked branches
- Familiar to team

Cons

- No public access
- Hard to add users
  - Not ideal for CI
- Minimal offline support

Perforce ⇒ Github

- Switched October 2013
- Lost some history granularity
- Easy public access
- Trial by fire with git-at-scale

Github model

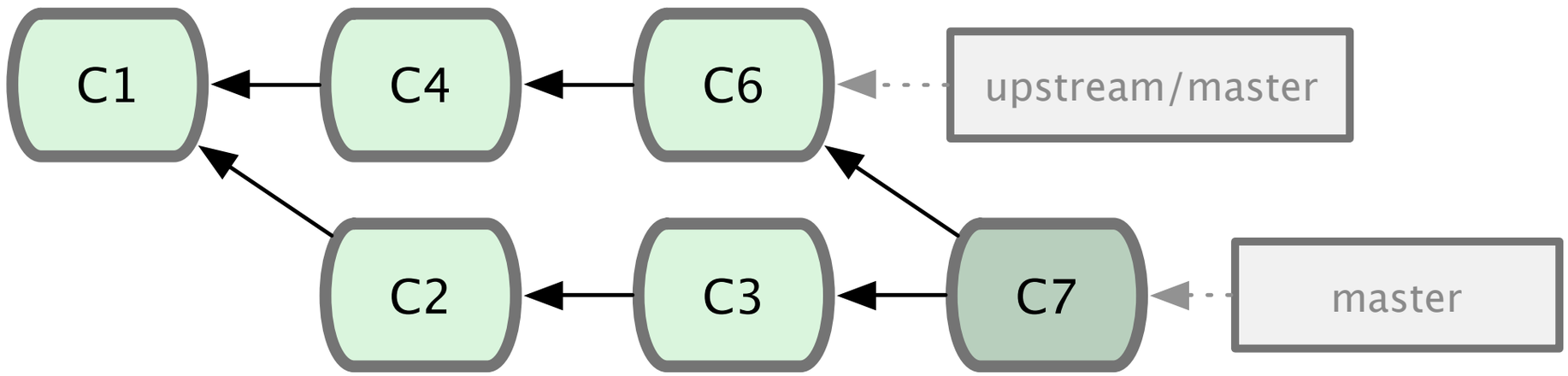
- Forked freebsd/freebsd repo
  - Weird effect: forking CheriBSD seems to fork FreeBSD
- All commits to master branch
- Merge changes from FreeBSD upstream

Merging: first attempt

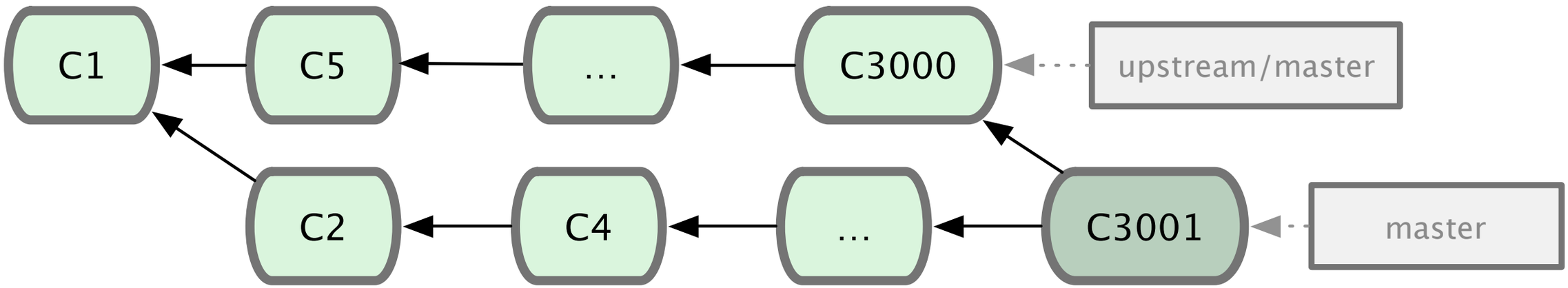
```
git fetch upstream

git merge upstream/master
```

- Merges everything at once!
- Works
- Rebase usually produces insane results
  - Don’t lose the push race!

Oops, we merged a bug!



Bisect is useless



mergify

- Merge one commit at a time
  - Mostly true assumption that commits are complete features
- Stream of small changes merging upstream and cheribsd
- Bisect is possible

mergify

- Problem: merging `tcpdump` went weird
- Vendor commits have the empty repo as a common parent with `master`!
- Solution: merge only direct commits

`mergify` Demo

Git rebase is broken

- Changes are reapplied in order
- Including merges from vendor branches!
- `mergify` doesn't fix this (yet)
- May be an issue with using git wrong or git-svn not handling vendor branches well

`mergify` TODOs

- rebase mode
- bisect mode
- check that things build/work at key points

Upstreaming

- Reduce merge conflicts
- What to upstream?
  - Drivers for things people can use
  - General infrastructure
  - Infrastructure shared by multiple external consumers
  - Low impact things that are conflict prone

What we've upstreamed

- FDT support for MIPS
- Drivers and driver improvements
- Working floating point support for MIPS
- Boot loaders for MIPS
- Unprivileged builds and installs

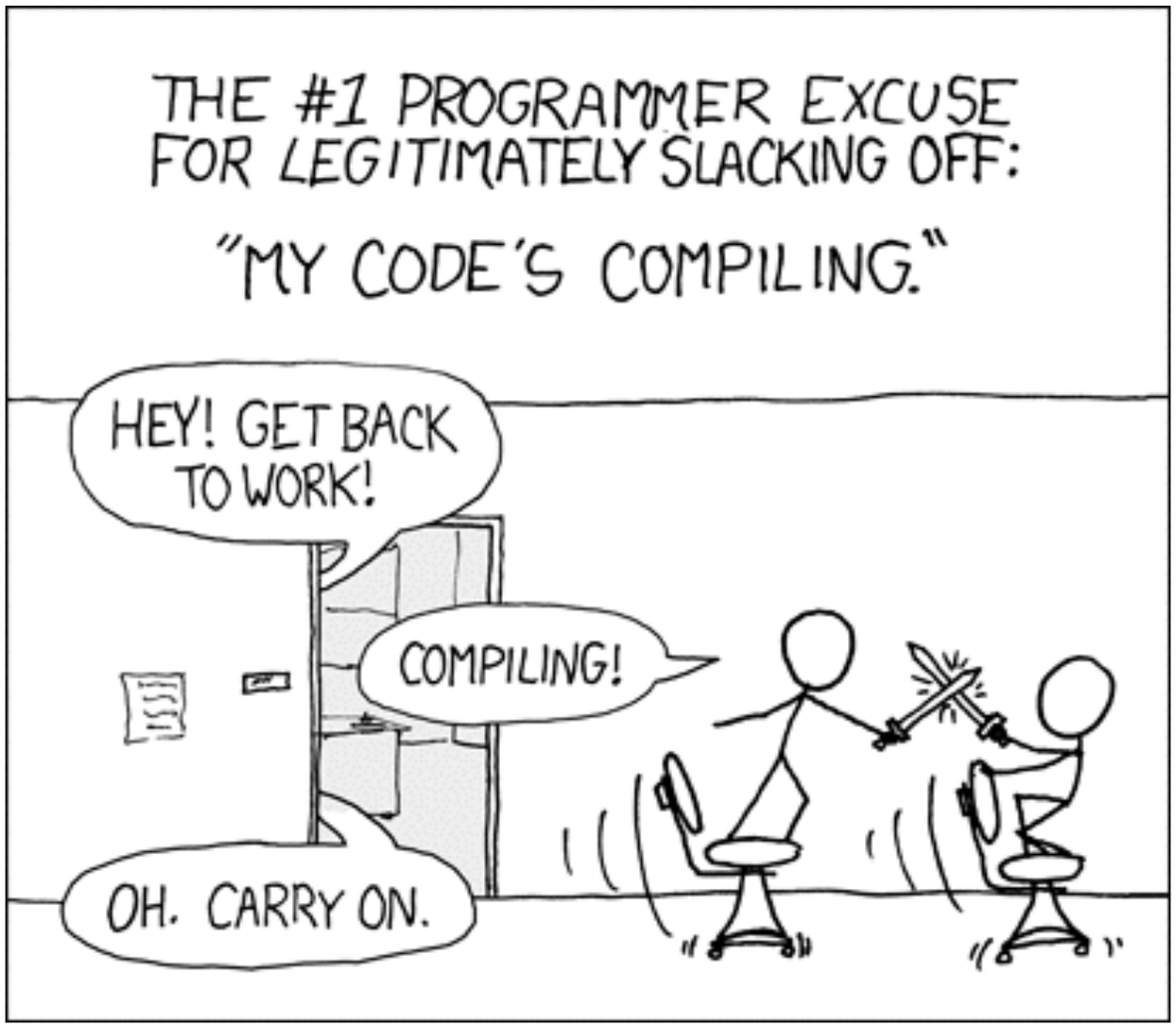
Related Upstreaming

- Improvements to external projects:
  - QEMU: FreeBSD MIPS64 usermode
    - MIPS64 and ARM packages!
  - Clang/LLVM: MIPS64 fixes
  - LLDB: FreeBSD improvements, MIPS64
  - Tcpdump: better compartmentalization interfaces

Releases

- Internal snapshots
- Restricted releases
- Public releases: http://cheri-cpu.org/
- Shared make-based build infrastructure

Tips for developers

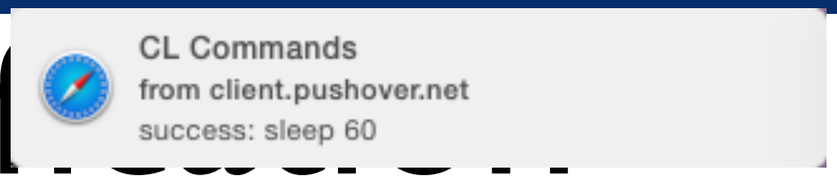


http://xkcd.com/303/ (CC BY-NC 2.5)

Tip 1: Use a big machine

- Enough RAM to hold source and output in cache
  - 128GB is enough for most people
- Fast disk
  - ZFS mirror with large L2ARC and ZIL on flash
- Enough cores
  - 32 on our system

Tip 2: Use a notification service



- I use pushover.net for notifications
- Simple RESTful interface
  - Notifications to iOS and Android devices
  - Also via browser
- Used with a command wrapper script

```
$ command-notice sleep 60
```

Tip 3: Build in tmux

- Switch away from running build
- Sending, buffering, and rendering output just to throw it away wasteful
- Even locally, buffering adds delay between end of compilation and control of the terminal

Tip 4: Continuous integration

- Full OS builds after each change or compiler update (out of tree compiler)
  - CHERI, MIPS64, and AMD64
- Daily release builds
  - Release kernels booted on hardware and in simulation
- Additional Jenkins jobs build release branches daily

Papers and reports

CHERI: A Hybrid Capability-System Architecture for Scalable Software Compartmentalization. Robert N. M. Watson, Jonathan Woodruff, Peter G. Neumann, Simon W. Moore, Jonathan Anderson, David Chisnall, Nirav Dave, Brooks Davis, Khilan Gudka, Ben Laurie, Steven J. Murdoch, Robert Norton, Michael Roe, Stacey Son, and Munraj Vadera. IEEE Security and Privacy 2015.

Beyond the PDP-11: Processor support for a memory-safe C abstract machine. David Chisnall, Colin Rothwell, Brooks Davis, Robert N.M. Watson, Jonathan Woodruff, Simon W. Moore, Peter G. Neumann and Michael Roe. ASPLOS 2015.

The CHERI capability model: Revisiting RISC in an age of risk. Jonathan Woodruff, Robert N. M. Watson, David Chisnall, Simon W. Moore, Jonathan Anderson, Brooks Davis, Ben Laurie, Peter G. Neumann, Robert Norton, and Michael Roe. ISCA 2014.

Capability Hardware Enhanced RISC Instructions: CHERI Instruction-Set Architecture. Robert N.M. Watson, Peter G. Neumann, Jonathan Woodruff, Jonathan Anderson, David Chisnall, Brooks Davis, Ben Laurie, Simon W. Moore, Steven J. Murdoch, and Michael Roe. UCAM-CL-TR-864, Cambridge, December 2014.

Future work

- Pure-capability FreeBSD
  - Run legacy MIPS64 code in sandboxes
  - CHERI in the kernel
  - 128-bit capabilities
  - Non-MIPS architectures

Q & A

October 2011: Deimos capability microkernel runs first sandbox on FPGA prototype

June 2012: CheriBSD capability context for user threads

July 2012: LLVM generates CHERI ISA code

November 2012: Sandboxed user code runs on CheriBSD; trojan mitigated

February 2013: SSHed into a CHERI system for the first time

December 2013: OS CCall/CReturn prototype supports lightweight compartmentalization

January 2014: CheriBSD now uses CHERI Clang/LLVM

November 2014: compartmentalized tcpdump performs multiple domain switches per packet

2010 2011 2012 2013 2014 2015

October 2010: Project start

November 2011: tPad FPGA tablet microkernel demo

May 2012: DE4 FPGA tablet runs FreeBSD

April 2013: multi-FPGA CheriCloud online for SRI/Cam users

July 2014: Multicore bringup in progress; capability-offset ('cursor') prototype; advanced application prototyping

May 2015: 128-bit compressed-capability prototype boots