CHERI – Architectural memory safety and compartmentalisation

Full steam ahead to commercialization

Robert N. M. Watson, Simon W. Moore, Peter G. Neumann, **Jonathan Woodruff**
Hesham Almatary, Ricardo de Oliveira Almeida, Jonathan Anderson, Alasdair Armstrong, Rosie Baish, Peter Blandford-Baker, John Baldwin, Hadrien Barrel, Thomas Bauereiss, Ruslan Bukin, Brian Campbell, David Chisnall, Jessica Clarke, Nirav Dave, Brooks Davis, Lawrence Esswood, Nathaniel W. Filardo, Franz Fuchs, Dapeng Gao, Ivan Gomes-Ribeiro, Khilan Gudka, Brett Gutstein, Angus Hammond, Graeme Jenkinson, Alexandre Joannou, Mark Johnston, Robert Kovacsics, Ben Laurie, Marno van der Maas, A. Theo Markettos, J. Edward Maste, Alfredo Mazzinghi, Alan Mujumdar, Prashanth Mundkur, Steven J. Murdoch, Edward Napierala, George Neville-Neil, Kyndylan Nienhuis, Robert Norton-Wright, Philip Paeps, Lucian Paul-Trifu, Allison Randal, Alex Richardson, Michael Roe, Colin Rothwell, Peter Rugg, Hassen Saidi, Peter Sewell, Thomas Sewell, Stacey Son, Ian Stark, Domagoj Stolfa, Andrew Turner, Munraj Vadera, Konrad Witaszczyk, Hongyan Xia, Vadim Zaliva, and Bjoern A. Zeeb

University of Cambridge and SRI International

RISC-V Tokyo Summit
February 27, 2025

UK Research and Innovation

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.

UNIVERSITY OF CAMBRIDGE

Approved for public release; distribution is unlimited.

This work was supported in part by the Innovate UK project Digital Security by Design (DSbD) Technology Platform Prototype, 105694.

This work was also supported by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL), under contract FA8750-10-C-0237 ("CTSRD"), with additional support from FA8750-11-C-0249 ("MRC2"), HR0011-18-C-0016 ("ECATS"), FA8650-18-C-7809 ("CIFV"), and HR001122C0110 ("ETC") as part of the DARPA CRASH, MRC, and SSITH research programs. The views, opinions, and/or findings contained in this report are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

We further acknowledge the EPSRC REMS Programme Grant (EP/K008528/1), the ERC ELVER Advanced Grant (789108), the Isaac Newton Trust, the UK Higher Education Innovation Fund (HEIF), Thales E-Security, Microsoft Research Cambridge, Arm Limited, Google, Google DeepMind, HP Enterprise, and the Gates Cambridge Trust.

Overview

- The CHERI architecture – Why and what
- The MIPS phase – Tags & compression
- The Arm Morello phase – Revocation & performance
- The RISC-V present – Simplification & implementation

An early experimental FPGA-based CHERI tablet prototype running the CheriBSD operating system and applications, Cambridge, 2013

Disclaimer: Real-life phases overlap and are technical contributions are more complicated. Liberties have been taken with talk structure to make the story better.

THE CHERI ARCHITECTURE

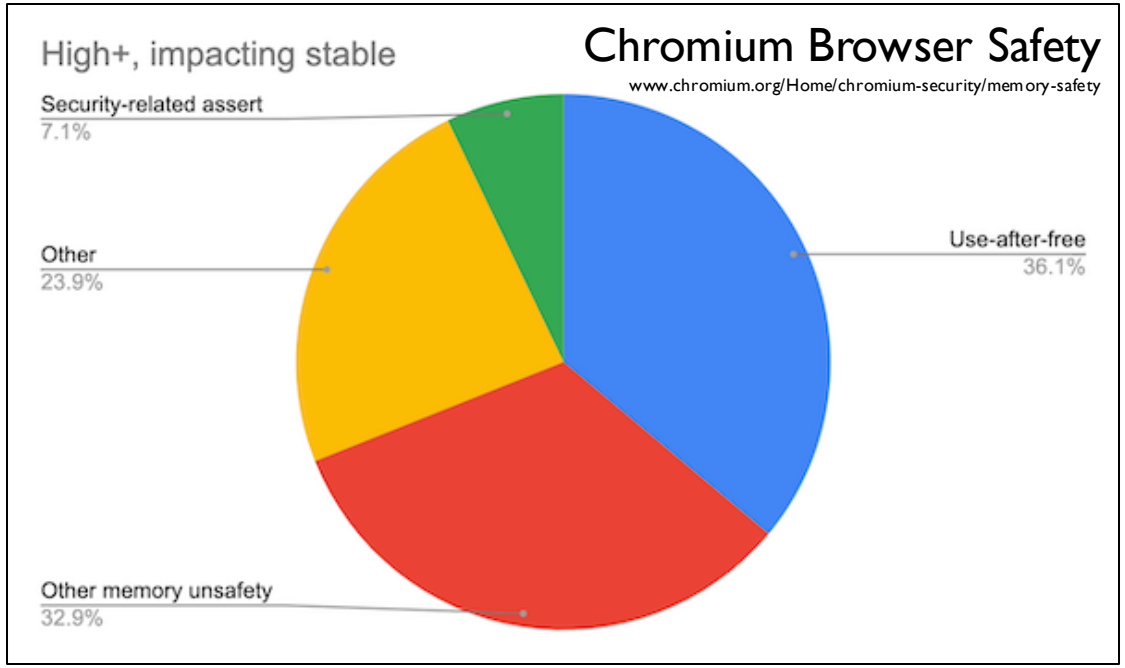
Motivation – The memory safety crisis

"Buffer overflows have not objectively gone down in the last 40 years.

The impact of buffer overflows have if anything gone up."

Ian Levy, Technical Director at NCSC

- Matt Miller (MS Response Center) @ BlueHat 2019:
  - From 2006 to 2018, year after year, 70% MSFT CVEs are memory safety bugs.
  - First place: spatial safety "Out-of-bounds"
    - Fixed at native speed by CHERI
  - Second place: temporal safety "Use-after-free"
    - High-performance fix enabled by CHERI
  - Other?
    - CHERI compartmentalisation can limit fallout



SRI International

UNIVERSITY OF CAMBRIDGE

CHERI: Unforgeable Bounded Pointers

Called "Capabilities"
Why?
Ask Computer Science history…

It's all in the title.

- "**Unforgeable**"

  The architecture remembers whether a word is a pointer. This allows the flow of pointers to be used for bona-fide privilege delegation.
  This requires **tagged memory**.

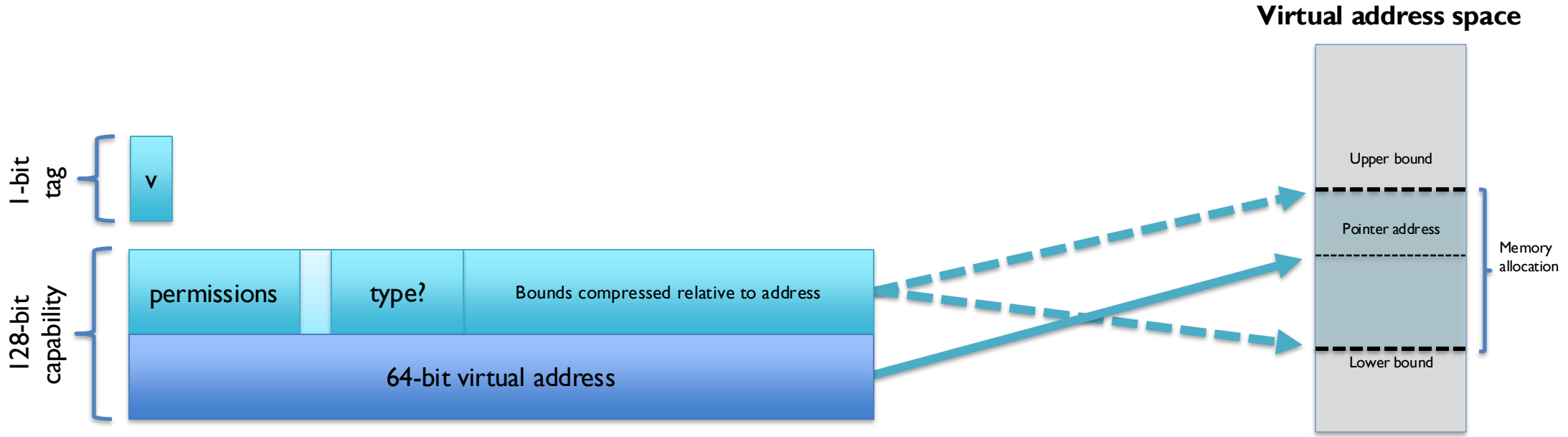
- "**Bounded**"

  Bounds and permissions metadata is added to pointers which cannot be expanded.
  Pointer delegation can thus constrain a program to a subset of the address space.
  This requires **wider pointers**.

- "**Pointers**"

  CHERI capability pointers are used wherever pointers are used in C and C++ programs. This enables wide, immediate & natural adoption.

CHERI Architecture: Bounded Pointers



- Capability pointers are double the width of a virtual address
- Bounds and permissions metadata are carried atomically with all pointers and asserted on memory access
- New instructions are added to manipulate metadata (e.g. SetBounds)

CHERI Software: CHERI LLVM

- CHERI Clang/LLVM compiles C and C++ code to pure-capability executables.
- All memory operations explicitly use CHERI capability pointers.
- This communicates language-level memory information in the architecture to be enforced at run-time.

```
struct timezone tz;


time_t get_unix_time(void) {
        struct timeval tv;

        gettimeofday(&tv, &tz);
        return tv.tv_sec;
}
```

```
get_unix_time_riscv:
 addi       sp, sp, -32
 sd         ra, 24(sp)
 addi       a0, sp, 8
 auipc      a1, %pcrel_hi(tz)
 addi       a1, a1, %pcrel_lo(tz)
 auipc      a2, %pcrel_hi(gettimeofday)
 jalr       a2, %pcrel_lo(gettimeofday)

 ld         a0, 8(sp)
 ld         ra, 24(sp)
 addi       sp, sp, 32
 ret
```

```
get_unix_time_cheririscv:
 caddi      csp, csp, -32
 sc         cra, 16(csp)
 scbndi     ca0, csp, 16
 auipc      ca1, %pcrel_hi(tz)
 lc         ca1, %pcrel_lo(tz)(ca1)
 auipc      ca2, %pcrel_hi(gettimeofday)
 clc        ca2, %pcrel_lo(gettimeofday)(ca2)
 jalr       cra, ca2
 ld         a0, 0(csp)
 lc         cra, 16(csp)
 caddi      csp, csp, 32
 ret
```

1. Adjust stack address/capability
2. Save return address/capability
3. Create address/capability to local 'tv'

4. Generate address/capability to global 'tz'

5. Call gettimeofday()

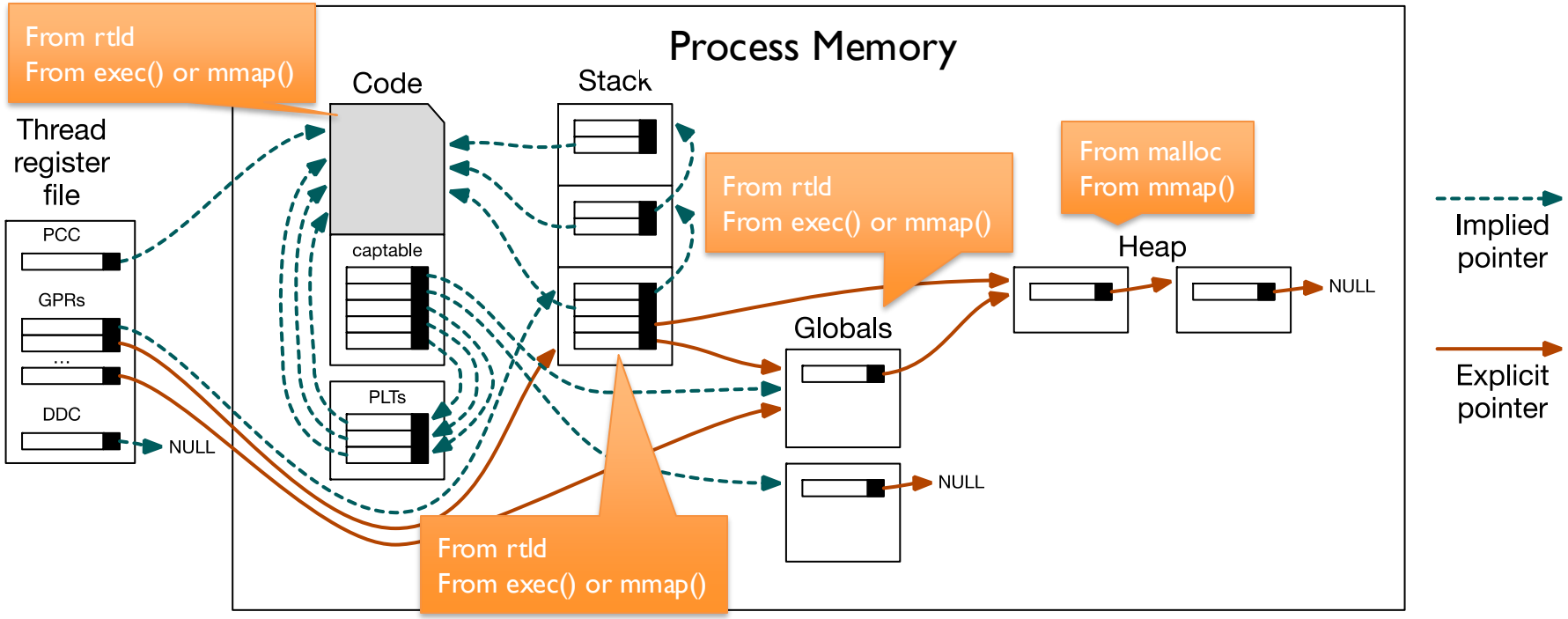
6. Load return value from 'tv'
7. Load return address/capability
8. Restore stack address/capability
9. Return

CHERI Software: CheriBSD

- Complete memory- and pointer-safe FreeBSD C/C++ kernel + userspace

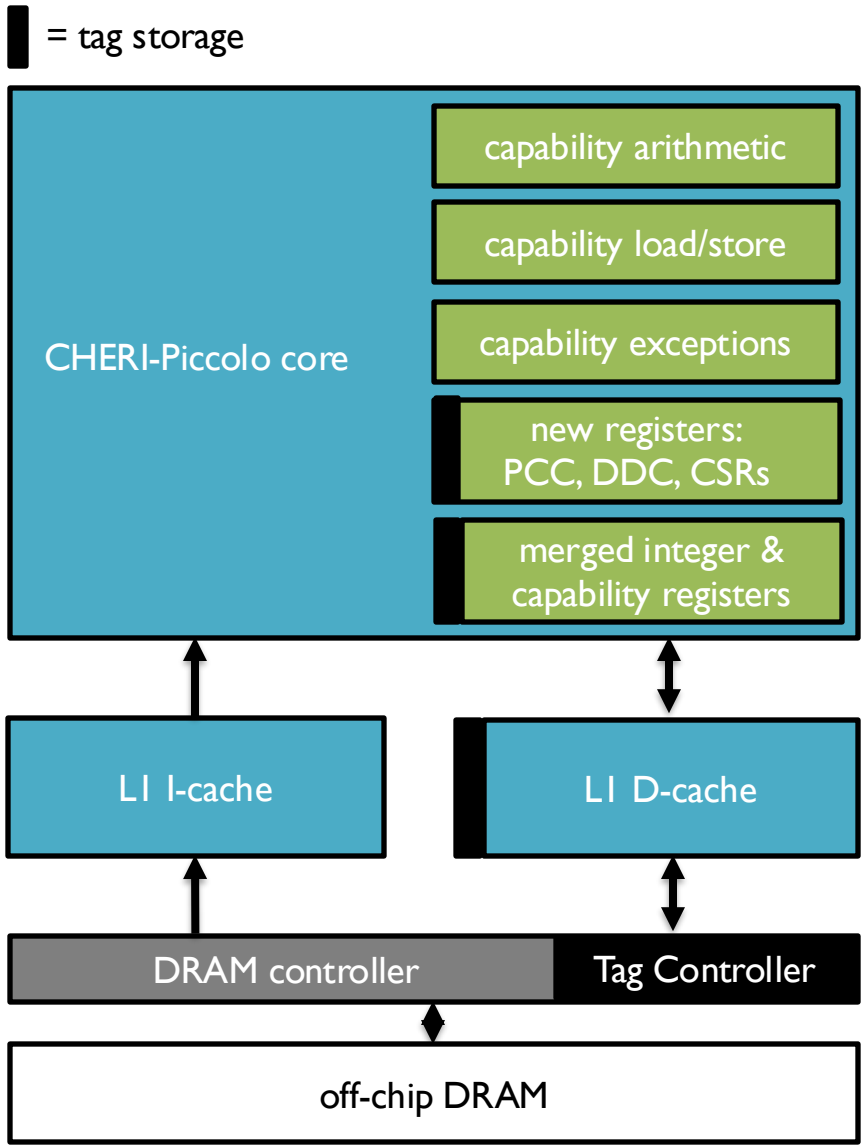
  Core OS kernel, filesystems, networking, device drivers, system libraries (ld-elf.so, libc, zlib), system tools and daemons (echo, sh, openssl), applications (PostgreSQL, nginx, WebKit)

- **Valid provenance**, **minimized privilege** for **pointers, implied VAs**
  - Userspace capabilities originate in **kernel-provided roots** via mmap(), etc.
  - Compiler, allocators, run-time linker, etc., **refine** bounds and perms



Example: CHERI-Piccolo microcontroller

CHERI Hardware



Changes to the core:

- capability arithmetic
- capability load/store operations with bounds checking
- extended exception model
- PC becomes a capability (PCC)
- default data capability (DDC)
- new control/status registers
- extended capability register file

Memory subsystem:

- AXI user-field added to transport tag bits & data width doubled
- caches extended to include tags

DRAM changes:

- New tag controller emulates tagged memory by maintaining a table of tag bits in the top of DRAM

SRI International

UNIVERSITY OF CAMBRIDGE

CHERI – THE MIPS ERA

CHERI-MIPS – Initial Compiler Support

Initially used explicit capability annotations.
Now we use **pure-capability ABI**.

```
__capability char *myString =
  (__capability char*)malloc(size);



myString[size] = 'd';
```

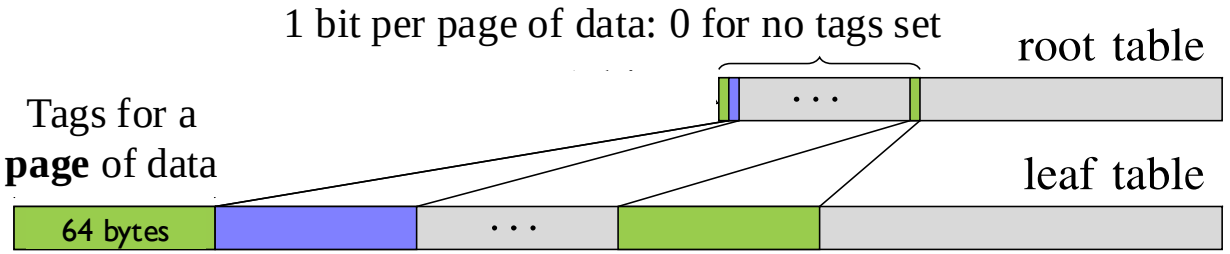
```
jalr        malloc

cincbase    $c1, $c0, returnValue
csetlen     $c1, $c1, size

csb         'd', size, 0($c1)
```

CHERI-MIPS – Tagged memory

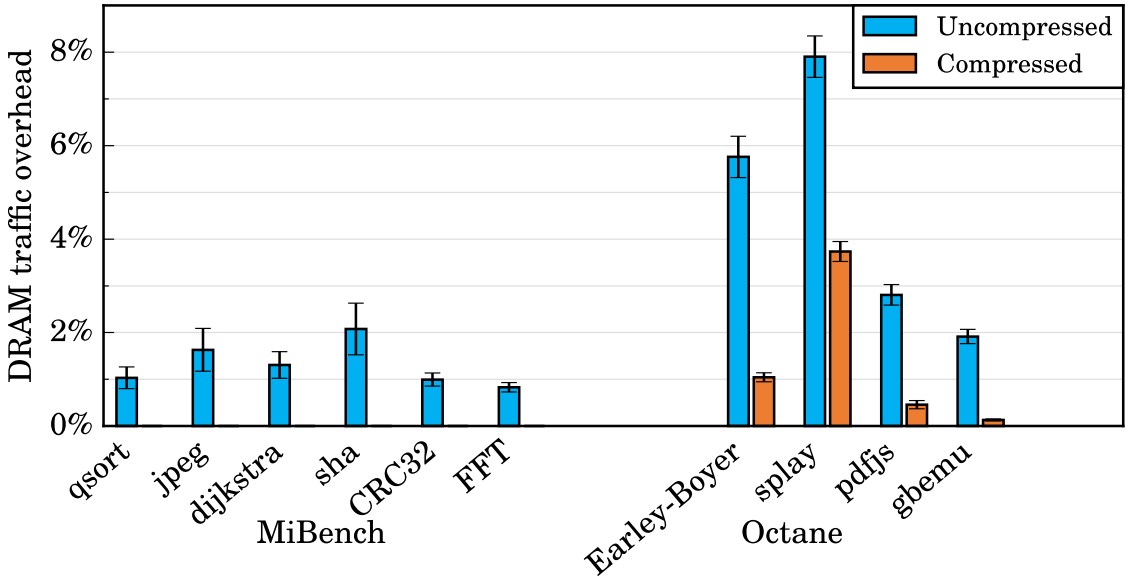
Tag Compression



- 2-level tag table
- Each bit in the **root** level indicates all zeros in a **leaf** group
- Reduces tag cache footprint
- Amplifies cache capacity

DRAM Traffic Overhead

Note: MiBench overheads with compression are approximately zero

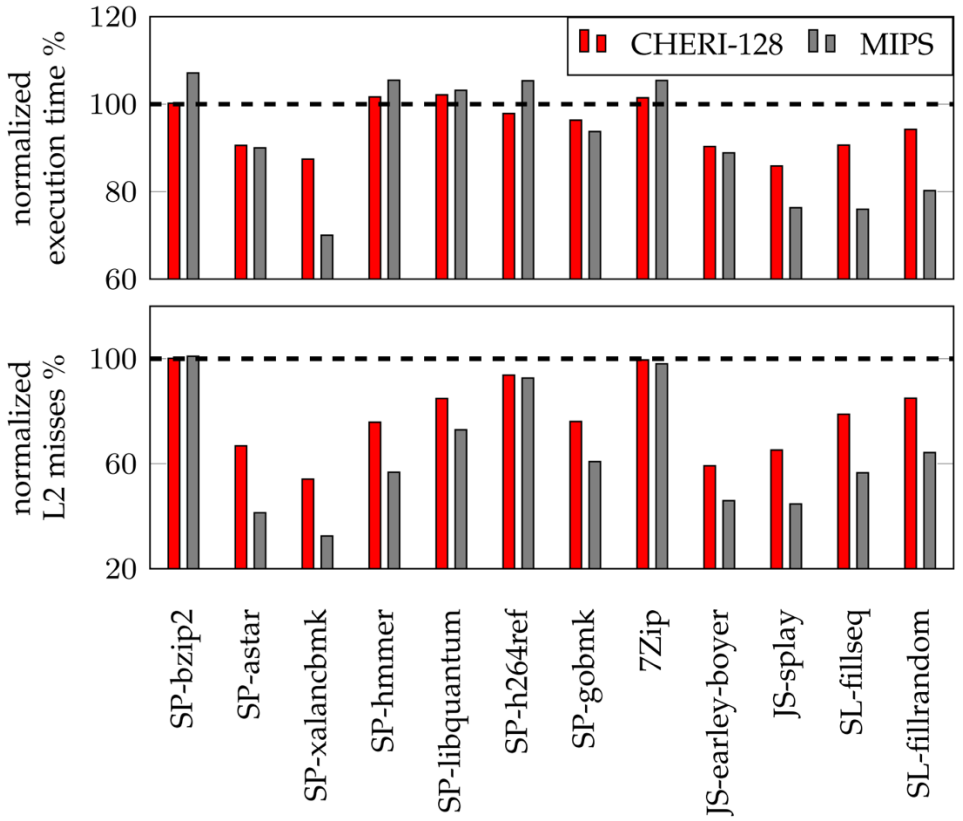


CHERI-MIPS – Capability Compression

Properties:

- Larger allocations have greater alignment requirements.
  <= 4096 byte-accurate for CHERI-128 and <= 512 for CHERI-64
- Granularity is the only alignment requirement.
  Allocations can span any alignment boundary.
- Metadata bits do not change with pointer arithmetic.
  Common case is simpler.
  Representability check is faster than full bounds check.

Performance vs. CHERI-256 baseline for Spec2006 (SP) and JavaScript (JS) benchmarks.

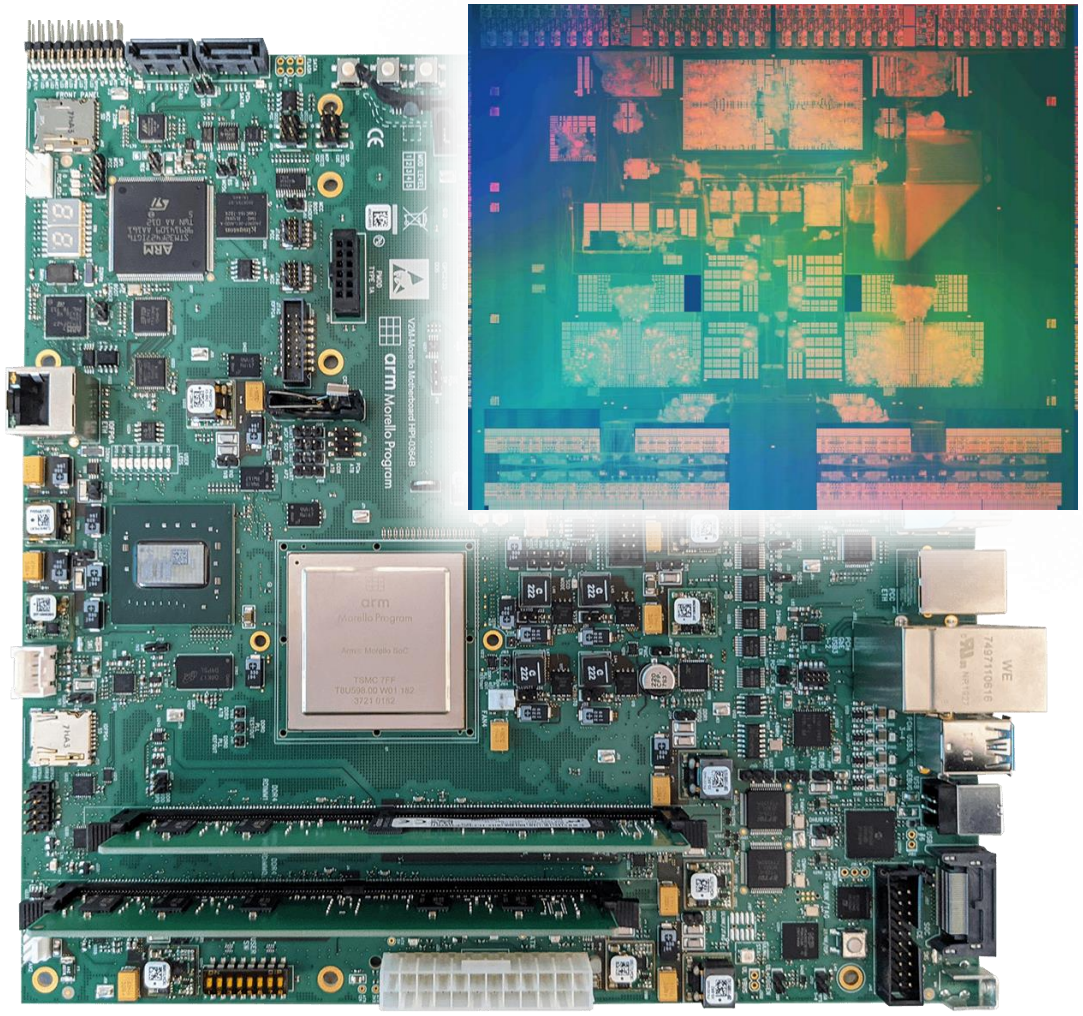


SRI International

UNIVERSITY OF CAMBRIDGE

CHERI – THE ARM MORELLO ERA

Arm Morello - Overview



- $225M government, academia, and industrial research program led by UK Research and Innovation (UKRI)
  - Announced partners: Arm, Google, Microsoft
  - 15+ UK universities with research grants
  - 70+ funded business incubation projects
- Baseline for design: Neoverse N1 core
  - 2.5GHz quad-core, superscalar
  - Implements CHERI extensions
  - Runs full CHERI-enabled software stacks
  - A prototype, but a very powerful one!
- Roughly a thousand chips manufactured for use by research + development labs

Arm Morello – Temporal Safety (1)

- Insight: CHERI enables temporal safety for C by:
  - Allowing pointers to be identified in memory
  - Preventing capability pointers from being fabricated or enlarged
  - Enabling references to be reliably revoked
- General strategy:
  1. Put freed memory in quarantine
  2. Mark memory free
  3. Sweep memory and delete capabilities that point to freed memory

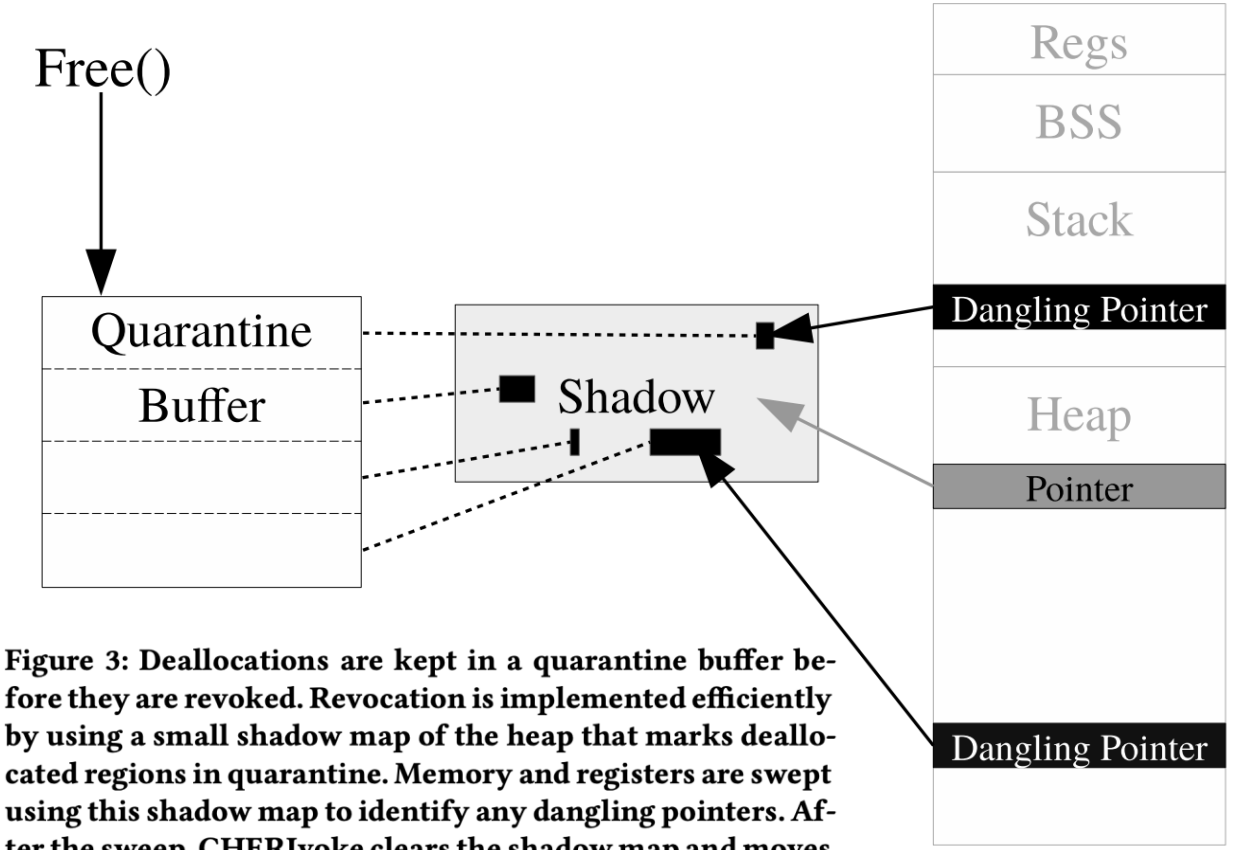
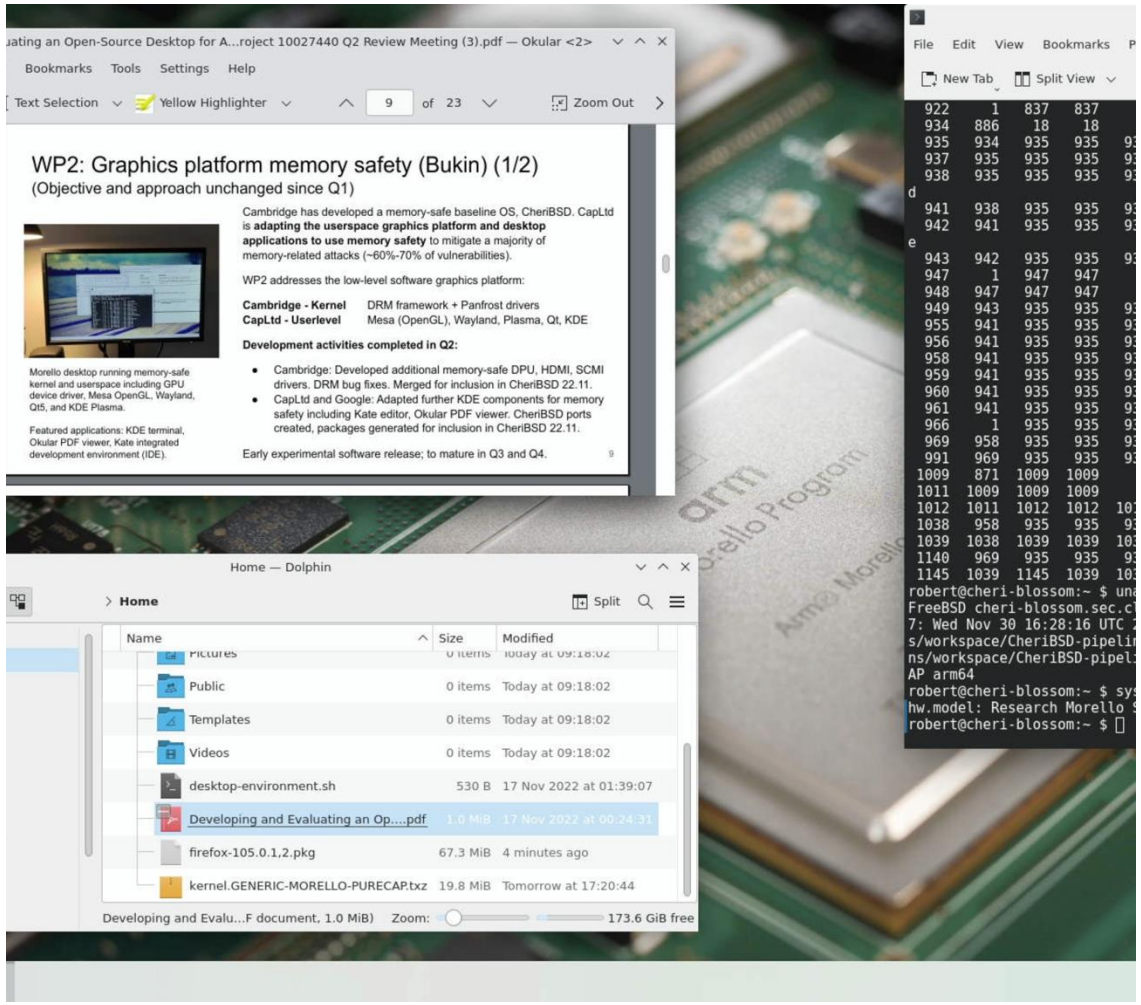


Figure 3: Deallocations are kept in a quarantine buffer before they are revoked. Revocation is implemented efficiently by using a small shadow map of the heap that marks deallocated regions in quarantine. Memory and registers are swept using this shadow map to identify any dangling pointers. After the sweep, CHERIvoke clears the shadow map and moves quarantined locations into the free list for reallocation.

Arm Morello – CheriBSD Full Desktop



Roughly 30MLoC on a shipping Arm Morello board today, with memory-safe:

- CheriBSD kernel with DRM + Panfrost drivers
- CheriBSD userspace with libraries, OpenSSH, ...
- OpenGL, Wayland display server
- Plasma, KDE base applications including Dolphin, Okular, Konsole.
- Aarch64 CHERI/Morello-aware GDB debugger
- 9K CheriABI packages, 20K aarch64 ("legacy") packages; notable exclusions for language runtimes
- Temporal safety by default
- Library compartmentalisation

CHERI – THE RISC-V ERA

CHERI-RISC-V – Simplifications

Sealing

"Otype" Sealing
- Dedicated bits for type (3-18)
+ Any capability can be sealed with any type
- Limited number of types
- Extra instructions/permissions to control type delegation
- Need a type manager to allocate

Subset Sealing
- "Unseal" by rebuilding from a superset capability
  Reconstruct with BuildCap
  Probably add a dedicated instruction
+ Type space scales with virtual memory
+ Use single "sealed" bit
- Objects of a type must be in the same contiguous memory (Can virtualize with indirection)

Still validating that Subset sealing is sufficient.

CHERI-RISC-V – Simplifications

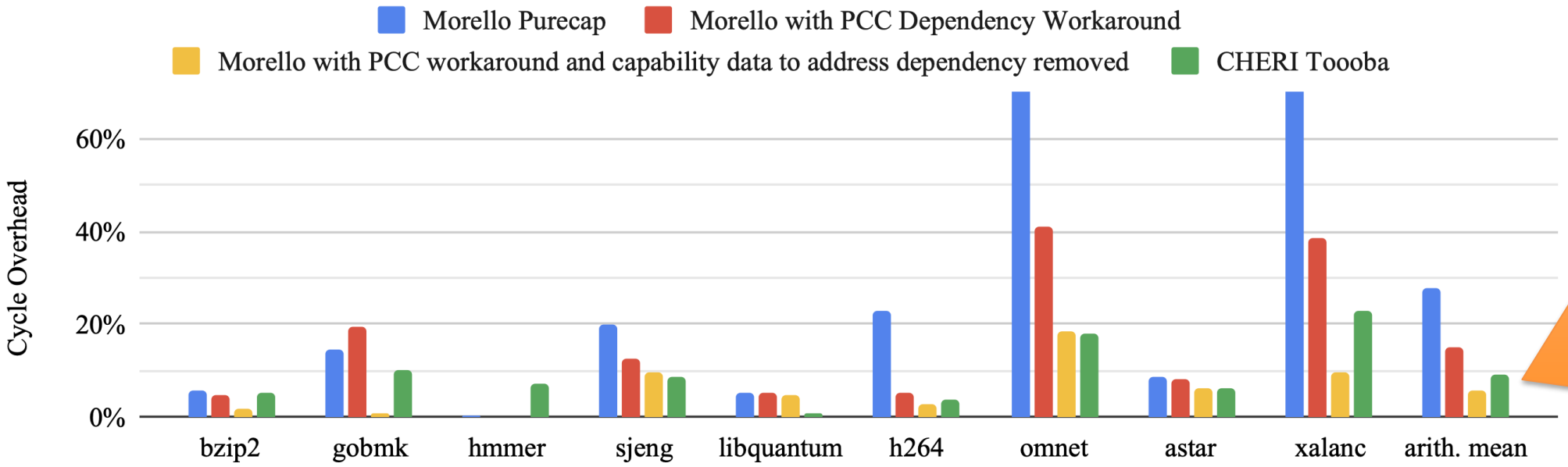
PTE Bits

- Morello had 4
- CHERI-RISC-V prototype had 5
- Capability Read
- Capability Write
- Capability Read Generation
- Capability Dirty

Trying to reduce to **2 bits** (4 states)

1. Cap writeable, read gen 0
2. Cap writeable, read gen 1
3. Trap on capability write,
   Strip tag on cap read
4. Auto "dirtyable" to cap writeable,
   read gen=current

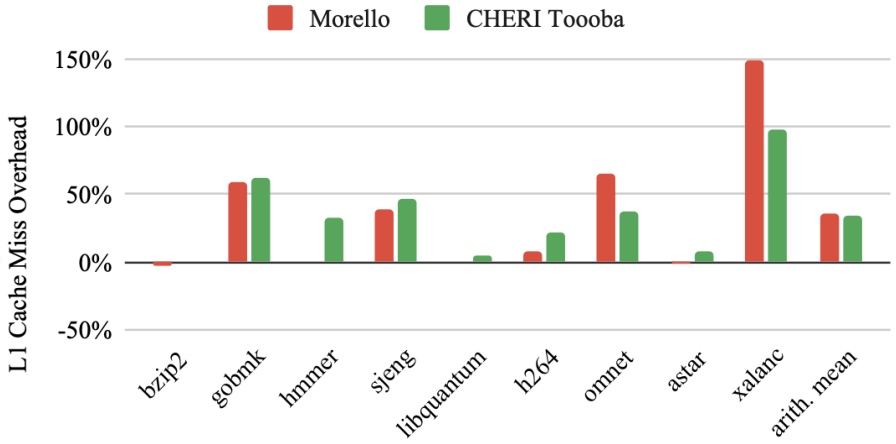
CHERI-RISC-V – Performance



Predicted average overhead of ~5%; current best average overhead of ~6% on CHERI Toooba. Still improving code generation; should improve further with ratified CHERI-RISC-V and Codasip compiler work.

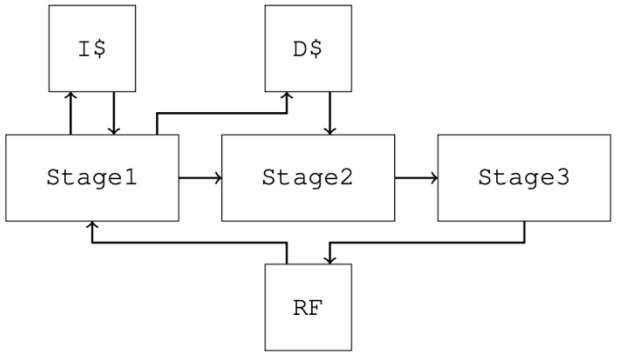
Fig. 13: Cycle overhead for variations of the Arm Morello design and for CHERI-Toooba on SPECint2006, train workload

Fundamental overhead is cache pressure. This is similar across Arm's Morello and RISC-V Toooba; this gives confidence that Toooba is an appropriate platform for further research.

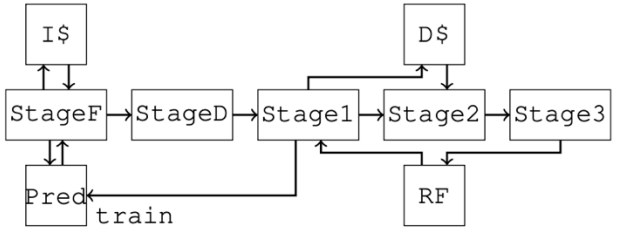


CHERI-RISC-V – Implementations
Cores from Cambridge in Bluespec

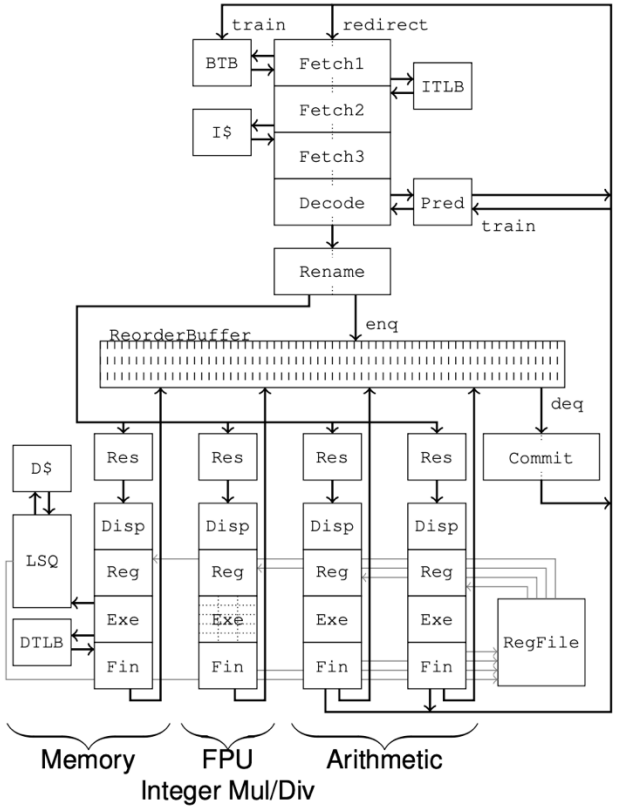
	Piccolo	Flute	Tooba
Frequency	50 MHz	100 MHz	25 MHz
`xlen`	32	64	64
Extensions	C,M	A,C,D,F,M	A,C,D,F,M
Priv. modes	M,U	M, S, U	M, S, U
Superscalar	1	1	2
Cores	1	1	2
TLB	—	16 entries	32 entries
Pipe stages	3	5	11
RAS	—	16 entries	16 entries
BTB	—	512 entries	1024 entries
Data cache	4 KiB 2way	8 KiB 2way	32 KiB 8way
Inst. cache	4 KiB 2way	8 KiB 2way	32 KiB 8way
L2 cache	—	—	1 MiB 16way
Tag cache	4 KiB 4way	4 KiB 4way	128 KiB 4way



(1a) Piccolo



(1b) Flute



(1c) Tooba

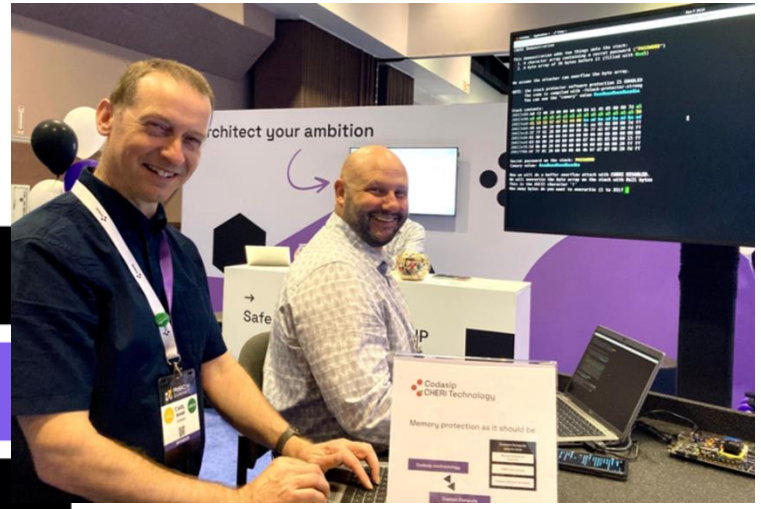
CHERI-RISC-V – Implementations
CHERIoT

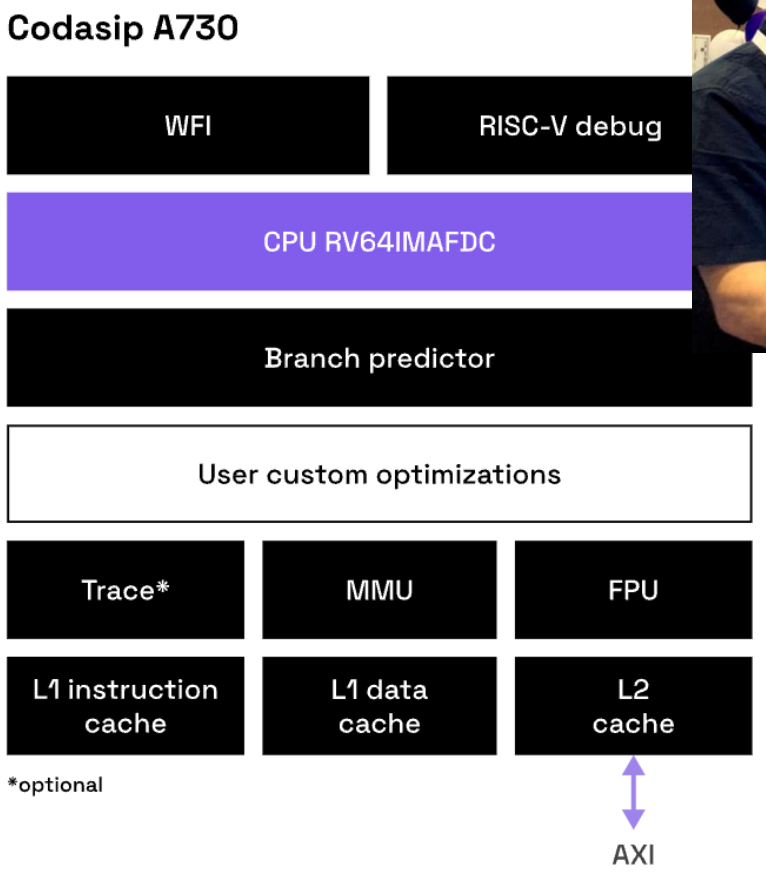
- Open source
- Developed by Microsoft
- Rv32 + custom CHERI ISA fork
- Includes embedded operating system with sophisticated hardware/software guarantees
- Spatial safety
- Hardware-enabled strict temporal safety
- Compartmentalisation without associative tables
- Several paths to commercialisation

CHERI-RISC-V – Implementations

Codasp 700 series

- Rv64, full rva22 core with pre-ratification CHERI extension
- Dual-issue, in-order
- Large team ensuring mature software support at launch





Conclusions

- CHERI is about to be deployed in specialized applications.
- Software ecosystem is well-developed and maturing fast.
- If CHERI accelerates general-purpose computing, it will move into the mainstream.
- Questions?

Low-cost compartments?
Microarchitectural knowledge?

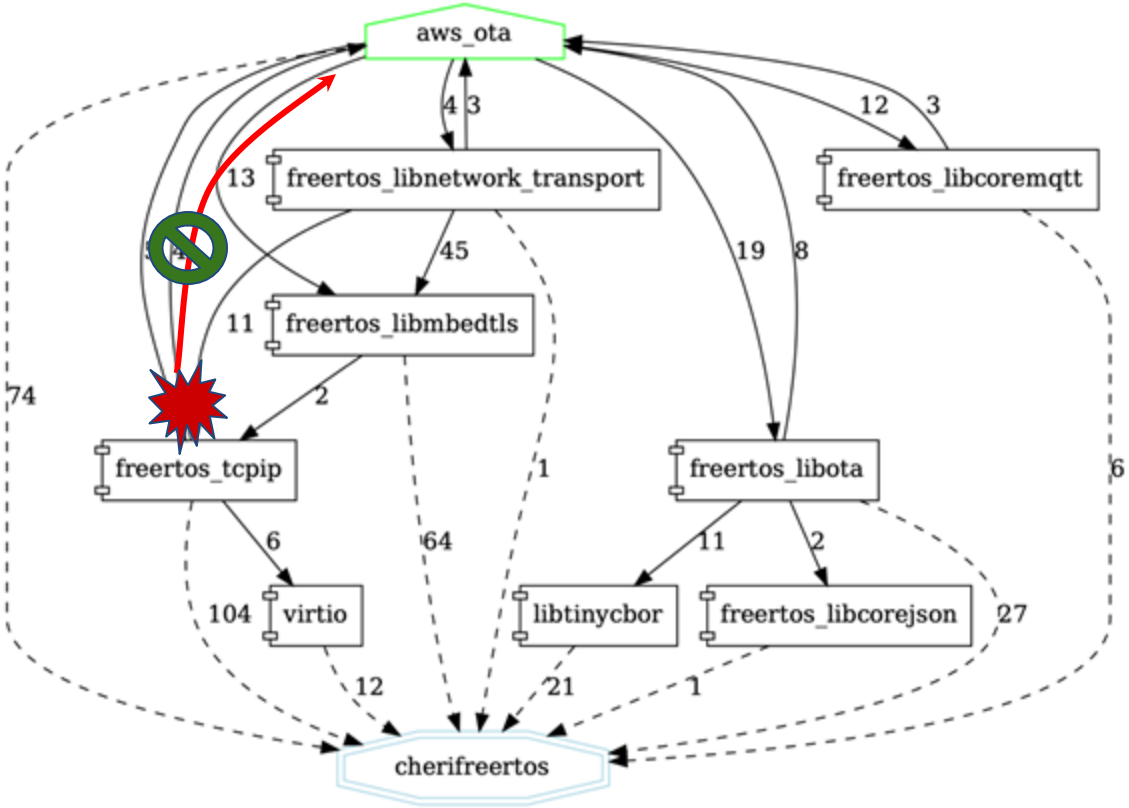
https://www.cl.cam.ac.uk/research/security/ctsrd/

Thanks to sponsors: DARPA, ARM, Google, EPSRC, ESRC, HEIF, Isaac Newton Trust, Thales E-Security, HP Labs

Jonathan.Woodruff@cl.cam.ac.uk
Dept. Computer Science & Technology

BACKUP SLIDES

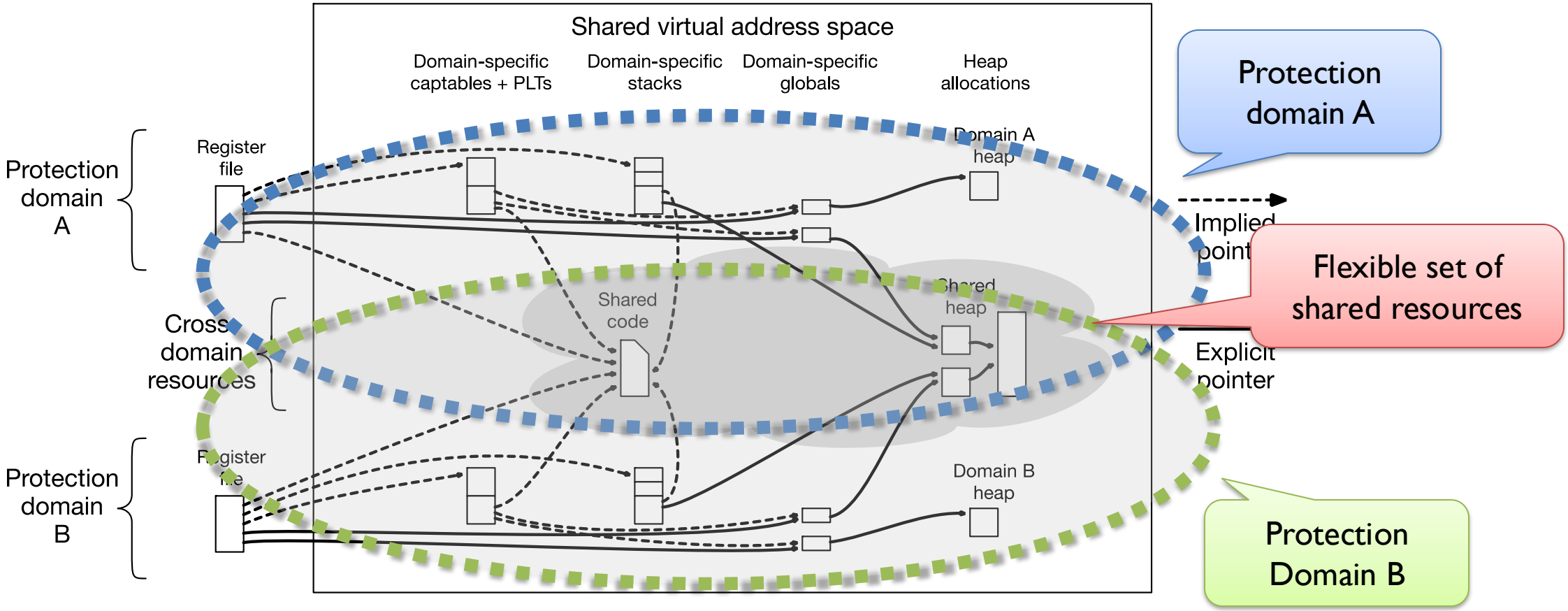
What is software compartmentalization?



CheriFreeRTOS components and the application execute in compartments. CHERI contains an attack within TCP/IP compartment, which access neither flash nor the internals of the software update (OTA) compartment.

- Fine-grained decomposition of a larger software system into **isolated modules** to constrain the impact of faults or attacks
- Goals is to **minimize privileges yielded by a successful attack, and to limit further attack surfaces**
- Usefully thought about as a **graph of interconnected components**, where the attacker's goal is to compromise nodes of the graph providing a route from a point of entry to a specific target

CHERI-based compartmentalization



- Isolated compartments can be created using closed graphs of capabilities, combined with a constrained non-monotonic domain-transition mechanism