Jacob Bramley
Dejice Jacob
Andrei Lascu
Jeremy Singer
Laurence Tratt

Picking a CHERI: Security and Performance Considerations

16/06/23

ISMM'23

Motivation

Rank	ID	Name	Score	KEV Count (CVEs)	Rank Change vs. 2021
1 ★	CWE-787	Out-of-bounds Write	64.20	62	0
2	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	45.97	2	0
3	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	22.11	7	+3 ▲
4	CWE-20	Improper Input Validation	20.63	20	0
5 ★	CWE-125	Out-of-bounds Read	17.67	1	-2 ▼
6	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	17.53	32	-1 ▼
7 ★	CWE-416	Use After Free	15.50	28	0
8	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	14.08	19	0
9	CWE-352	Cross-Site Request Forgery (CSRF)	11.53	1	0
10	CWE-434	Unrestricted Upload of File with Dangerous Type	9.56	6	0
11 ★	CWE-476	NULL Pointer Dereference	7.15	0	+4 ▲

Source: https://cwe.mitre.org/top25/archive/2022/2022_cwe_top25.html

CHERI

CHERI

Capability Hardware Enhanced RISC Instructions

CHERI

Capability Hardware Enhanced RISC Instructions

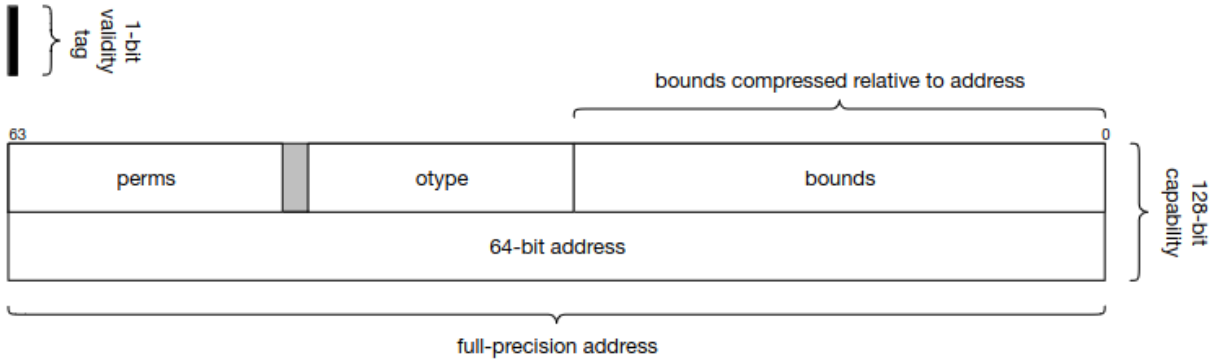


Figure 2.1: 128-bit CHERI Concentrate capability representation: 64-bit address and metadata in addressable memory and 1-bit tag out of band.

Source: An Introduction to CHERI - https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-941.pdf

CHERI

Capability **Hardware Enhanced** RISC Instructions

CHERI

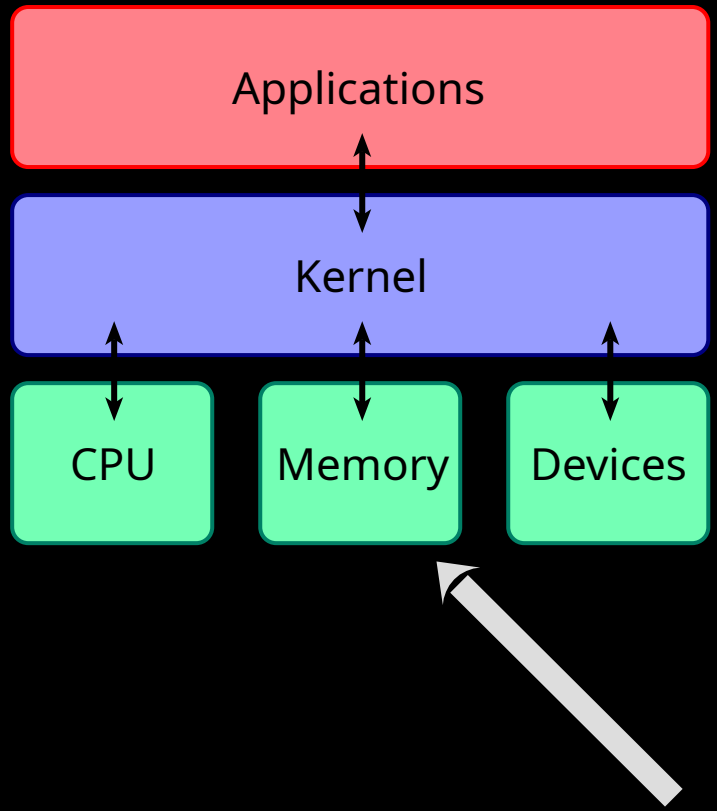


Diagram source: https://en.wikipedia.org/wiki/Kernel_(operating_system)

CHERI allocators

jemalloc	dlmalloc	snmalloc
xalloc	dmalloc	ElectricFence
ptmalloc	libmalloc_simple	

CHERI allocators

jemalloc	dlmalloc	snmalloc
xalloc	dmalloc	ElectricFence
ptmalloc	libmalloc_simple	

Security

Security

```
uint8_t *arr = malloc(256);
for (uint8_t i = 0; i < 256; i++)
    arr[i] = i;
arr = realloc(arr, 1);
arr = realloc(arr, 256);
for (uint8_t i = 0; i < 256; i++)
    assert(arr[i] == i);
```

✓ snmalloc
× libmalloc_simple
× jemalloc

Security

```
uint8_t *arr = malloc(16);
assert(cheri_perms_get(arr) & CHERI_PERM_STORE));
arr = cheri_perms_clear(arr);
assert((cheri_perms_get(arr) & CHERI_PERM_STORE) == 0);
arr = realloc(arr, 16);
assert(cheri_perms_get(arr) & CHERI_PERM_STORE);
```

✓ snmalloc
✓ libmalloc_simple
× jemalloc

Security

```
uint8_t *arr = malloc(16);
assert(cheri_tag_get(arr));
arr = cheri_tag_clear(arr);
assert(!cheri_tag_get(arr));
arr = realloc(arr, 16);
assert(cheri_tag_get(arr));
```

✓ snmalloc
✓ libmalloc_simple
✓ jemalloc

Performance

Performance

Purecap

```
void *
```

Hybrid

```
void __capability__ *
```

Performance - machine

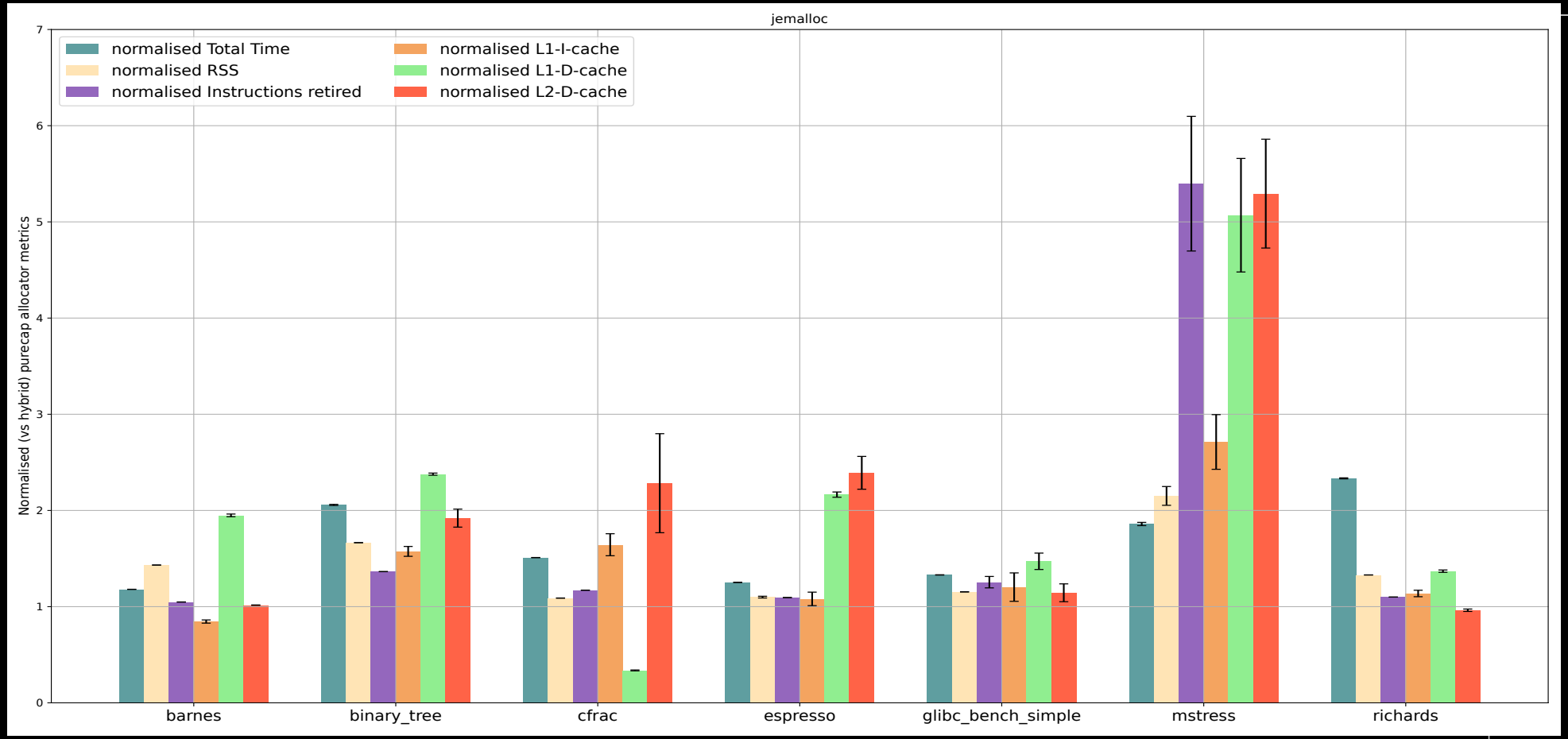
- quad-core Armv8-A 2.5GHz CPU
- 64KiB L1 data cache
- 64KiB L1 instruction cache
- 1MiB L2 unified cache per core
- two 1MiB L3 unified caches (shared between a pair of cores)
- 16GiB DDR4 RAM
- CheriBSD 22.12

Performance - benchmarks

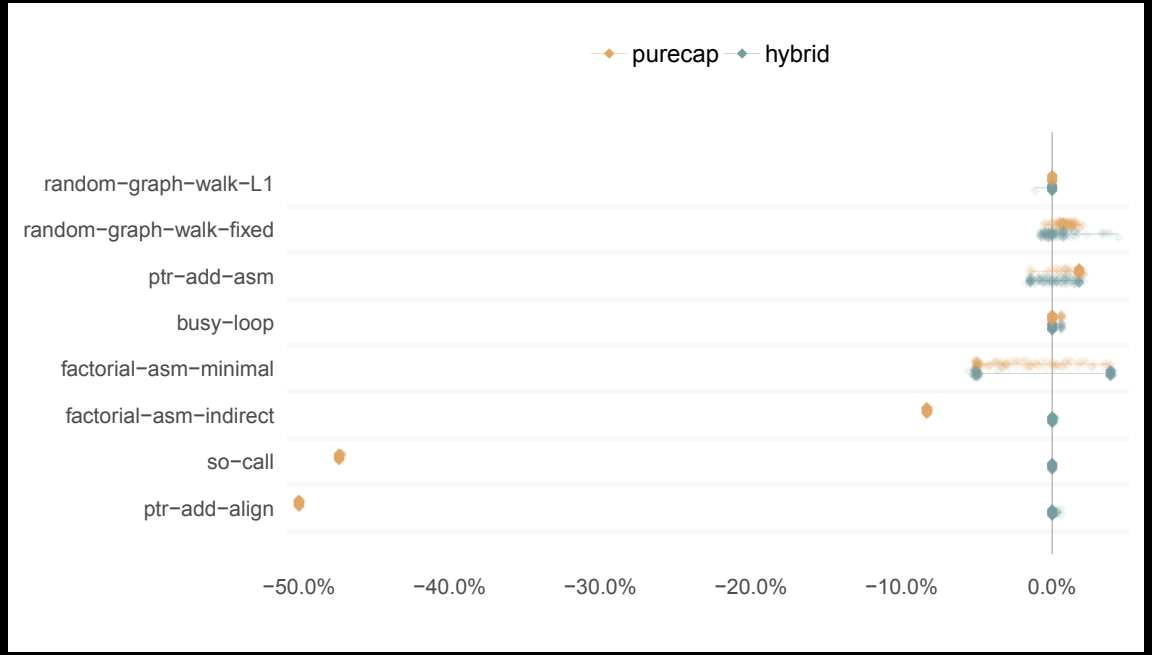
Benchmark	Source	Characterisation
barnes	mimalloc	Floating-point compute
binary-tree	boehm	Alloc & pointer indirection
cfrac	mimalloc	Alloc & int compute
espresso	mimalloc	Alloc & int compute
glibc-simple	mimalloc	Alloc
mstress	mimalloc	Alloc
richards	richards	Pointer indirection

Table 3. Our benchmark suite. We list the benchmark name, the source of the benchmark, and a brief characterisation of it as a workload. *Alloc* is short-hand for 'allocator intensive' (i.e. frequent allocation and deallocation).

Performance - jemalloc



Performance



Conclusions

- Security implementation versus expectations
- Performance tooling and space
- snmalloc as default?

Questions?

https://github.com/capablevms/cheri_misidioms