Measuring, analysing, and understanding performance on Morello

Robert N. M. Watson[*], **Jessica Clarke**[*], Peter Sewell[*], Jonathan Woodruff[*], **Simon W. Moore**[*], Graeme Barnes[**], Richard Grisenthwaite[**], Kathryn Stacer[**], Silviu Baranga[**], Alexander Richardson[†]

[*] University of Cambridge [**] Arm Limited [†] Google LLC

DSbD All Hands Meeting – 8 November 2023

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 contracts 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") . 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.

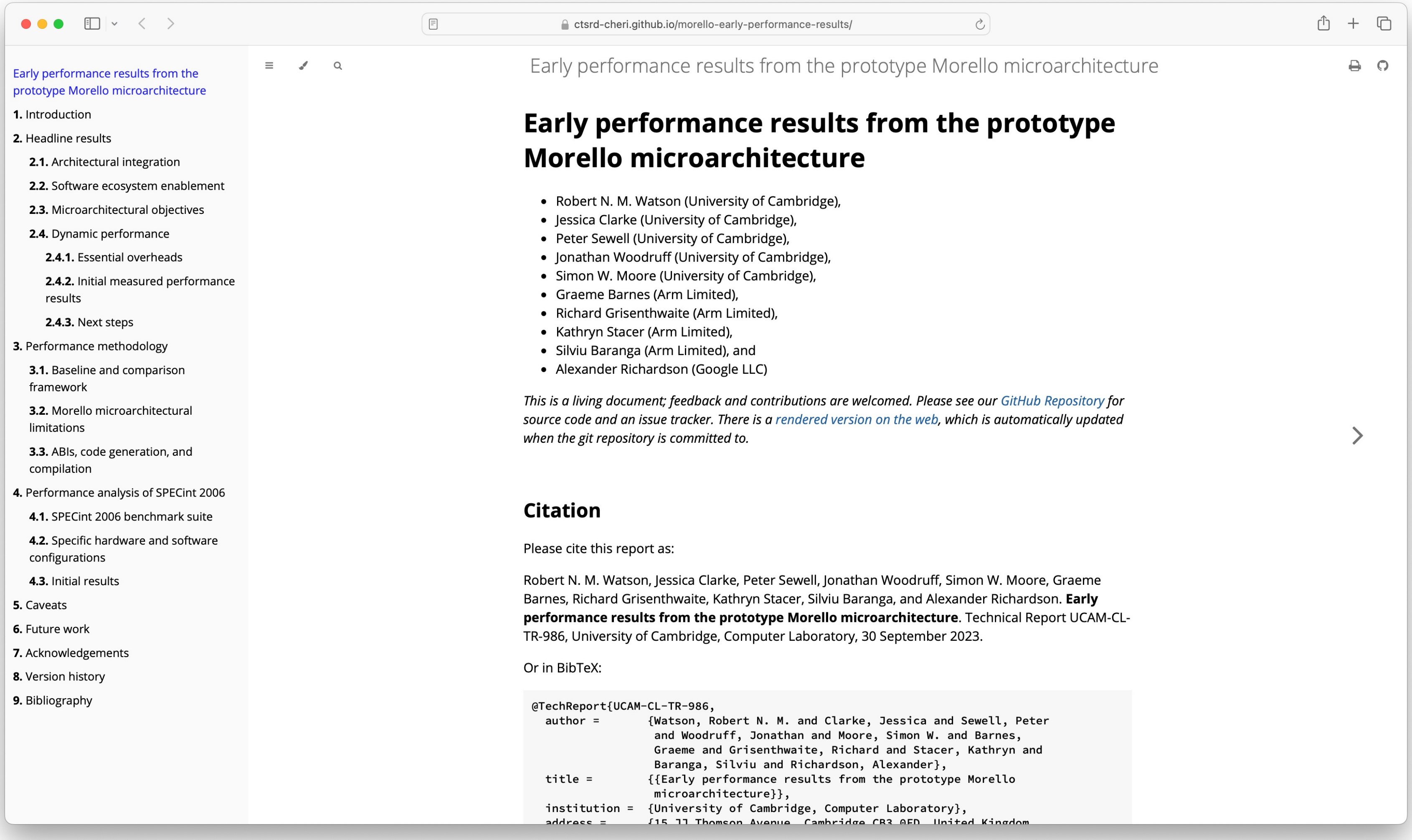
This work was supported in part by the Innovate UK project Digital Security by Design (DSbD) Technology Platform Prototype, 105694 and Innovate UK project Developing and Evaluating an Open-Source Desktop for Arm Morello, 10027440.

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”), HR001122C0110 (“ETC”), and HR001123C0031 (“MTSS”) as part of the DARPA I2O CRASH, I2O MRC, and MTO SSITH research programs, and by the Office of Naval Research (ONR) under contract N00014-22-1-2463 (“SWISH”). 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 E{SRC CHaOS grant, 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.

Introduction

- Morello is a first-generation prototype
  - Developed as an extension to Neoverse N1, not from-scratch
  - Morello ISA aspects of the design are not as well optimized as baseline Armv8.2a
  - A full hardware optimization cycle was not permitted on the research timeline
  - Little or no workload data available [yet] to drive optimisation cycle
- Frequency (2.5GHz) and area overhead (<6%) objectives met
- Prototype microarchitecture has quirks affecting performance, which would be resolved in a mature second-generation implementation
- In this workshop:
  - Learn about the microarchitecture
  - Learn about its effects on performance
  - Learn how to conduct and interpret performance measurements



Early performance results from the prototype Morello microarchitecture
1. Introduction
2. Headline results
2.1. Architectural integration
2.2. Software ecosystem enablement
2.3. Microarchitectural objectives
2.4. Dynamic performance
2.4.1. Essential overheads
2.4.2. Initial measured performance results
2.4.3. Next steps
3. Performance methodology
3.1. Baseline and comparison framework
3.2. Morello microarchitectural limitations
3.3. ABIs, code generation, and compilation
4. Performance analysis of SPECint 2006
4.1. SPECint 2006 benchmark suite
4.2. Specific hardware and software configurations
4.3. Initial results
5. Caveats
6. Future work
7. Acknowledgements
8. Version history
9. Bibliography

Early performance results from the prototype Morello microarchitecture

Early performance results from the prototype Morello microarchitecture

- Robert N. M. Watson (University of Cambridge),
- Jessica Clarke (University of Cambridge),
- Peter Sewell (University of Cambridge),
- Jonathan Woodruff (University of Cambridge),
- Simon W. Moore (University of Cambridge),
- Graeme Barnes (Arm Limited),
- Richard Grisenthwaite (Arm Limited),
- Kathryn Stacer (Arm Limited),
- Silviu Baranga (Arm Limited), and
- Alexander Richardson (Google LLC)

This is a living document; feedback and contributions are welcomed. Please see our GitHub Repository for source code and an issue tracker. There is a rendered version on the web, which is automatically updated when the git repository is committed to.

Citation

Please cite this report as:

Robert N. M. Watson, Jessica Clarke, Peter Sewell, Jonathan Woodruff, Simon W. Moore, Graeme Barnes, Richard Grisenthwaite, Kathryn Stacer, Silviu Baranga, and Alexander Richardson. **Early performance results from the prototype Morello microarchitecture**. Technical Report UCAM-CL-TR-986, University of Cambridge, Computer Laboratory, 30 September 2023.

Or in BibTeX:

```
@TechReport{UCAM-CL-TR-986,
  author =       {Watson, Robert N. M. and Clarke, Jessica and Sewell, Peter
                  and Woodruff, Jonathan and Moore, Simon W. and Barnes,
                  Graeme and Grisenthwaite, Richard and Stacer, Kathryn and
                  Baranga, Silviu and Richardson, Alexander},
  title =        {{Early performance results from the prototype Morello
                  microarchitecture}},
  institution =  {University of Cambridge, Computer Laboratory},
  address =      {15 JJ Thomson Avenue, Cambridge CB3 0FD, United Kingdom
```

Technical Report

UCAM-CL-TR-986
ISSN 1476-2986

Number 986

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Early performance results from the prototype Morello microarchitecture

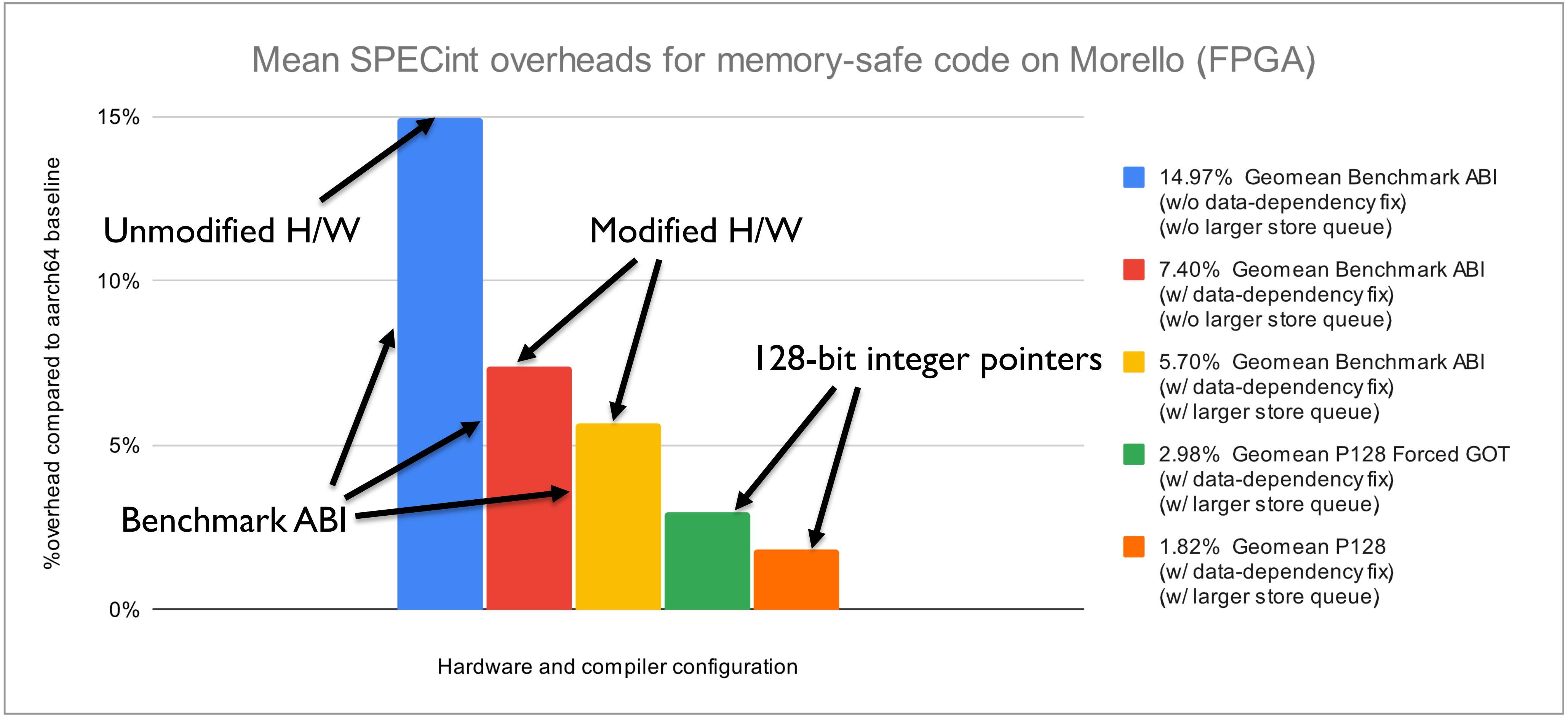
Robert N. M. Watson, Jessica Clarke,
Peter Sewell, Jonathan Woodruff,
Simon W. Moore, Graeme Barnes,
Richard Grisenthwaite, Kathryn Stacer,
Silviu Baranga, Alexander Richardson

September 2023

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500
https://www.cl.cam.ac.uk/

https://ctsrd-cheri.github.io/morello-early-performance-results/

Headline results



Capability branch prediction

- Microarchitecture only predicts PCC's address in the Morello prototype
  - This is due to the research engineering timeline, lack of optimization data, and desire to avoid floorplan changes
  - Arm has strong confidence that this could be addressed in a production microarchitecture
- Instructions that consume PCC's metadata (e.g. C64 BL/BLR and ADRP) need to wait for prior capability branches (NB: includes RET) to execute
  - Includes ADRP+LDR sequence to load from GOT for globals
- Capability branch-heavy code incurs additional stalls

Benchmark ABI: Overview

- Aims to work around lack of capability branch prediction
- Models expected performance of an improved second-generation microarchitecture
- PCC given bounds for the whole address space
- Indirect branches and returns use integer branches
  - Return addresses and function pointers remain as capabilities in memory; only branches themselves altered
- NB: Weakens control flow protection, **not intended for security evaluation**

Benchmark ABI: Compiling

- Pure-capability (standard): `cc -mabi=purecap`
- Pure-capability benchmark ABI: `cc -mabi=purecap-benchmark`
- Preprocessor macro: `__ARM_MORELLO_PURECAP_BENCHMARK_ABI`

Benchmark ABI: Identifying (1/2)

- Use CheriBSD’s file command:

```
$ file helloworld-purecap
helloworld-purecap: ELF 64-bit LSB pie executable, ARM aarch64,
C64, CheriABI, version 1 (SYSV), dynamically linked,
interpreter /libexec/ld-elf.so.1, for FreeBSD 14.0 (1400094),
FreeBSD-style, with debug_info, not stripped

$ file helloworld-purecap-benchmark
helloworld-purecap-benchmark: ELF 64-bit LSB pie executable,
ARM aarch64, C64, CheriABI, version 1 (SYSV), dynamically
linked, interpreter /libexec/ld-elf.so.1, for FreeBSD 14.0
(1400094), FreeBSD-style, pure-capability benchmark ABI, with
debug_info, not stripped
```

Benchmark ABI: Identifying (2/2)

- Use Morello LLVM’s llvm-readelf command:

```
$ llvm-readelf -n helloworld-purecap
...
Displaying notes found in: .note.cheri
  Owner                Data size  Description
  ...
  CHERI                0x00000004 NT_CHERI_MORELLO_PURECAP_BENCHMARK_ABI
(Morello purecap benchmark ABI)
    Purecap benchmark ABI enabled: 0 (no)

$ llvm-readelf -n helloworld-purecap-benchmark
...
Displaying notes found in: .note.cheri
  Owner                Data size  Description
  ...
  CHERI                0x00000004 NT_CHERI_MORELLO_PURECAP_BENCHMARK_ABI
(Morello purecap benchmark ABI)
    Purecap benchmark ABI enabled: 1 (yes)
```

Benchmark ABI: Differences (1/4)

PCC has bounds covering the entire (user) address space

```
int main(void) { printf("%#p\n", cheri_pcc_get()); return (0); }

$ ./pcc-bounds-purecap
PCC: 0x1107cc [rxR,0x100000-0x130b00]

$ ./pcc-bounds-purecap-benchmark
PCC: 0x1107cc [rxR,0x0-0x1000000000000]
```

Benchmark ABI: Differences (2/4)

LSB of functions always 0, but still C64 code

```
$ llvm-readelf -s purecap
   ...
   Num:    Value          Size Type    Bind   Vis       Ndx Name
    ...
    42: 00000000000107c1    56 FUNC    GLOBAL DEFAULT    15 main

$ llvm-readelf -s purecap-benchmark
   ...
   Num:    Value          Size Type    Bind   Vis       Ndx Name
    ...
    42: 00000000000107c0    60 FUNC    GLOBAL DEFAULT    15 main
```

When using a capability jump target, an LSB of 1 sets the C64 opcode interpretation, and is then ignored.
When using an integer jump target, the LSB of 1 is used as instruction fetch address, so we must clear it.

Benchmark ABI: Differences (3/4)

Indirect calls use integer branch instructions

```
void foo(void (*f)(void)) { ...; (*f)(); ... }

$ llvm-objdump -d purecap
   ...
   blr   c0
   ...

$ llvm-objdump -d purecap-benchmark
   ...
   blr   x0
   ...
```

Benchmark ABI: Differences (4/4)

Returns use integer branch instructions and mask LSB (from BL/BLR)

```
$ llvm-objdump -d purecap
   ...
   ret    c30

$ llvm-objdump -d purecap-benchmark
   ...
   and    x30, x30, #~1
   ret    x30
```

Data-dependent exception delivery

- Used to track capabilities for heap temporal safety
  - Deliver a precise exception based on the value stored to memory, not just the address it is stored to
- Not a requirement in the baseline Neoverse N1 design, and as a result there isn't the necessary plumbing to make it microarchitecturally efficient
  - Stores of capabilities stall until both address and data are known
- A similar requirement affects recent Arm microarchitectures
- Modified Morello design on FPGA allows us to experiment with eliminating this overhead

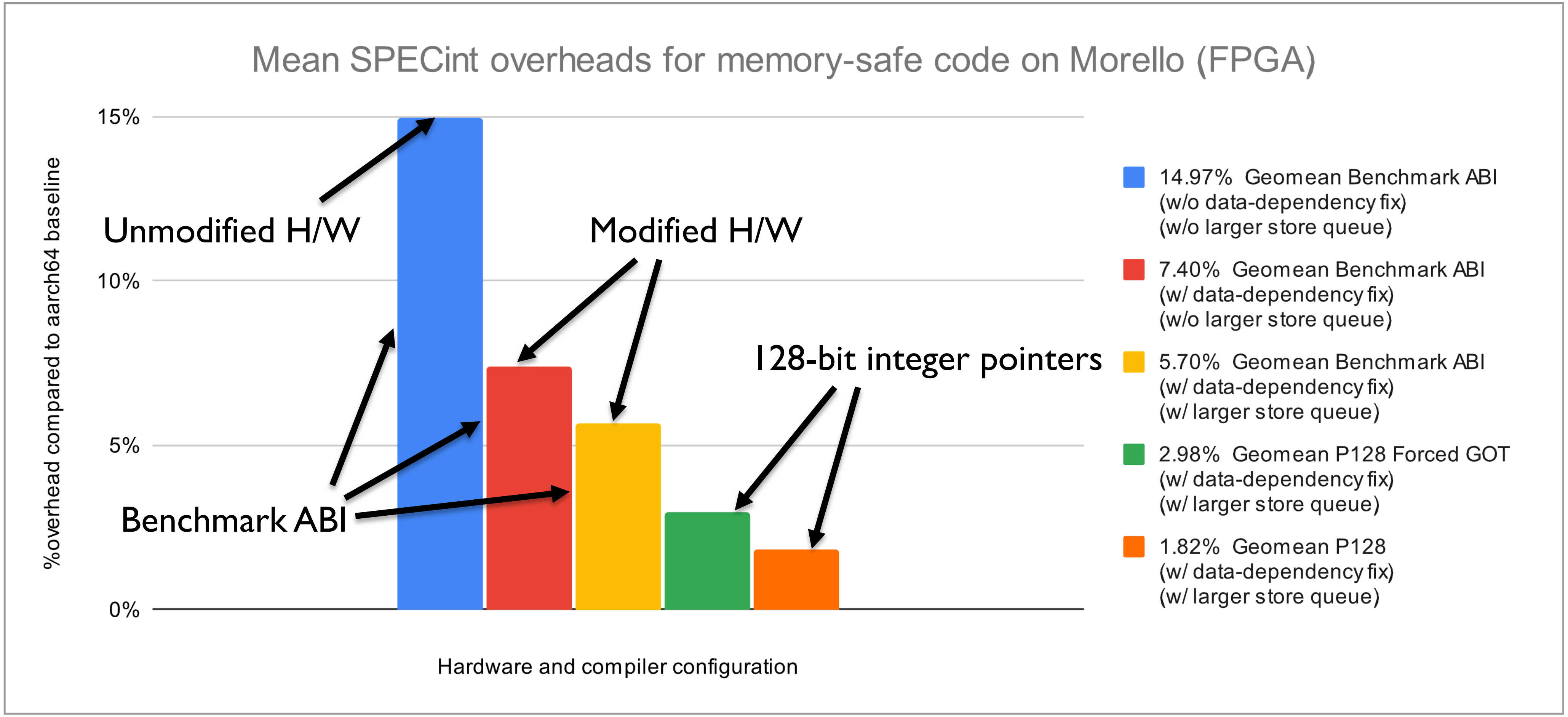
Untuned store queues

- The baseline Neoverse N1 has store-buffer queues (which track in-flight memory stores) tuned to the memory traffic generated by the Armv8-A
  - With a 128-bit bus, “store pair” instructions for 64-bit integers could be issued as a single operation
- Morello has “store pair” instructions for 128-bit capabilities
  - These cannot be satisfied by a single 128-bit memory operation
  - Store pair capability is therefore “cracked” microarchitecturally into two 128-bit operations
  - The store-buffer queue can become full as a result of the potential to double the number of in-flight transactions, stalling memory accesses
  - Modified Morello design on FPGA allows us to experiment with increasing the store-buffer queue size

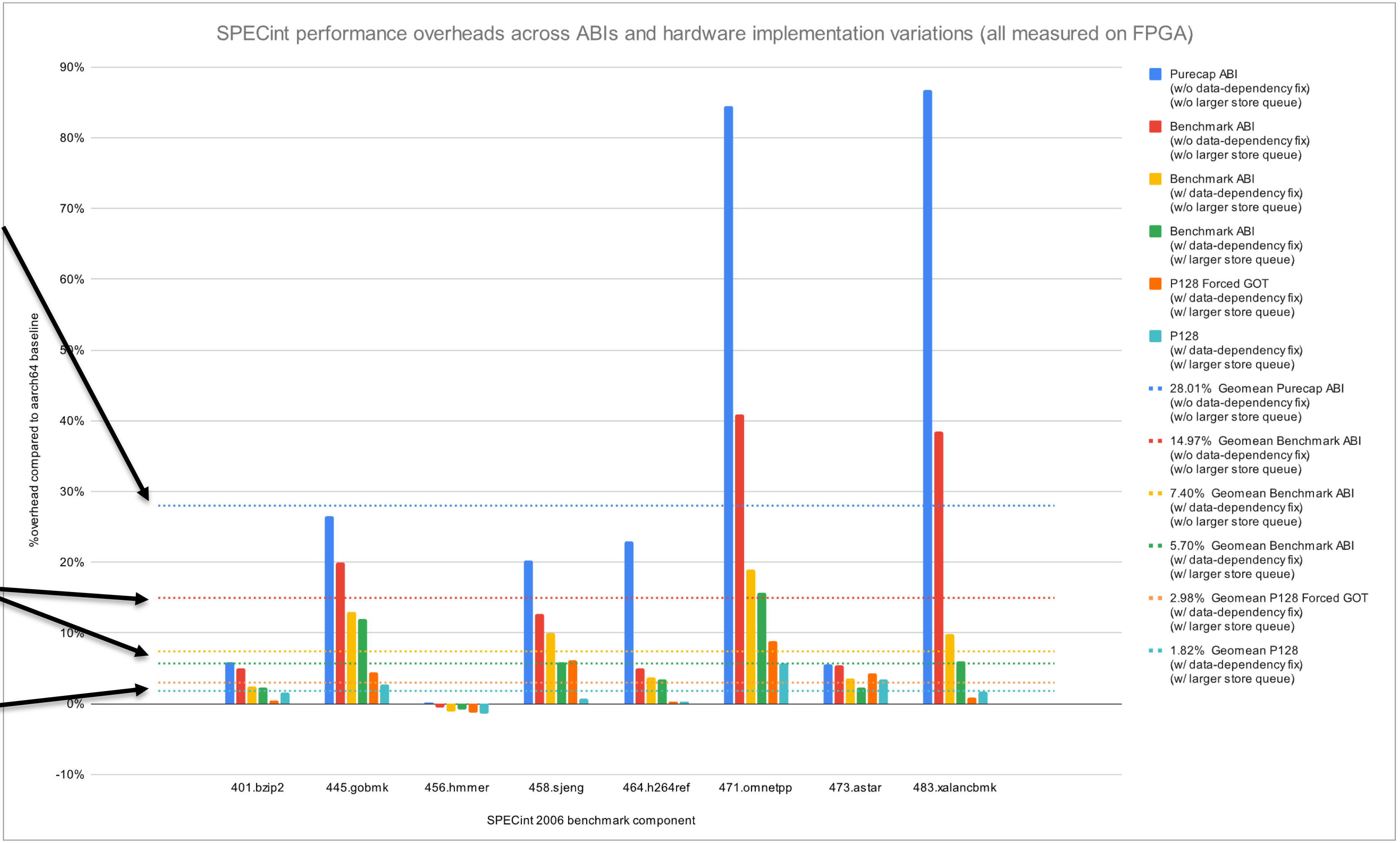
P128 code generation

- A key conclusion of the Morello project is somewhat expected: that the essential overhead to CHERI is pointer-size growth (64 → 128 bits)
  - Other costs, such as the implementation of tags, capability compression, instruction scheduling, etc., turned out not to be significant in this work
- To understand how a more optimized and mature microarchitecture might perform, we modified Morello LLVM to target the Armv8.2-A ISA while using 128-bit storage for language-level pointers to identify new upper bounds for overheads
  - Sub-language pointers (GOT entries, return addresses, etc.) currently remain as 64-bit integers
  - Treated as 64-bit values when in registers (NB: including spilling to stack)
- Two variants depending on whether (a) all loads and stores are forced through the GOT, or (b) PC-relative loads and stores are used
  - A mature CHERI-enabled compiler would use a combination of the two strategies based on security and performance considerations

Headline results (redux)



SPECint benchmark breakdown

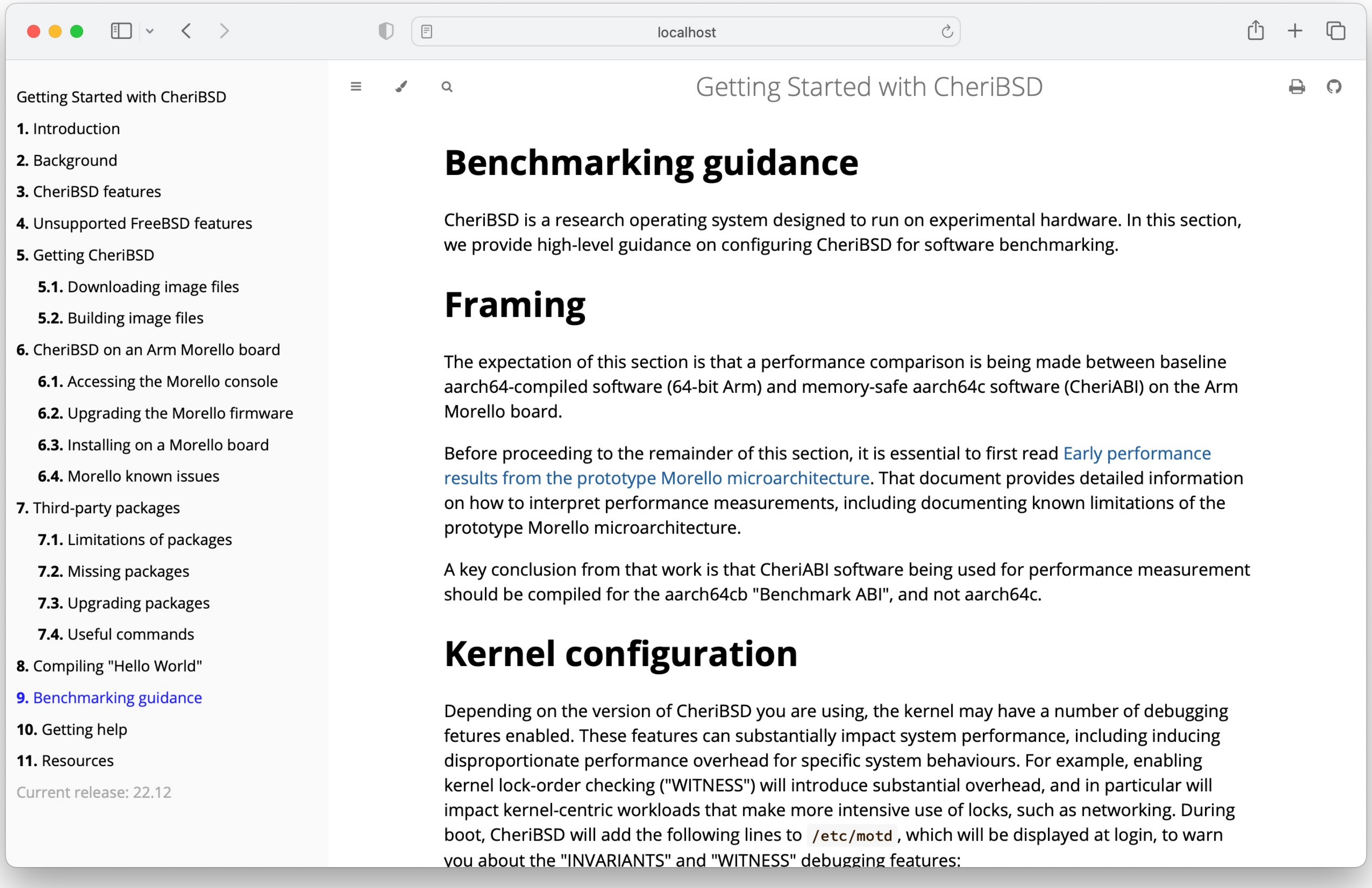


Purecap ABI (not shown on prior slides)

Benchmark ABI

128-bit integer pointers

Benchmarking guidance



Getting Started with CheriBSD
1. Introduction
2. Background
3. CheriBSD features
4. Unsupported FreeBSD features
5. Getting CheriBSD
5.1. Downloading image files
5.2. Building image files
6. CheriBSD on an Arm Morello board
6.1. Accessing the Morello console
6.2. Upgrading the Morello firmware
6.3. Installing on a Morello board
6.4. Morello known issues
7. Third-party packages
7.1. Limitations of packages
7.2. Missing packages
7.3. Upgrading packages
7.4. Useful commands
8. Compiling "Hello World"
9. Benchmarking guidance
10. Getting help
11. Resources
Current release: 22.12

Benchmarking guidance

CheriBSD is a research operating system designed to run on experimental hardware. In this section, we provide high-level guidance on configuring CheriBSD for software benchmarking.

Framing

The expectation of this section is that a performance comparison is being made between baseline aarch64-compiled software (64-bit Arm) and memory-safe aarch64c software (CheriABI) on the Arm Morello board.

Before proceeding to the remainder of this section, it is essential to first read Early performance results from the prototype Morello microarchitecture. That document provides detailed information on how to interpret performance measurements, including documenting known limitations of the prototype Morello microarchitecture.

A key conclusion from that work is that CheriABI software being used for performance measurement should be compiled for the aarch64cb "Benchmark ABI", and not aarch64c.

Kernel configuration

Depending on the version of CheriBSD you are using, the kernel may have a number of debugging fetures enabled. These features can substantially impact system performance, including inducing disproportionate performance overhead for specific system behaviours. For example, enabling kernel lock-order checking ("WITNESS") will introduce substantial overhead, and in particular will impact kernel-centric workloads that make more intensive use of locks, such as networking. During boot, CheriBSD will add the following lines to `/etc/motd`, which will be displayed at login, to warn you about the "INVARIANTS" and "WITNESS" debugging features:

- Updates to the **Getting Started with CheriBSD 23.11** guide include a new Benchmarking Guidance section
- This is a living document, and feedback / suggestions are greatly appreciated
- A few highlights here …

CheriBSD benchmarking: Kernel

- Default kernel has various debugging options enabled, which gives the following warnings at login:

```
WARNING: INVARIANTS kernel option defined, expect reduced
performance
WARNING: WITNESS kernel option defined, expect reduced
performance
```

- Use a `-NODEBUG` kernel when benchmarking (requires rebooting):
  - Use option `6. Kernel: default/kernel (1 of 4)` in CheriBSD's loader menu
  - Set `kernel="kernel.GENERIC-MORELLO-NODEBUG"`, or `kernel="kernel.GENERIC-MORELLO-PURECAP-NODEBUG"` if you require a memory-safe kernel, in `/boot/loader.conf` to change default

CheriBSD benchmarking: Userspace

- Capability revocation for userspace heap temporal safety is enabled by default, which gives the following warning at login:

```
WARNING: capability revocation enabled by default, this
may affect performance
```

- Disable revocation when benchmarking (requires rebooting):
  - Set `security.cheri.runtime_revocation_default=0` in `/boot/loader.conf`

General methodology

- Use the benchmark ABI to avoid PCC-based stalls
- Use Morello’s ISA baseline (Armv8.2-A plus extensions) for non-CHERI rather than plain Armv8.0-A
- Only compare results running on the same hardware
  - Morello’s configuration (e.g. caches) differs from N1SDP
  - Other N1-based designs may differ in additional ways
- Enable compiler optimisations (-O2 or -O3 recommended)
  - -O0 code’s performance not meaningful
- If you see unexpected overheads or need help interpreting results, please contact us

Next directions

- Work so far has focused on static linkage – next step is to switch to dynamic linkage – most likely more affected by PCC bounds issue than static linking

- Would like to widen workloads to include language runtimes (e.g.,V8 as that matures) and also potentially as-yet unexplained (e.g., KCL microbenchmarks) – other suggestions welcome

- Will have continuing access to modified Morello on FPGA, but using it is time consuming and expensive – but offers incredibly detailed insight into instruction scheduling, etc.

- The website is a live document, which we will update as we learn more.

Q&A