27 March 2025

Porting V8 to CHERI

An Overview

Domagoj Stolfa, Graeme Jenkinson,
Mingle Chen, Alfredo Mazzinghi,
Robert N. M. Watson
Capabilities Limited

What is V8?

- JavaScript and WebAssembly language runtime.
- Roughly 2 million lines of C++ code, with an additional 300-400 thousand generated during build time.
- Crucial part of Chromium, NodeJS, Deno, Electron, Edge, CEF…
- 6 JIT compilers, 1 AOT compiler, multiple allocators and GCs.
- Can be used in compressed and uncompressed pointer configuration.
  - Pointer compression turns each JS heap pointer into a 31-bit integer.
  - We focus on uncompressed in this work to make full use of capabilities.
- At least 19 critical memory-safety vulnerabilities from July 2023 - July 2024 (1 year span from our version of V8).
- V8 is not just architecture specific in code generation, but also incorporates strong assumptions about integers/pointers in architecture-neutral code

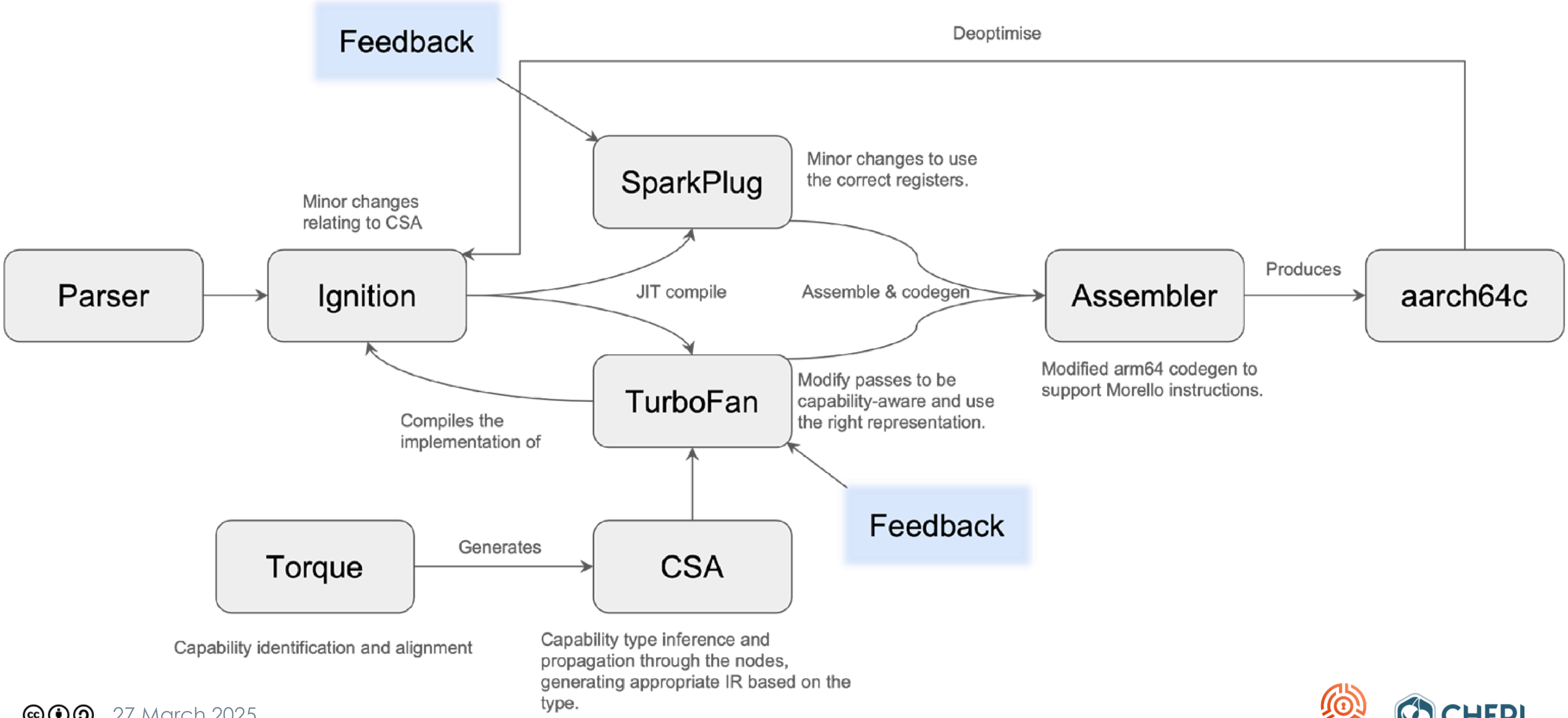
Issues encountered

- Old Google style guide, using `intptr_t, uintptr_t` for machine words.
  - Cannot assume that an `intptr_t` or `uintptr_t` will be a capability, because it isn't most of the time.
- Ambiguous pointer provenance.
  - Depending on the code path, pointers can end up in nodes that normally hold offsets, and offsets end up in the nodes that hold pointers. Needs runtime checks.
- Assumes everything is packed – alignment can be tricky to add.
- Assumes `sizeof(double) >= sizeof(void*)`.
  - Causes problems in snapshot serialisation and deserialisation.
- Pointer and integer conflation embedded into all the DSLs and IR.
- … and many other smaller issues.

Successes

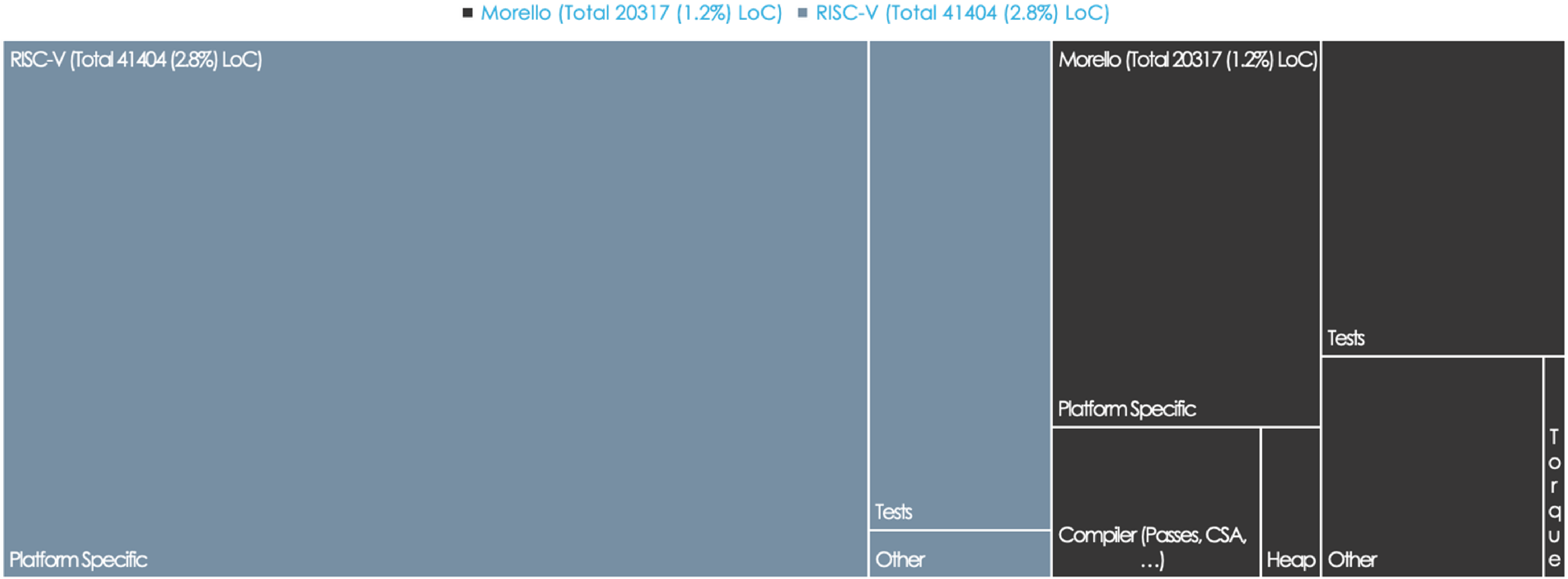
- Automated capability marking propagation dependent on node origins, types and IR opcodes in CodeStubAssembler (CSA).
  - `if constexpr (is_capability<T>::value) { … }`
- Able to automate most of the alignment requirements in Torque and heap allocators.
  - Torque compiler needs changes to identify things that could be capabilities.
- Able to handle a good amount of real-world JavaScript on websites visited via purecap Chromium.
- Most of the work was done over the course of 6 months by one staff member – which is not a lot of work for a high-friction porting activity!
  - In comparison: The Chromium security team alone has over 100 staff members!

⬡ Summary of changes (simplified)


Feedback
Deoptimise
SparkPlug
Minor changes to use the correct registers.
Minor changes relating to CSA
Parser
Ignition
JIT compile
Assemble & codegen
Assembler
Produces
aarch64c
TurboFan
Modify passes to be capability-aware and use the right representation.
Modified arm64 codegen to support Morello instructions.
Compiles the implementation of
Feedback
Torque
Generates
CSA
Capability identification and alignment
Capability type inference and propagation through the nodes, generating appropriate IR based on the type.

⬡ Adding RISC-V vs porting to Morello (so far)

Tree Map of LoC changes between RISC-V and Morello



Test results

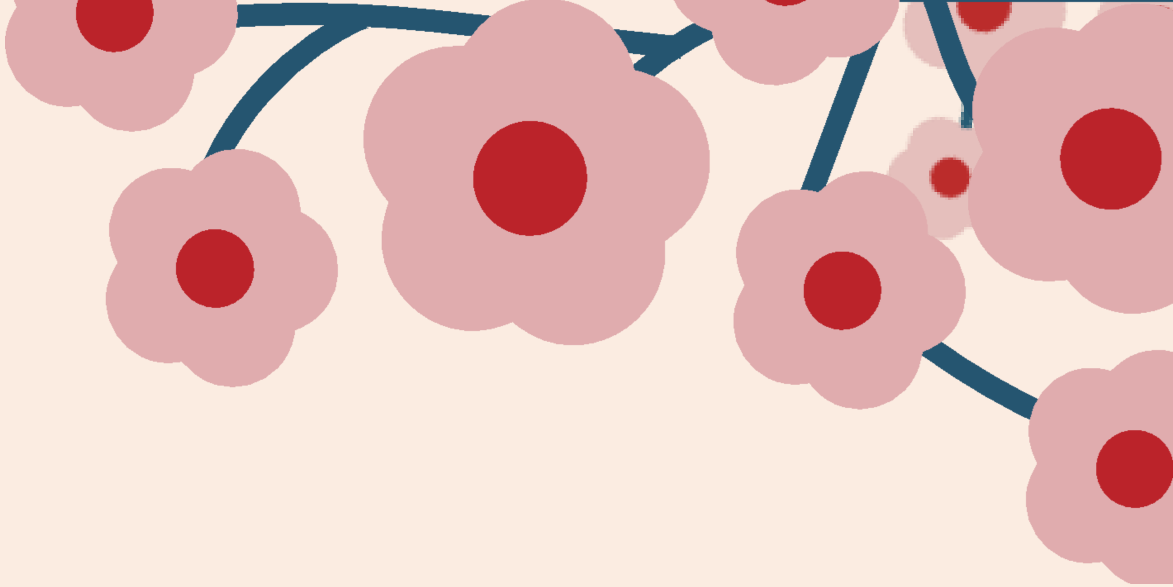
Test suite	With JIT	Without JIT
unittests	4986 Pass / 100 Fail	2910 Pass / 70 Fail
cctest	2993 Pass / 62 Fail	Not Applicable
mjsunit	4823 Pass / 479 Fail	5172 Pass / 122 Fail
test262	91937 Pass / 1529 Fail	92179 Pass / 1287 Fail
mozilla	1761 Pass / 147 Fail	1904 Pass / 4 Fail
webkit	505 Pass / 37 Fail	528 Pass / 14 Fail
message	313 Pass / 0 Fail	313 Pass / 0 Fail
intl	274 Pass / 22 Fail	285 Pass / 11 Fail
inspector	199 Pass / 148 Fail	338 Pass / 9 Fail
debugger	277 Pass / 27 Fail	298 Pass / 6 Fail
fuzzer	35 Pass / 0 Fail	35 Pass / 0 Fail
benchmarks	26 Pass / 29 Fail	52 Pass / 3 Fail
wasm-js	Not Applicable	Not Applicable
wasm-api-tests	Not Applicable	Not Applicable
wasm-spec-tests	Not Applicable	Not Applicable

Limitations

- WebAssembly adaptation not yet started.
- Currently only uncompressed pointers are supported.
  - Probably not far from working
- No support for Maglev (cache-friendly CFG compiler) yet.
  - Doesn’t work in uncompressed pointer configuration on our baseline commit.
- Doesn’t yet work well enough to handle Node’s snapshotting process.
- A version from 5th July 2023. Needs to be pulled forward.
  - Currently stuck because of the API version that our version of Chromium uses.

Future directions

- Tightening bounds on all JavaScript objects.
- WebAssembly support, using CHERI to make Memory64 cheaper.
- Support for pointer compression.
- Maglev support and merging forward to the latest versions of V8 more frequently.
- Support for NodeJS, Deno.
- CHERI-RISC-V support – unclear how mature baseline RISC-V support is.



CHERI

THANK YOU

Contact domagoj@capabilitieslimited.co.uk
ds815@cam.ac.uk

Web www.capabilitieslimited.co.uk

This work was generously funded by UKRI DSbD Desktop/MOJO projects.

27 March 2025

CAPABILITIES
LIMITED