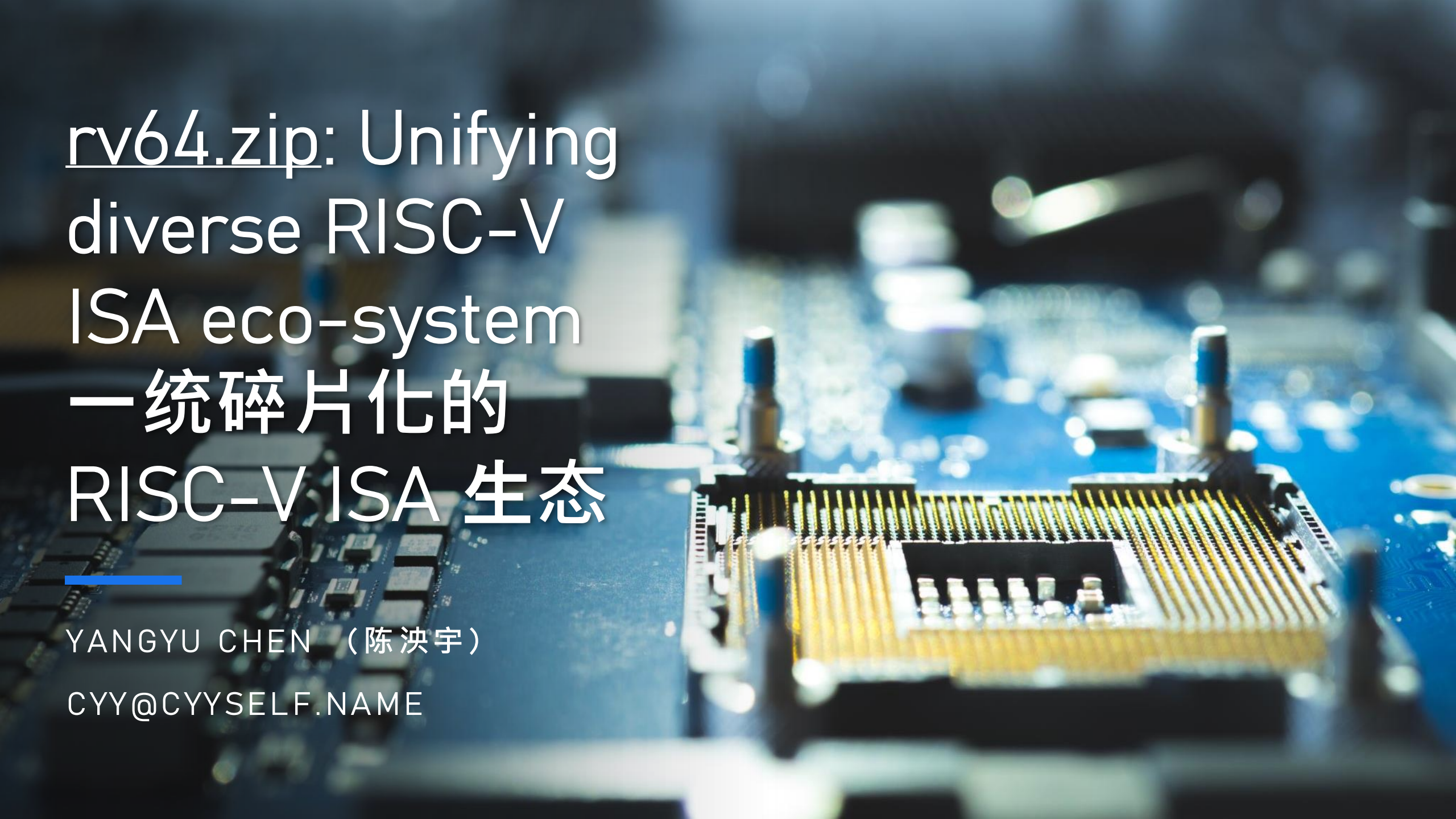


rv64.zip: Unifying diverse RISC-V ISA eco-system
一统碎片化的 RISC-V ISA 生态

YANGYU CHEN （陈泱宇）

CYY@CYYSELF.NAME

Background

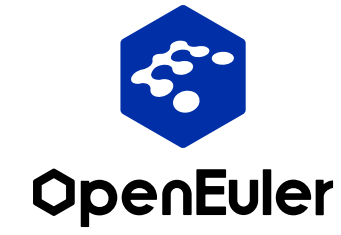
- Our dream: RVA23U64
- However, existing hardware:

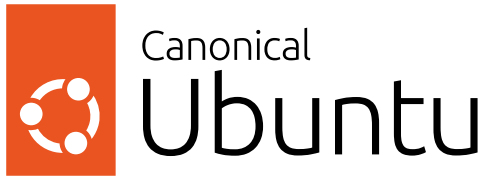
XUANTIE	C920v1	2023	RVA20	
SiFive	P550	2024	RVA20+	Zba Zbb
SPACEMIT 进迭时空	X60	2024	RVA22+	Vector Zicond
XIANGSHAN	Nanhu v2	2024	RVA20+	Zba Zbb Zbs

RVA23U64 Mandatory Extensions

Software distribution

- Binary distribution: RV64GC
- Manually dispatch multi-version:
  - glibc
  - ffmpeg
  - OpenSSL
  - ...



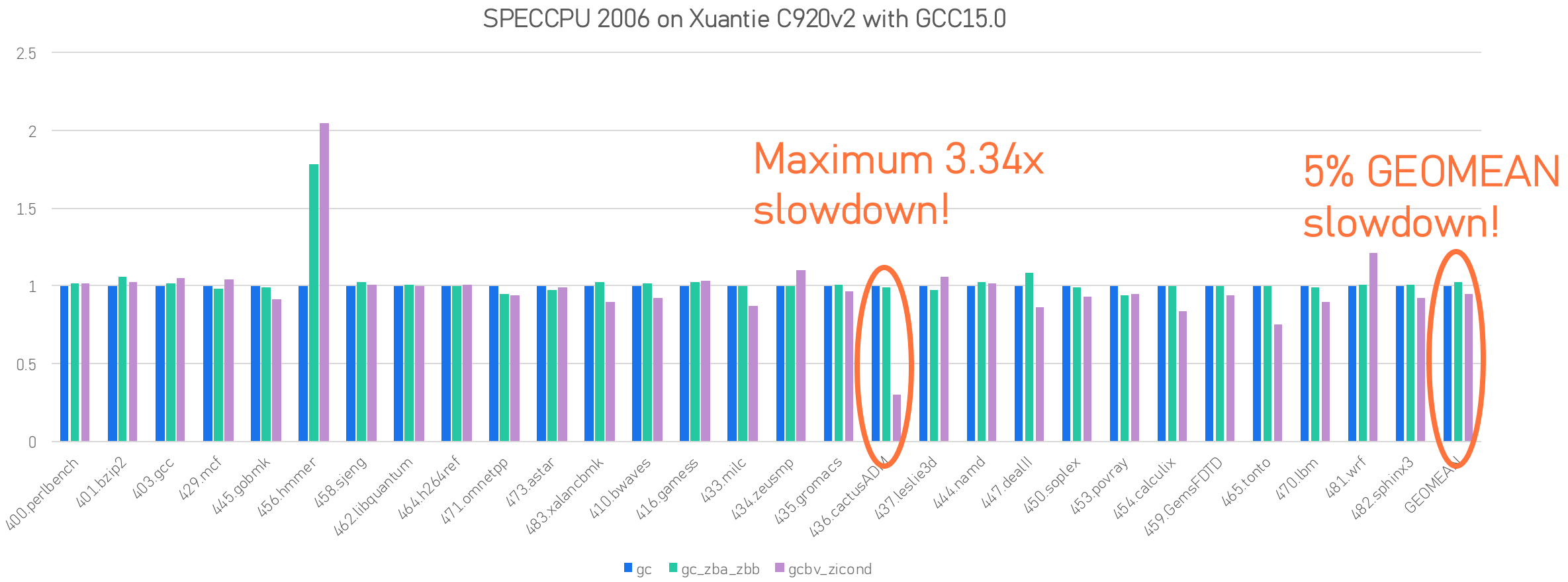


RVA23 distro is not as beautiful as your imagination

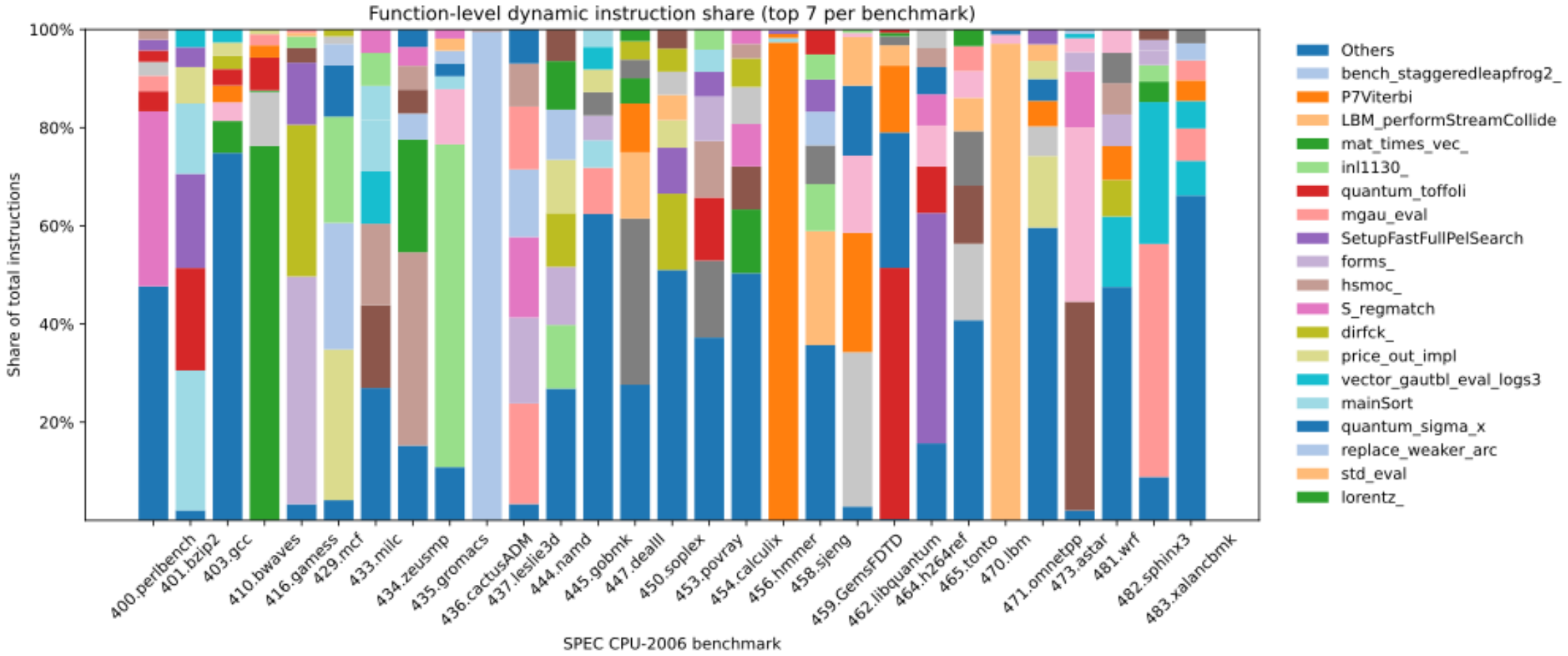
- RVA23 compilation can yield worse performance than RV64GC with current compiler and hardware, especially for Vector extensions.
- RISC-V is not alone.
  - This issue is also present on the aarch64 with the SVE2 [1]

[1] https://blog.cyyself.name/wp-content/uploads/2025/02/Screenshot-2025-02-09-at-19.17.59.png

Performance degradation with new Extensions



Opportunities are always there



Remark: Recorded using GCC14.2 + O3 + RV64GC

Observation

- New extensions can also limit the performance
- The number of performance-sensitive functions is small

Solution: Function clone of different extensions

- Preserve binary compatibility (even with RV64GC)
- Higher performance on RVA23 hardware via dynamic dispatch

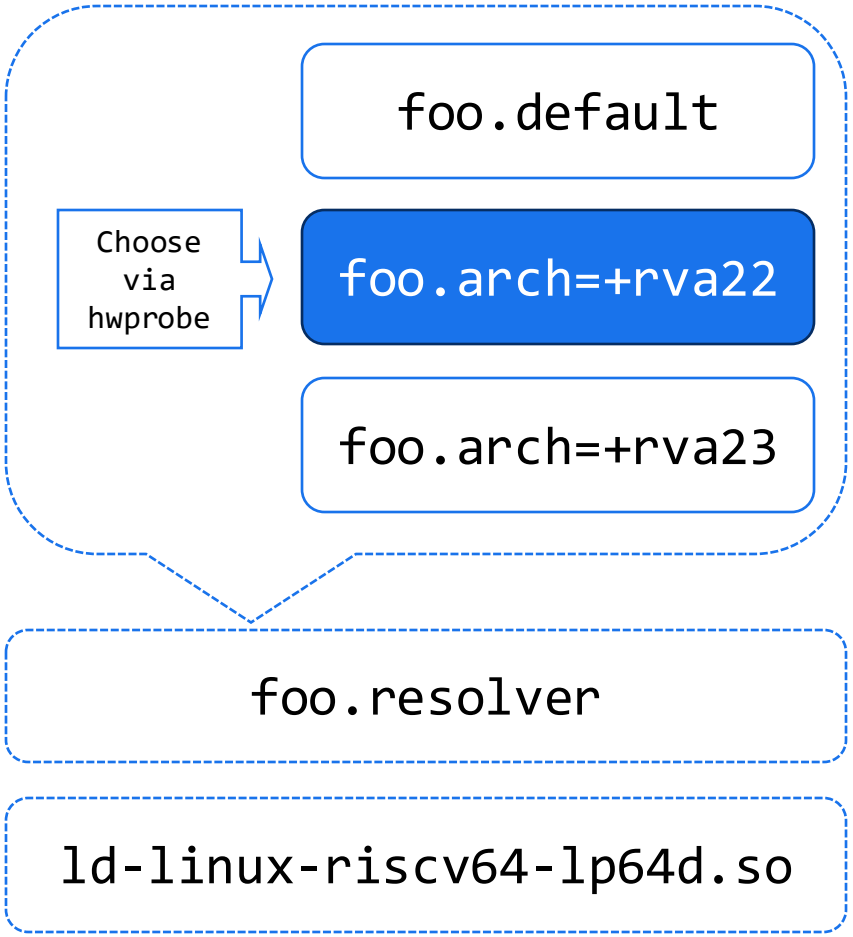
Existing function-clone solutions

- Using target_clones in GCC (>=15) / LLVM (>=20), based on IFUNC

```
__attribute__((target_clones("default", "arch=+zba,+zbb", "arch=+b,+v,+zicond")))
void foo() { ... }
```

  - Require manual setting by the developer
  - Make the code non-portable between different ISAs (#ifdef __riscv)
- Software's own approach
  - Probe ISAs (RISC-V hwprobe) and call to different functions
  - Example: OpenSSL, ffmpeg, glibc

IFUNC based multi-versioning



Global Offset Table

Function	Pointer
foo	&foo.arch=+rva22
bar	
…	…
…	…

Procedure to call foo:

```
auipc t0, %hi(foo)
jalr  ra, %lo(foo)(t0)

Sometimes relaxed to:

jal   ra, foo
```

Indirect-call

```
auipc t0, %hi(foo)
ld    t0, %lo(foo)(t0)
jalr  ra, t0
```

How we can make it better

- Existing disadvantage:
  - Manually setting `__attribute__(target_clones(...))`
  - Make the code non-portable
  - Indirect-calling overhead

Newly introduced function clone table[1]

- JSON formatted table
- Separate from source code
- Combine all architectures
- Make binary reproducible
- Easy to use via:

`gcc -ftarget-clones-table=foo.json 1.c`

```
{
 "foo": {
  "x86_64": ["avx2", "avx512f"],
  "riscv64": ["arch=+v", "arch=+zba,+zbb"]
 }
 "bar": {
  "x86_64": ["popcnt"],
  "aarch64": ["sve2"]
 }
}
```

[1] https://patchwork.sourceware.org/project/gcc/cover/tencent_815F8860AE36BFA3102E4ECC77C843231606@qq.com/

Automatic generation of function clone table

- A Straight-forward way:
  - Use linux-perf to record the running cycles of each function on different extensions with the same input.
  - Generate function clone table based on perf result.
- A more advanced way:
  - It's my current research project
    - Welcome to join me together for a shared future!

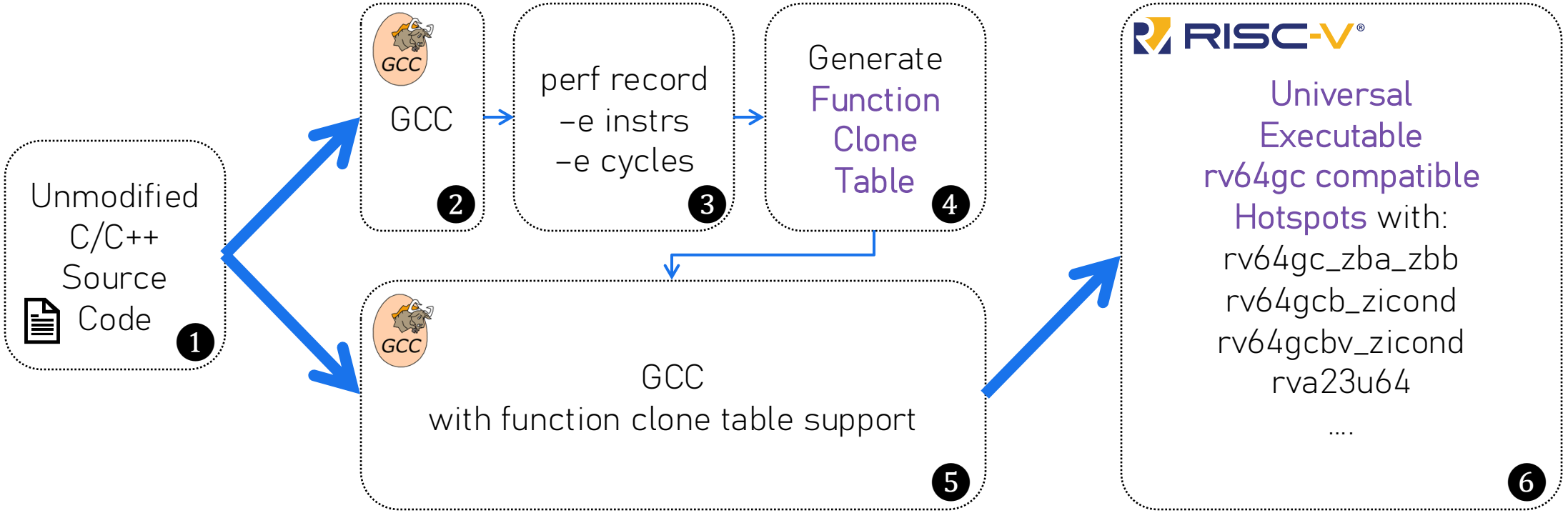
Indirect-calling overhead

- Indirect branch predictor
  - Nearly zero overhead with modern decoupled frontend

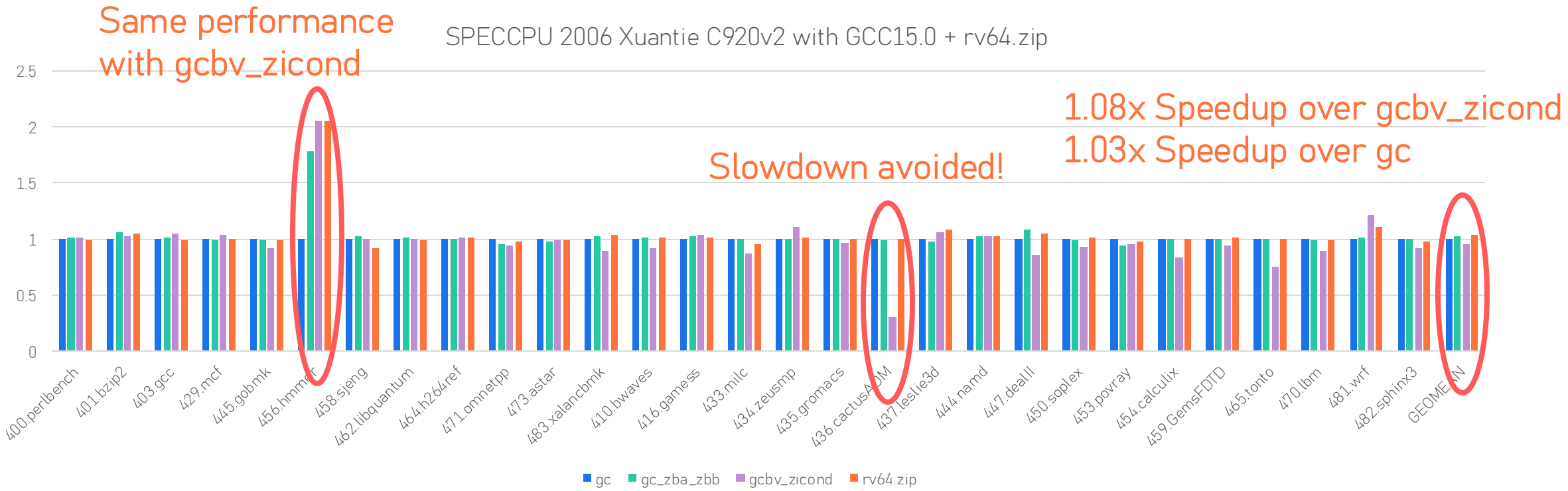
- Direct call to best compatible callee[1]
  - Works well with LTO

[1] https://patchwork.sourceware.org/project/gcc/patch/tencent_0F1B5FF8800411B43D1C07701F633E424009@qq.com/

Solution


Unmodified
C/C++
Source
Code
1
GCC
GCC
2
perf record
–e instrs
–e cycles
3
Generate
Function
Clone
Table
4
GCC
GCC
with function clone table support
5
RISC-V
Universal
Executable
rv64gc compatible
Hotspots with:
rv64gc_zba_zbb
rv64gcb_zicond
rv64gcbv_zicond
rva23u64
....
6

Results on SPECCPU 2006



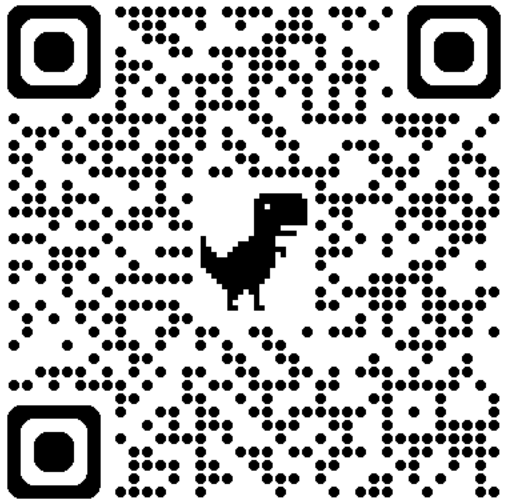
We only cloned 55 functions in the entire SPECCPU 2006 suite

Conclusion

- Need for auto-generating function multi-versioning
- How: Compile => Perf Record => Clone Table => Compile
- Binary reproducible, make distros happy
- Evaluated with CoreMark and SPECCPU 2006
  - Outperforms most possible ISA exts by 8.4% on C920v2
  - Retaining binary compatibility with base RV64GC

Thanks for listening

- Feel free to reach out if you have any questions or would like to collaborate with me.



https://rv64.zip

cyy@cyyself.name

Scan to join our group on  and