# Supporting architecture psABIs with GNU Guix

**Efraim Flashner**

```
This program interpreter self-identifies as:
/gnu/store/...-glibc-2.35/lib/ld-linux-x86-64.so.2

Shared library search path:
  (libraries located via /etc/ld.so.cache)
  /gnu/store/...-glibc-2.35/lib (system search path)

Subdirectories of glibc-hwcaps directories, in
priority order:
  x86-64-v4
  x86-64-v3 (supported, searched)
  x86-64-v2 (supported, searched)
```

# Where are the directories?

¯_(ツ)_/¯

# Where are the directories?

```
glibc.git $ git grep glibc-hwcaps
elf/tst-glibc-hwcaps-2-cache.script:
mkdirp 0770 $L/glibc-hwcaps/x86-64-v2
mkdirp 0770 $L/glibc-hwcaps/x86-64-v3
cp $B/elf/libx86-64-isa-level-3.so \
    $L/glibc-hwcaps/x86-64-v2/libx86-64-isa-level.so
cp $B/elf/libx86-64-isa-level-4.so \
    $L/glibc-hwcaps/x86-64-v3/libx86-64-isa-level.so
```

# Where are the directories?

```
glibc.git $ git grep const\ char\ _dl_hwcaps_subdirs
sysdeps/powerpc/powerpc64/le/dl-hwcaps-subdirs.c: \
    const char _dl_hwcaps_subdirs[] = \
    "power10:power9";
sysdeps/s390/s390-64/dl-hwcaps-subdirs.c: \
    const char _dl_hwcaps_subdirs[] = \
    "z16:z15:z14:z13";
sysdeps/x86_64/dl-hwcaps-subdirs.c: \
    const char _dl_hwcaps_subdirs[] = \
    "x86-64-v4:x86-64-v3:x86-64-v2";
```

# Where are the directories?

- /lib/
- /lib/glibc-hwcaps/x86-64-v2/
- /lib/glibc-hwcaps/x86-64-v3/
- /lib/glibc-hwcaps/x86-64-v4/

# Where are the directories?

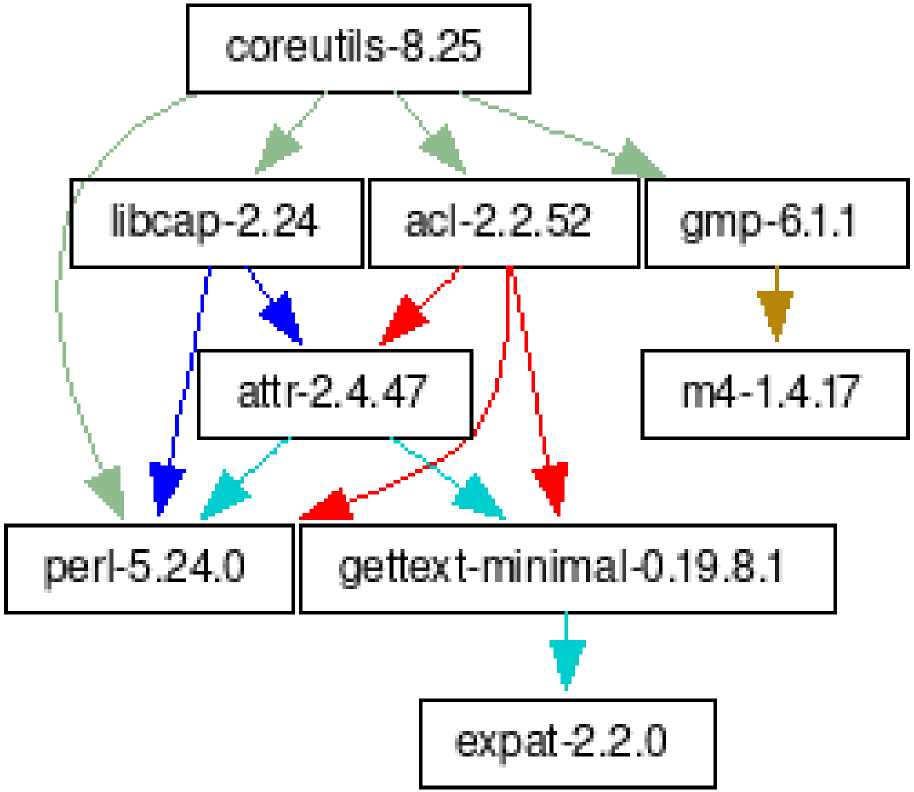
- /lib/
- /lib/glibc-hwcaps/power9/
- /lib/glibc-hwcaps/power10/

* little-endian only

# Where are the directories?

- /lib/
- /lib/glibc-hwcaps/z13/
- /lib/glibc-hwcaps/z14/
- /lib/glibc-hwcaps/z15/
- /lib/glibc-hwcaps/z16/

# Directed Acyclic Graph



# Is it worth it?

# Is it worth it?

**Does it matter?**


GIANT SIZE
Good Source of FIBER & Made with WHOLE GRAIN
Buena fuente de fibra y hecho con granos enteros
Kellogg's
-funroll-
LOOPS
SWEETENED MULTI-GRAIN CEREAL
Cereal multigrano endulzado
NATURAL FRUIT FLAVORS
Sabores de frutas naturales
Kellogg's FAMILY REWARDS
110 CALORIES
0.5g SAT FAT 3% DV
150mg SODIUM 6% DV
10g SUGARS
PER 1 CUP SERVING

# ncdu x86-64-v1 vs x86-64-v3

| | | | |
|---|---|---|---|
| 238 | » mov····%rcx,0x20(%rax) | 238 | » mov····%rcx,0x20(%rax) |
| 239 | » movups·-0x68(%rbp),%xmm0 | 239 | » vmovups·-0x68(%rbp),%ymm0 |
| 240 | » movups·-0x58(%rbp),%xmm1 | | |
| 241 | » movups·%xmm1,0x10(%rax) | | |
| 242 | » movups·%xmm0,(%rax) | 240 | » vmovups·%ymm0,(%rax) |
| 243 | » mov····%fs:0x28,%rax | 241 | » mov····%fs:0x28,%rax |
| 244 | » mov····-0x8(%rbp),%rcx | 242 | » mov····-0x8(%rbp),%rcx |
| 245 | » cmp····%rcx,%rax | 243 | » cmp····%rcx,%rax |
| 246 | » jne····2391e1·<sinf@@Base-0xeea5f> | 244 | » jne····23927e·<sinf@@Base-0xef712> |
| 247 | » mov····-0xa0(%rbp),%rax | 245 | » mov····-0xa0(%rbp),%rax |
| 248 | » add····$0xb0,%rsp | 246 | » add····$0xb0,%rsp |
| 249 | » pop····%rbp | 247 | » pop····%rbp |
| | | 248 | » vzeroupper· |
| 250 | » retq | 249 | » retq |
| | | 250 | » vzeroupper· |
| 251 | » call···332820·<__stack_chk_fail@plt> | 251 | » call···333c40·<__stack_chk_fail@plt> |
| 252 | » cs·nopw·0x0(%rax,%rax,1) | 252 | » cs·nopw·0x0(%rax,%rax,1) |

# Let’s see some code

```scheme
(define (gsl-hwabi psabi)
  (package/inherit gsl
    (name (string-append "gsl-" psabi))
    (arguments
     (substitute-keyword-arguments (package-arguments gsl)
       ((#:make-flags flags #~'())
        #~(append (list (string-append "CFLAGS=-march=" #$psabi)
                        (string-append "CXXFLAGS=-march=" #$psabi))
                  #$flags))
       ((#:configure-flags flags #~'())
        #~(append (list (string-append "--libdir=" #$output
                                       "/lib/glibc-hwcaps/" #$psabi))
                  #$flags))
       ;; The building machine can't necessarily run the code produced.
       ((#:tests? _ #t) #f)
       ((#:phases phases #~%standard-phases)
        #~(modify-phases #$phases
            (add-after 'install 'remove-extra-files
              (lambda _
                (for-each (lambda (dir)
                            (delete-file-recursively (string-append #$output dir)))
                          (list (string-append "/lib/glibc-hwcaps/" #$psabi "/pkgconfig")
                                "/bin" "/include" "/share"))))))))
    (supported-systems (list "x86_64-linux" "powerpc64le-linux"))
    (properties `((hidden? . #t)
                  (tunable? . #f)))))
```

```scheme
;; This copy of gsl will automatically use the libraries that target the
;; x86_64 psABI which the hardware supports.
(define-public gsl-hwcaps
  (package/inherit gsl
    (name "gsl-hwcaps")
    (arguments
     (substitute-keyword-arguments (package-arguments gsl)
       ((#:phases phases #~%standard-phases)
        #~(modify-phases #$phases
            (add-after 'install 'install-optimized-libraries
              (lambda* (#:key inputs outputs #:allow-other-keys)
                (let ((hwcaps "/lib/glibc-hwcaps/"))
                  (for-each
                    (lambda (psabi)
                      (copy-recursively
                        (string-append (assoc-ref inputs (string-append "gsl-" psabi))
                                       hwcaps psabi)
                        (string-append #$output hwcaps psabi)))
                    (list "x86-64-v2" "x86-64-v3" "x86-64-v4")))))))))
    (native-inputs
     (modify-inputs (package-native-inputs gsl)
                    (append (gsl-hwabi "x86-64-v2")
                            (gsl-hwabi "x86-64-v3")
                            (gsl-hwabi "x86-64-v4"))))
    (supported-systems (list "x86_64-linux"))
    (properties `((tunable? . #f)))))
```

```
$ tree /gnu/store/j1w1sdd6f3jz61mv5hcrf98imfn9bgls-gsl-hwcaps-2.7.1/
├── bin
│   ├── gsl-config
│   ├── gsl-histogram
│   └── gsl-randist
├── etc
│   └── ld.so.cache
├── include
│   └── gsl
├── lib
└── share
    ├── aclocal
    │   └── gsl.m4
    ├── doc
    │   └── gsl-hwcaps-2.7.1
    │       └── COPYING
    ├── info
    │   └── gsl-ref.info.gz
    └── man
```

```
$ tree /gnu/store/j1w1sdd6f3jz61mv5hcrf98imfn9bgls-gsl-hwcaps-2.7.1/lib/
├── glibc-hwcaps
│   ├── x86-64-v2
│   │   ├── libgsl.la
│   │   ├── libgsl.so -> libgsl.so.27.0.0
│   │   ├── libgsl.so.27 -> libgsl.so.27.0.0
│   │   └── libgsl.so.27.0.0
│   ├── x86-64-v3
│   │   ├── libgsl.la
│   │   ├── libgsl.so -> libgsl.so.27.0.0
│   │   ├── libgsl.so.27 -> libgsl.so.27.0.0
│   │   └── libgsl.so.27.0.0
│   └── x86-64-v4
├── libgsl.la
├── libgsl.so -> libgsl.so.27.0.0
├── libgsl.so.27 -> libgsl.so.27.0.0
├── libgsl.so.27.0.0
└── pkgconfig
    └── gsl.pc
```

# Using the custom packages

```scheme
(define use-glibc-hwcaps
  (package-input-rewriting/spec
    ;; Replace some packages with ones built targeting custom packages build
    ;; with glibc-hwcaps support.
    `(("gsl" . ,(const gsl-hwcaps))
      ("sdsl-lite" . ,(const sdsl-lite-hwcaps))
      ("seqwish" . ,(const seqwish-hwcaps))
      ("odgi" . ,(const odgi-hwcaps))
      ("wfmash" . ,(const wfmash-hwcaps)))))

(define-public pggb-with-hwcaps
  (package
    (inherit (use-glibc-hwcaps pggb))
    (name "pggb-with-hwcaps")))
```

# Is it worth it?

**Only sometimes**

# Experiments in Performance

Distribution focused analysis

## x86-64-v3: Mixed Bag of Performance

The pursuit of performance has long been sought after by advanced users, from custom compiling select packages to building your whole system from source. The inclusion of new variations to the default `x86_64` in the psABI has seen this extended to the distribution level, where some distributions are looking (or in the process) to include higher levels such as `x86-64-v2` or `x86-64-v3`. The belief in the performance improvements have even resulted in new distributions being created to maximize performance.

## A Lot of Hype and Marketing

One of the newer entries into the performance arena is CachyOS, which rebuilds some of the Arch Linux repos with `x86-64-v3` (plus some further modifications to performance sensitive packages). There are other modifications as well (which you can read on their website), but for this we are only interested in the impact of providing these packages **without** any of the other changes.

On the wiki it states that "if x86-64-v3 is detected it will automatically use the optimized packages, which yields more than 10% performance improvement." This is a very powerful statement...but is it true? In fairness, they are telling new users on discord that not all packages will benefit from using `x86-64-v3`, but adding caveats doesn't make it sound nearly as good. Do package optimizations bring better performance across the board?

> Inspecting the CachyOS packages, it appears they are setting `-march=x86-64-v3 -mpclmul -O3` vs `-march=x86-64 -O2` (other flags being the upstream Arch Linux defaults) in Arch Linux, so we are testing wider optimizations than just switching to `x86-64-v3`. There are also some instances of `-mtune=skylake` which may have been set at one stage. Increasing the `-march` level and optimization levels are usually touted as the easiest ways to improve performance.

## bzip2

| | Arch – Time | Power | CachyOS – Time | Power |
|---|---|---|---|---|
| Compress Kernel (-3) | 47.01 | 689.8 | -2.5% | -1.5% |
| Compress Kernel (-9) | 49.80 | 736.8 | -1.5% | -1.4% |
| Decompress Kernel | 15.31 | 225.5 | 7.1% | 6.9% |

### flac

| | Arch – Time | Power | CachyOS – Time | Power |
|---|---|---|---|---|
| Decode | 0.71 | 10.5 | -10.1% | -12.1% |
| Encode (-3) | 0.90 | 13.3 | -15.0% | -12.4% |
| Encode (-8) | 2.95 | 49.1 | -20.2% | -23.4% |

To no surprise `flac` benefits a lot from optimization. The source code already includes AVX2 runtime functions to improve the performance without requiring it to be enabled via the `-march` flag. Quite an improvement given it will already use AVX2 on the Arch build.

## gawk

| | Arch – Time | Power | CachyOS – Time | Power |
|---|---|---|---|---|
| Gawk | 2.88 | 42.9 | -2.3% | -0.7% |

Not a commonly thought about package for benchmarking and only a slim improvement.

## gzip

| | Arch – Time | Power | CachyOS – Time | Power |
|---|---|---|---|---|
| Compress Kernel (-3) | 10.33 | 152.1 | -9.5% | -10.0% |
| Compress Kernel (-9) | 35.03 | 501.6 | -2.9% | -5.8% |
| Decompress Kernel | 3.49 | 50.2 | -0.7% | -1.6% |

## lz4

| | Arch – Time | Power | CachyOS – Time | Power |
|---|---|---|---|---|
| Compress Kernel (–best) | 44.60 | 642.7 | 1.6% | 10.5% |
| Compress Kernel (-5) | 9.70 | 141.7 | 2.9% | 10.2% |
| Decompress Kernel | 0.73 | 12.1 | -5.4% | -2.9% |

## python

| | Arch – Time | Power | CachyOS – Time | Power |
|---|---|---|---|---|
| Pybench | 1472 | 290 | 3% | 3.6% |

My understanding is that the CachyOS package also includes BOLT optimization on top of `x86-64-v3` (which is said by upstream to improve performance by a % or two on such a benchmark). Even with additional optimizations it still lagged behind performance wise.

### r

| | Arch – Time | Power | CachyOS – Time | Power |
|---|---|---|---|---|
| R-benchmark-25 | 39.41 | 2992.5 | 0% | 0.4% |

CachyOS doesn’t rebuild the `r` package, but it does rebuild blas and lapack. According to `perf`, the test spends 86% of the time in `blas`, yet we still see no benefit. But, the proper way to actually improve performance here is to use `openblas` instead.

### vorbis

| | Arch – Time | Power | CachyOS – Time | Power |
|---|---|---|---|---|
| Decode | 2.22 | 33.6 | -10.7% | -8.4% |
| Encode (-b 128) | 6.48 | 99.2 | -20.8% | -19.5% |
| Encode (-q 10) | 8.48 | 129.4 | -14.7% | -12.8% |

## XZ

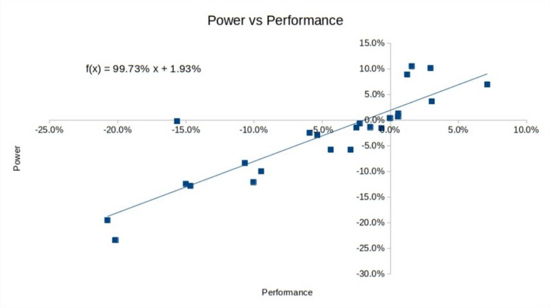
| | Arch – Time | Power | CachyOS – Time | Power |
|---|---|---|---|---|
| Compress Kernel (-3 -T1) | 75.03 | 1149.5 | 0.6% | 0.7% |
| Compress Kernel (-9 -T4) | 95.63 | 2638.2 | 1.2% | 8.9% |
| Decompress Kernel | 5.76 | 88.3 | 0.6% | 1.3% |

### zstd

| | Arch – Time | Power | CachyOS – Time | Power |
|---|---|---|---|---|
| Compress Kernel (-19 -T4) | 77.64 | 2443.8 | -4.4% | -5.7% |
| Compress Kernel (-8 -T1) | 11.09 | 181.8 | -5.9% | -2.5% |
| Decompress Kernel | 0.99 | 16.4 | -15.7% | -0.2% |

## A Mixed Bag, But a Win Overall

Optimized packages can provide considerable advantages over their generic `x86-64` counterparts for many packages. But where software doesn't see much benefit from these newer instructions, we are often left with worse performance and higher power consumption. This makes it a complex question over whether it's worthwhile to build every package with greater optimizations.




This graph shows that overall the winners were bigger than the regressions. However, this post was intended to be more about `x86-64-v3`, but some quick tests (which requires further analysis) suggest that CachyOS using `-O3` is what's actually responsible for some of the larger gains rather than `x86-64-v3`.

# Thanks

**Q & A**