Is Guile fast?

The effective speed of GNU Guile depends on the task. It can be more than factor 10 faster than Python (i.e. for math with huge numbers) or slower (string manipulation from files which in Python is directly handed off to C-code).

In general, Guile is among the faster Schemes, but performance of Schemes varies strongly by task. A performance-comparison of different Schemes over a wide variety of tasks is available in the r7rs-benchmarks by ecraven.

Using geometric mean to obtain one number shows that Guile is roughly factor 2.3 slower than Racket, but as said, this varies by task.

I once got out of performance problems by just translating a Python program into a Guile one using exact numbers. The performance of Guile for huge numbers is awesome!

To get an idea how this compares to other programming languages, you can have a look at the results from the benchmarksgame which includes Racket: In the central 50% of tests (the inter-quartile-range, IQR) Racket is on average factor 5-10 slower than C (though with code golf and unsafe ops it got down to 2x C-without-assembler in the spectral norm), so Guile could be factor 10-20 slower than C, while Python is factor 20-200 slower than C.

If you need maximum performance, you'll want to externalize performance-critical parts to C and access them via FFI or use existing bindings like guile-ffi-cblas.

Benchmarking Guile Versions with the R7RS Benchmarks

To test changes in speed between different guile Versions, I use the R7RS benchmarks by ecraven as well as my own evaluation to get the geometric mean of the slowdown compared to the fastest. How to reproduce:

```
hg clone https://hg.sr.ht/~arnebab/wisp # needs https://mercurial-scm.org
git clone https://git.savannah.gnu.org/git/guile.git # needs https://git-scm.com
git clone https://github.com/ecraven/r7rs-benchmarks
cd guile && git checkout main # replace main with the revision to test
guix environment guile # opens a shell, needs to run on https://guix.gnu.org
guix shell gperf # needed to build guile from shell
./autogen.sh && ./configure --prefix=$HOME/.local && make
cd ../r7rs-benchmarks
rm results.Guile
```

```
VERSION=3
guix shell guile -- bash -c \
  'GUILD=../guile/meta/guild GUILE=../guile/meta/guile ./bench guile all'
sed -i s/guile-/guile-${VERSION}-/g results.Guile
sed -i s/guile,/guile-${VERSION},/g results.Guile
mv results.Guile results.Guile${VERSION}
# repeat for other versions of Guile (git checkout + VERSION=...)
grep CSVLINE results.Guile* | sed 's,+!CSVLINE!+,,' >  /tmp/all.csv
cd ../wisp/examples
for i in 3; do ./evaluate-r7rs-benchmark.w /tmp/all.csv guile-$i; done
```

An example run:

```
for i in 2 3 jit nojit; do
  ./evaluate-r7rs-benchmark.w /tmp/all.csv guile-$i 2>/dev/null
done | grep -A2 "Geometric Mean" 2>/dev/null
```

Results of just one run (these are no representative statistics!):

```
=== Guile-2 Geometric Mean slowdown (successful tests / total tests) ===

2.8389855923409266 (56 / 57)

=== Guile-3 Geometric Mean slowdown (successful tests / total tests) ===

1.395836097066007 (56 / 57)

=== Guile-Jit Geometric Mean slowdown (successful tests / total tests) ===

1.0074935538381193 (56 / 57)

=== Guile-Nojit Geometric Mean slowdown (successful tests / total tests) ===

3.0961877404441847 (56 / 57)
```

Guile Jit and Guile Nojit are runs with just-in-time compilation forced (Jit:) or disabled (Nojit). See GUILE_JIT_THRESHOLD in the handbook.

(This should not be interpreted as recommendation to always force the JIT. In normal operation letting Guile decide when to JIT-compile should provide a better tradeoff than basing such decisions on synthetic benchmark results)

[2022-11-19 Sa]