Ruby Association 2012 Grant

Creosote

Final Report

Sam Rawlins

December 11, 2012

Targets

The targets for this final report include the following:

1. Up-to-date GMP, MPFR, and Msieve libraries, and extensive documentation for each
2. Ruby bindings for GMP-ECM
3. Ruby bindings for MPC
4. Ruby bindings for MPFRCX and FLINT
5. Creosote, a library allowing various mathematics libraries to be bridged, in Ruby
6. Extra Benchmarking

Effort was made against each of these targets, except for the fourth. Bindings for MPFRCX and FLINT have not been written.

1. Up-to-date GMP, MPFR, and Msieve libraries, and extensive documentation for each

These three projects are grouped together because they are the three projects that I had already started, and largely completed. Bindings for GMP and MPFR are packaged into a single gem, `gmp`. Msieve is packaged in its own gem, `msieve`.

GMP

Additionally, the `gmp` gem has been improved upon during the grant period. Highlights include:

- Implement `GMP::sprintf` (d53c40a). This is discussed in a blog post for the project:

  ```
  GMP.sprintf("%5Zd * %d = %5Zd, %s", z127, 2, z254, "Yay!")
  => "  127 * 2 =   254, Yay!"
  GMP.sprintf("0x%Zx * %d = 0x%Zx, %s", z127, 2, z254, "Yay!")
  => "0x7f * 2 = 0xfe, Yay!"
  ```

  Very exciting. This also represents the very first Ruby code in the `gmp` gem; its the first time I've had resort to falling back to Ruby. `GMP::sprintf` was found to require Ruby 1.9.3's Oniguruma-style Regular Expression syntax. `sprintf` is now explicitly not provided for Ruby 1.8.x.

- extend `GMP::F#to_s` to allow a base to be passed in (5d340a4). This was necessary for the `gnu_mpc` gem tests, allowing for things like:

  ```
  GMP::F.new("0x1921FB54442D18p-51", 53, 16)
  => 0.31415926535897931e+1
  ```

  So $0x1921FB54442D18p - 51$ is an approximation of pi written in Hex Float format.
- Add `Rakefile`! (3be64a5).
- Add some `GMP::F#to_s` tests (3be64a5), (9a1630f).
- Added several new mappings:
- `GMP::Z#gcdext2`, that only calculates g and s (not t) for $as + bt = g$
- `GMP::Z#lcm`, with tests
- `GMP::Z#divisible?` and `GMP::Z#congruent?`
- Added an allocation function for `GMP::Z`, so that `#dup` and `#clone` work now.
- During the grant period, the GMP team announced a new eminent release: GMP 5.1.0. Using the release candidates, I was able to write bindings to the three new functions: `GMP::Z.2fac` (and alias: `GMP::Z.double_fac`), `GMP::Z.mfac`, and `GMP::Z.primorial`

At this point, the `gmp` gem exposes over 70 functions from GMP's Integer Functions interface, over 25 from the Rational Number Functions interface, and over 33 from the Floating-point Functions interface.

In addition, seven Ruby methods expose the Random Number Functions interface.

The `gmp` gem is currently documented with a 31-page manual and rdoc.

MPFR

The MPFR bindings, inside the `gmp` gem, have been improved upon during the grant period. Highlights include:

- implement `GMP::F#integer?` (36f0735).
- MPFR 3.1's PRNG changed; fix tests (092db3c).

At this point the `gmp` gem exposes over 55 functions from MPFR's interface.

The MPFR bindings in the `gmp` gem are currently documented with a 31-page manual and rdoc.

Msieve

No new features were added to the `msieve` gem during the grant period. Only two real changes were made:

- Fixed memory freeing issue that caused a Segmentation Fault.
- Upgraded tests to work under Ruby 1.9.

2. Ruby Bindings for GMP-ECM

Bindings for GMP-ECM were written during the grant period.

The GMP-ECM API consists entirely of one very large struct, `ecm_params` (with 29 members), and one method (`ecm_factor(mpz_t, mpz_t, double, ecm_params)`). `ecm_factor` was bridged to Ruby as `GMP::Z#ecm_factor`. The difficult component of these bindings is supporting all of the fields in `ecm_params`. In total, the parameter types include `mpz_t` (`GMP::Z` in Ruby), `int` (`Fixnum` in Ruby), `double` (`Float` in Ruby), `FILE*` (`IO` in Ruby), `char*` (`String` in Ruby), and `gmp_randstate_t` (`GMP::RandState` in Ruby).

During the grant period, support was written in the bindings for about half of the fields in `ecm_params`, which can be found in the `gmp_ecm` rubygem.

3. Ruby Bindings for MPC

Bindings for GNU's MPC library were written from the ground up, and largely completed during the grant period:

- more than 40 functions from the MPC Complex Numbers interface have been bridged in the `gnu_mpc` gem.
- more than 90% of the methods exposed in the `gnu_mpc` gem are heavily tested. The test suite includes over 160 test examples.
- Approximately 10 functions from the MPC interface have not been bridged.
- The `gnu_mpc` gem has largely been documented in `manual.md`, which gets compiled into a 12-page `manual.pdf` and `manual.html`, using Pandoc (`make` will compile the manual if Pandoc is installed).

4. Ruby bindings for MPFRCX and FLINT

I had hoped to be able to write a few more mathematics C extensions, like MPFRCX ("univariate polynomials over arbitrary precision real (MPFR) or complex (MPC) numbers") and FLINT ("Fast Library for Number Theory"). I was unable during the grant period to write these C extensions.

I was able to evaluate the MPFRCX library in order to guess at how difficult it would be to write such bindings. MPFRCX uses, is inspired by, and is written in the same style as the other GNU mathematics packages (GMP, MPFR, MPC). It should be a straightforward process for me to write bindings in the future.

5. Creosote, a library allowing various mathematics libraries to be bridged, in Ruby

This libaray is largely ready for a public release. At present, the library "knows about" GMP, MPFR, MPC, and Msieve. It can be queried for the latest version of each package. It can also unpack, configure, make, check, and install eacho of these packages. The packages install to `$HOME/.creosote/usr`.

Installing the packages is made useful when we install certain gems and specify specific arguments for `ruby extconf.rb`. This is made possible with the `creosote gem install` command.

Here are some examples:

Basic package install

In this example, the user asks Creoste to `install` the `gmp package` into the default location. Creosote sees that the package has no other requirements, and proceeds with the install, into `~/.creosote/usr`.

```
$ ruby bin/creosote pkg install gmp --default
cd /home/vagrant/.creosote/src
tar -xjf gmp-5.0.5.tar.bz2                                                  [OK]
cd /home/vagrant/.creosote/src/gmp-5.0.5
./configure --prefix=/home/vagrant/.creosote/usr                            [OK]
make clean                                                                  [OK]
make                                                                        [OK]
make check                                                                  [OK]
make install                                                                [OK]
```

Package install with package dependencies

In this example, the user asks Creoste to `install` the `mpc package` into the default location. Creosote sees that the package has other requirements: namely `gmp` and `mpfr`, and installs them first. Then, Creosote proceeds with the install, into `~/.creosote/usr`.

```
$ ruby bin/creosote package install mpc --default
mpfr.h is available in default include paths.                                 [OK]
mpfr library with mpfr_init() is not available in default lib paths.        [FAIL]
gmp.h is available in default include paths.                                  [OK]
gmp library with __gmpz_init() is not available in default lib paths.       [FAIL]
cd /home/vagrant/.creosote/src
tar -xjf gmp-5.0.5.tar.bz2                                                    [OK]
cd /home/vagrant/.creosote/src/gmp-5.0.5
./configure --prefix=/home/vagrant/.creosote/usr                              [OK]
make clean                                                                    [OK]
make                                                                          [OK]
make check                                                                    [OK]
make install                                                                  [OK]
cd /home/vagrant/.creosote/src
downloading ftp.gnu.org/gnu/mpfr//mpfr-3.1.1.tar.bz2.../
tar -xjf mpfr-3.1.1.tar.bz2                                                   [OK]
cd /home/vagrant/.creosote/src/mpfr-3.1.1
./configure --prefix=/home/vagrant/.creosote/usr --with-gmp=/home/vagrant/.creosote/usr
                                                                              [OK]
make clean                                                                    [OK]
make                                                                          [OK]
make check                                                                    [OK]
make install                                                                  [OK]
cd /home/vagrant/.creosote/src
tar -xzf mpc-1.0.1.tar.gz                                                     [OK]
cd /home/vagrant/.creosote/src/mpc-1.0.1
./configure --prefix=/home/vagrant/.creosote/usr --with-mpfr=/home/vagrant/.creosote/usr
                                                                              [OK]
make clean                                                                    [OK]
make                                                                          [OK]
make check                                                                    [OK]
make install                                                                  [OK]
```

Simple Gem Install

Here we use Creosote to install a Rubygem that has known package requirements: gmp. Creosote will detect that the gmp library is not installed, install it to `~/.creosote/usr`, and then install the gmp gem, passing `--with-gmp-dir=/home/vagrant/.creosote/usr` to the `extconf.rb`, so that it may be compiled properly.

This example shows the driving force behind Creosote. With this functionality, Creosote can be a gem that helps Ruby developers quickly manage packages that do not fall under the purview of Rubygems.

```
$ sudo ruby bin/creosote gem install gmp
gmp.h is available in default include paths.                                [OK]
gmp library with __gmpz_init() is not available in default lib paths.     [FAIL]
requirement gmp is not yet installed. Installing...
cd /home/vagrant/.creosote/src
tar -xjf gmp-5.0.5.tar.bz2                                                  [OK]
cd /home/vagrant/.creosote/src/gmp-5.0.5
./configure --prefix=/home/vagrant/.creosote/usr                            [OK]
make clean                                                                  [OK]
make                                                                        [OK]
make check                                                                  [OK]
make install                                                                [OK]
Building native extensions.  This could take a while...
$ gem list |grep gmp
gmp (0.6.7)
```

Gem Install with Dependencies

Here we use Creosote to install a Rubygem that has known package requirements, gem dependencies, *and* the gem dependencies have known package requirements: the gnu_mpc gem. Creosote will detect that the gmp gem is a dependency, and that is not installed. Creosote will then recursively install the gmp gem (and the gmp package, seeing that it, too, is not installed yet). After the dependency is installed, Creosote will install the gnu_mpc gem's requirements (mpc package) to `~/.creosote/usr`, and then install the gnu_mpc gem, passing `--with-mpc-dir=/home/vagrant/.creosote/usr` to the `extconf.rb`, so that it may be compiled properly.

```
$ sudo ruby bin/creosote gem install gnu_mpc
gnu_mpc requires gmp, a gem with known requirements. Installing...
gmp.h is available in default include paths.                              [OK]
gmp library with __gmpz_init() is not available in default lib paths.   [FAIL]
requirement gmp is not yet installed. Installing...
cd /home/vagrant/.creosote/src
tar -xjf gmp-5.0.5.tar.bz2                                                [OK]
cd /home/vagrant/.creosote/src/gmp-5.0.5
./configure --prefix=/home/vagrant/.creosote/usr                          [OK]
make clean                                                                [OK]
make                                                                      [OK]
make check                                                                [OK]
make install                                                              [OK]
Fetching: gmp-0.6.7.gem (100%)
Building native extensions.  This could take a while...
mpc.h is available in default include paths.                              [OK]
mpc library with mpc_init2() is not available in default lib paths.     [FAIL]
requirement mpc is not yet installed. Installing...
mpfr.h is available in default include paths.                             [OK]
mpfr library with mpfr_init() is available in default lib paths.          [OK]
cd /home/vagrant/.creosote/src
tar -xzf mpc-1.0.1.tar.gz                                                 [OK]
cd /home/vagrant/.creosote/src/mpc-1.0.1
./configure --prefix=/home/vagrant/.creosote/usr --with-mpfr=/home/vagrant/.creosote/usr
                                                                          [OK]
make clean                                                                [OK]
make                                                                      [OK]
make check                                                                [OK]
make install                                                              [OK]
Building native extensions.  This could take a while...
$ gem list |grep gnu_mpc
gnu_mpc (0.8.2)
$ gem list |grep gmp
gmp (0.6.7)
```

As creosote grows to help users install the dependant packages, it will also be able to help in bridging individual libraries together.

A release of the creosote gem is eminent. When a user does not encounter an error, it works rather well. As soon as an error occurs though, it can be difficult to recover, and sometimes the user must fix files in `$HOME/.creosote` themselves.

6. Extra Benchmarking

During the grant period, the benchmark tests were completely overhauled, and improved:

- There are now several variants of the benchmarks, to overcome some shortcomings, and to better break out different components. There are now pure Ruby tests, as well as Ruby coupled with gmp gem tests.
- The gcdext test was added. The only thing missing from the reference benchmarks (gmpbench 0.2) is the pi test.
- In the pure Ruby variant of the tests, several methods are not available in Ruby's Bignum API: `#gcd`, `#gcdext`, `#invert`, `#[]=`, and `#powmod`. Ruby implementations of `#gcdext`, `#invert`, and `#powmod` were borrowed from John Nishinaga, available at `https://gist.github.com/2388745`.
- There is an opportunity here to add to Ruby's Bignum library: The modular exponentiation/inverse feature request has been accepted and is currently assigned to Matz. This ticket proposes `Bignum#powmod` and `Bignum#inverse` methods. I may tackle this as a follow-up to this grant.

In addition to improving the benchmarks themselves, a completely new performance report has been written. It is available in the gmp gem's GitHub project: `https://github.com/srawlins/gmp/raw/master/performance.pdf`. This is a 12-page, comprehensive report on the performance of GMP, the gmp gem, and Ruby's Bignum, with specific future ideas on how to improve the performance of the gmp gem, Ruby's Bignum, and how to better run the benchmarks.