theine2 Documentation

Release latest

Dec 06, 2019

Contents

Theine is a Rails application pre-loader designed to work on JRuby. It is similar to Zeus, Spring and Spork. The problem with Zeus and Spring is that they use `fork` which doesn't work on JRuby.

An example:

```
time rails runner "puts Rails.env"
48.31s user 1.96s system 242% cpu 20.748 total  # normal
time theine runner "puts Rails.env"
0.12s  user 0.02s system 32%  cpu 0.449  total  # Theine
```

CHAPTER 1

Installing

You need to install screen on your system. For example on Ubuntu:

```
sudo apt-get install screen
```

Then install the gem.

```
gem install theine2
```

CHAPTER 2

Upgrading

If you want to use CRuby for the client, you will probably need to re-run `theine_set_ruby` after upgrading.

CHAPTER 3

Using

Start up the theine server in the root of your Rails project:

`theine_server`

or

`theine_start` for a detached process

Stopping the server (this only works if the server was started with `theine_start`)

`theine_stop`

You can also add *–debug* mode so that the spawned workers will be running in debug mode (debug mode is ignored if silent is set to true):

`theine_server --debug`

Then run any rails command using theine (don't use bundle exec):

```
#Rails commands
theine runner "puts 'Hello world'"
theine server
theine console

#Rake, rspec, cucumber
theine rake db:migrate
theine rspec spec
theine cucumber
```

CHAPTER 4

Configuration

Theine will look for a `.theine` file in your Rails app's root directory and in your home directory. Both files are loaded if they exist, the one in your Rails app will overwrite settings from your home.

Example `.theine file` (YAML):

```
base_port: 11000
max_port: 11100
min_free_workers: 2
spawn_parallel: true
silent: true
```

4.1 base_port and max_port

The Theine server will use the base_port TCP port. Workers use base_port + 1 to max_port TCP ports. Use this setting if you need to run multiple Rails apps at the same time (use a different base_port for each).

4.2 min_free_workers

The minimum amount of workers that Theine should try to keep around. If you set this to 2 for example, once Theine has loaded 2 workers, you can execute two commands immediately. When you execute a third command, you will have to wait until Theine has spawned new workers. Theine will spawn a new worker as soon as you execute a command.

4.3 spawn_parallel

When set to *true*, Theine will start min_free_workers all at once. When set to *false*, it will start one worker first, and when it is loaded, it will start the next worker. When *false*, the first worker should start just a little bit faster. If you have a high number of min_free_workers then I recommend setting this to *false*.

4.4 silent

When set to *true*, Theine will start in silent mode. This mode gives no output to the shell and is perfect for using *thiene_server &* for a detached process. This is especially helpful in docker environments when you cannot have multiple shells open without tmux. recommend *false* for non docker workflows.

CHAPTER 5

Speed up Theine

If you are using RVM or rbenv, you can tell theine to use Ruby 2.6 to run the client, which will make the Theine client start much faster.

If you have any problems in this mode, please revert back to using the same Ruby for the client.

5.1 Tell Theine to use CRuby for the client

Figure out where your Ruby 2.6.5 binary is:

```
$ rvm ruby-2.6.5 do which ruby
/Users/mrbrdo/.rvm/rubies/ruby-2.6.5-p114/bin/ruby
```

Then where you plan to use Theine, do:

```
$ rvm use jruby
$ theine_set_ruby /Users/mrbrdo/.rvm/rubies/ruby-2.6.5-p114/bin/ruby
```

Enjoy mega fast client:

```
time theine runner "puts 'hello world'"
0.12s user 0.02s system 30% cpu 0.470 total
```

CHAPTER 6

Using with Foreman

Theine works with Foreman:

```
theine_server: theine_server
server: rails server
```

If you have problems, try adding theine to your Gemfile. If you want to use the theine client in foreman, you should use theine_current_ruby because Foreman uses bundle exec. But there is no point in doing that, since theine needs to spawn a process for each command anyway, so there is no benefit in comparison to just running the command (like rails server) directly.

CHAPTER 7

Using with Docker

See docker_workflow

CHAPTER 8

How it works

Theine's server spawns processes in the background that load your Rails application. When you run a command through theine, it will be executed in one of these pre-loaded processes. I used to do IO redirection (similarly to pry-remote) but it ended up being very unreliable, so now I am using screen to take care of this. After your command is done, the process will exit. When you run a new command, it will run in another pre-loaded process.

Theine will automatically spawn additional processes as needed.

The client (*theine* command) does not need to run on JRuby (or the same Ruby that you use in your Rails application), because it is only used to connect to the server, all the code is then actually executed on the server.