Image Optimization in Rails

For an up-to-date version and discussions see
https://github.com/consti/image-optimization-howto

Constantin Hofstetter

November 21, 2014

Reduce file size of static assets

ImageOptim

Reduce image file sizes significantly by using multiple optimization (lossless compression) libraries packaged into one tool. Available as:

- GUI Application
- CLI Application
- Extension for Middleman

ImageAlpha

Convert PNG24 images with transparent background to PNG8+alpha images to reduce file size. Available here.

Reduce file size of static assets

ImageOptim CLI as pre-commit hook

Automatically optimize images that are committed to your git repo:

```
# in your_project/.git/hooks/pre-commit
images=$(git diff --exit-code --cached --name-only --diff-
    filter=ACM -- '*.png' '*.jpg')
$(exit $?) || echo $images | imageoptim && git add $images
```

Paperclip integration

Using `imagemagick` parameters

Add options to the `convert_options` parameter:

- `strip`: remove EXIF data
- `quality PERCENTAGE`: compress image data (not lossless)
- `interlace METHOD`: make the image progressive

Example:

```
# In your model
has_attached_file :avatar, {
  styles: { thumb: '60x60#' },
  convert_options: {
    all: '-strip -quality 70 -interlace Line'
  }
}
```

Paperclip integration

Using `paperclip-optimizer` processor

The `paperclip-optimizer` gem uses the `image_optim` library instead of `imagemagick` to yield better optimization results.

Paperclip integration

Using `paperclip-optimizer` processor

```
# Add paperclip-optimizer gem
# Install image_optim binaries / image_optim_bin gem (HEROKU)
has_attached_file :avatar, {
  styles: {
    thumb: {
      geometry: '60x60#',
      paperclip_optimizer: {
        # add compression (not lossless) to optimization (
            lossless)
        jpegoptim: { max_quality: 70 },
        # Disable unavailable binaries
        svgo: false
      }
    }
  },
  # We still need to strip EXIF and make the image progressive
  convert_options: { all: '-strip -interlace Line' },
  processors: [:paperclip_optimizer]
}
```

Paperclip integration

Multiple geometries

Make sure to include the `:thumbnail` processor if you want to save your image in different geometries.

```
# in your model
has_attached_file :avatar, {
  ...
  processors: [:thumbnail, :paperclip_optimizer]
}
```

Image delivery

Store images on file storage web service

Use a file storage web service such as Amazon Simple Storage Service (S3) to store image uploads and static assets.

Use a CDN

Content Delivery Network: images and static assets are served from the closest edge server, based on the clients location, e.g. Amazon CloudFront

Allow Clients to cache images and static assets

Use `Cache-Control` or `Expires` headers.

S3: Paperclip

```
# in your Gemfile
gem 'aws-sdk'

# in your Paperclip config:
PAPERCLIP_STORAGE_OPTIONS = {
  :storage => :s3,
  :s3_host_name => 'REMOVE_THIS_LINE_IF_UNNECESSARY',
  :s3_credentials => {
    :bucket => 'S3_BUCKET_NAME'
  }
}

# in your aws.yml
development:
  access_key_id: AWS_ACCESS_KEY_ID
  secret_access_key: AWS_SECRET_KEY_ID

production:
  access_key_id: AWS_ACCESS_KEY_ID
  secret_access_key: AWS_SECRET_KEY_ID
```

S3: Static Assets

Add `asset_sync` to your Gemfile.

```
# In your asset_sync.rb
s3_config = YAML.load_file("#{Rails.root}/config/s3.yml")[Rails
    .env]

config.aws_access_key_id = s3_config['access_key_id']
config.aws_secret_access_key = s3_config['secret_access_key']
config.fog_directory = s3_config['bucket']
# config.fog_region = 'eu-west-1'
config.gzip_compression = true
```

Then run:

```
# RAILS_GROUPS=assets will make asset sync run in verbose mode
bundle exec rake assets:precompile RAILS_GROUPS=assets
```

Expiration

Expires or Cache-Control?

Expires is a fixed future date, Cache-Control is the amount of time (in seconds) the client is allowed to cache the object.

Expires or Cache-Control for CloudFront & S3

CloudFront accepts both values `Expires` and `Cache-Control`. If you specify values both for `Cache-Control` and for `Expires`, CloudFront uses only the value of `Cache-Control`.

`Cache-Control` is the recommended way to set expiration.

Expiration: Paperclip

Cache-Control in Paperclip

Include `Cache-Control` in your `s3_headers`-Paperclip config:

```
# in your Paperclip config:
PAPERCLIP_STORAGE_OPTIONS = {
  ...,
  s3_headers: lambda { |_attachment|
    {
      'Cache-Control' => "public, max-age=#{ 1.month.to_i }"
    }
  }
}
```

Expiration: Static Assets

```
# in your asset_sync.rb
config.custom_headers = {
  # for fonts and svg, you have to set Access-Control-Allow-
      Origin
  '.*\.[svg|woff|ttf|eot]' => {
    'Access-Control-Allow-Origin' => '*',
    cache_control: "public, max-age=#{ 1.month.to_i }",
    expires: nil
  },
  # Everything else, set cache-control to max-age 1 month
  '.*' => {
    cache_control: "public, max-age=#{ 1.month.to_i }",
    expires: nil
  }
}
```

Domain Sharding

Clients have a max-limit of connections to a single host (e.g. modern browsers up to ten connections).
Domain Sharding helps by splitting images and static assets onto different hosts (i.e. domains that point to the same resource).

SPDY and HTTP 2.0

SPDY and HTTP 2.0 (which integrates SPDY) allow clients to keep connections open to stream files.
Domain Sharding will actually hurt the performance of HTTP 2.0 and SPDY compatible browsers (unnecessary domain lookups).

Domain Sharding

Create different CloudFront hosts that point to the same S3 bucket (where you store images/static assets).

```
# in your app config (e.g. production.rb):
config.assets.cdn_hosts = %w(xxxxxx01.cloudfront.net
                             xxxxxx02.cloudfront.net
                             xxxxxx03.cloudfront.net)
```

Domain Sharding: Paperclip

We calculate the asset host through the `update_at` in seconds of the file (to make sure the file always gets served from the same host).

```
# in Paperclip config:
PAPERCLIP_STORAGE_OPTIONS = {
  ...,
  s3_host_alias: lambda { |attachment|
    Rails.configuration.assets.cdn_hosts[
      attachment.updated_at.to_i % Rails.configuration.assets.
          cdn_hosts.count
    ]
  },
  url: ':s3_alias_url'
}
```

Domain Sharding: Static Assets

We calculate the asset host through the MD5 sum of the file (to make sure the file always gets served from the same host).

```
# in your app config (e.g. production.rb):
config.action_controller.asset_host = Proc.new { |source|
  'https://' + Rails.configuration.assets.cdn_hosts[
    Digest::MD5.hexdigest(source).to_i(16) % Rails.
        configuration.assets.cdn_hosts.count
  ]
}
```