FreeBSD Administration - Support #496

Installing Redis Server and PHP/Ruby Caching

12/15/2014 11:35 PM - Daniel Curtis

Status:	Closed	**Start date:**	12/15/2014
Priority:	Normal	**Due date:**	
Assignee:	Daniel Curtis	**% Done:**	100%
Category:	Caching Server	**Estimated time:**	0.60 hour
Target version:	FreeBSD 9	**Spent time:**	1.50 hour

Description

Installing

- Install redis FreeBSD port collection:

```
portsnap fetch extract
cd /usr/ports/databases/redis
make install clean
```

- Or with portmaster

```
portmaster databases/redis
```

- Or pkg2ng

```
pkg install redis
```

Redis configuration file

- Editing the redis config file:

```
vi /usr/local/etc/redis.conf
```

Auto-start Redis at Boot

- To run redis from startup:

```
echo 'redis_enable="YES"' >> /etc/rc.conf
```

Bind Redis to IP**

- To bind Redis to a single interface, add the following line to /usr/local/etc/redis.conf:

```
#bind 127.0.0.1
bind 192.168.0.1
```

Commands

- Stop/Start Redis

```
service redis stop
```

```
service redis start
```

- Accessing Redis

```
telnet localhost 6397
```

- Or the Redis CLI client

```
redis-cli
```

PHP5-Extension for Redis

- Install php5-redis extension from the ports tree

```
cd /usr/ports/databases/php5-redis/ && make install clean
```

- Or portmaster

```
portmaster databases/php5-redis
```

- Or pkg2ng

```
pkg install php5-redis
```

PHP Session handler

php5-redis can be used to store PHP sessions. To do this, configure session.save_handler and session.save_path in php.ini to tell php5-redis where to store the sessions:

```
session.save_handler = redis
session.save_path = "tcp://host1:6379?weight=1, tcp://host2:6379?weight=2&timeout=2.5, tcp://host3
:6379?weight=2"
```

session.save_path can have a simple host:port format too, but you need to provide the tcp:// scheme if you want to use the parameters.

The following parameters are available:

1. **weight** (*integer*): the weight of a host is used in comparison with the others in order to customize the session distribution on several hosts. If host A has twice the weight of host B, it will get twice the amount of sessions. In the example, host1 stores 20% of all the sessions (1/(1+2+2)) while host2 and host3 each store 40% (2/1+2+2). The target host is determined once and for all at the start of the session, and doesn't change. The default weight is 1.
2. **timeout** (*float*): the connection timeout to a redis host, expressed in seconds. If the host is unreachable in that amount of time, the session storage will be unavailable for the client. The default timeout is very high (86400 seconds).
3. **persistent** (*integer*, should be 1 or 0): defines if a persistent connection should be used. **(experimental setting)**
4. **prefix** (*string*, defaults to "PHPREDIS_SESSION:"): used as a prefix to the Redis key in which the session is stored. The key is composed of the prefix followed by the session ID.
5. **auth** (*string*, empty by default): used to authenticate with the server prior to sending commands.
6. **database** (*integer*): selects a different database.

Sessions have a lifetime expressed in seconds and stored in the INI variable "session.gc_maxlifetime". You can change it with ini_set(). The session handler requires a version of Redis with the SETEX command (at least 2.0). php5-redis can also connect to a unix domain socket: session.save_path = "unix:///var/run/redis/redis.sock?persistent=1&weight=1&database=0.

Ruby & Rails Session Handler

- Install the redis-rails gem:

```
pkg install rubygem-redis-rails
```

- Gemfile Installation

```
gem 'redis-rails'
```

- Usage in config/application.rb:

```
config.cache_store = :redis_store, 'redis://localhost:6379/0/cache', { expires_in: 90.minutes
}
```

- Configuration values at the end are optional. If you want to use Redis as a backend for sessions, you will also need to set in config/initializers/session_store.rb

```
MyApplication::Application.config.session_store :redis_store
```

- And if you would like to use Redis as a rack-cache backend for HTTP caching, config/environments/production.rb

```
config.action_dispatch.rack_cache = {
  metastore:   'redis://localhost:6379/1/metastore',
  entitystore: 'redis://localhost:6379/1/entitystore'
}
```

Then you're good to go. Your cache store should now be using Redis.

Resources

- http://jonlabelle.com/snippets/view/markdown/redis-and-freebsd
- https://github.com/nicolasff/phpredis
- http://jimneath.org/2011/03/24/using-redis-with-ruby-on-rails.html

History

#1 - 12/18/2014 10:44 PM - Daniel Curtis

- Description updated

- Status changed from New to Resolved

- % Done changed from 0 to 100

#2 - 12/26/2014 07:17 PM - Daniel Curtis

- Status changed from Resolved to Closed

#3 - 02/14/2015 10:44 AM - Daniel Curtis

- Target version set to FreeBSD 9

#4 - 02/14/2015 12:02 PM - Daniel Curtis

- Category set to Caching Server