Implement Dynamic Pools & Circuit Breaker

Personal information

Name: Eugene Mironov
Geo: Russia, Chelyabinsk, GMT/UTC + 05:00

I finish my master's degree in Chelyabinsk State University. faculty is Institute of Information Technologies.

Contact information

Email: helper2424@gmail.com
Phone number: +79068616390
IRC Handle: helper2424
GitHub: helper2424
Personal blog: All interesting events I post in vk https://vk.com/std_x01d (russian social network) or fb https://www.facebook.com/helper.helperov
GitHub: helper2424
Twitter: https://twitter.com/helper2424
StackOverflow: http://stackoverflow.com/users/4711095/helper-helperov
LinkedIn: https://www.linkedin.com/in/helper2424?trk=hp-identity-photo
Google+: https://plus.google.com/u/0/111620516305092059109

Background:

I have 8 years of programming experience, 6 of these are with production projects. Php,

pyhon, scala, js were used in little home projects or in university labs. I have used ruby, java, c++ in production projects. Ruby on Rails is for web applications, c++ and java are for real time games backends. I have big experience with development multithreading applications on c++. Also I often use jruby for development assistant tools: benchmarking, testing.

Project

I plan to implement Dynamic Pools system of actors and Circuit Breaker pattern for Celluloid project. Ideas are described here: https://github.com/celluloid/GSoC/wiki/Idea:-Dynamic-Pools and https://github.com/celluloid/GSoC/wiki/Idea:-Circuit-Breaker. Celluloid is a ruby multithreading library based on Actor model.

Dynamic Pools

The project essence is about creating extension for library https://github.com/celluloid/celluloid-pool . Now library is allowing to create static pool of actors. It's mean that user sets size of pool when creating it. It's necessary to have a library which can adapt to task, and increase or decrease actors pool size depending on busy of actors and abilities of system.

What is value? Ruby has several big problems:

1. CPU performance
2. A lot of memory requirements

So, dynamic pool allows create actors depending to task and to reuse actors which are released. It are decreases count of actors, so decreases memory usage and cpu usage to switch between idle actors.

Behavior of dynamic pool is similar to Java Thread Pool Executor https://docs.oracle.com/javase/7/docs/api/java/util/concurrent/ThreadPoolExecutor.html . From there i will implement mechanisms of allocating and destroying actors and copy api methods:

set_core_pool_size
set_core_pool_size
set_max_core_pool_size
get_max_core_pool_size
prestart_actor
prestart_all_actor
get_queue

Circuit breaker

The project essence is implementation of Circuit Breaker pattern for Celluloid library.

If you use Celluloid in complex distributed systems, inevitably some of the many service dependencies will be failed. If the host application is not isolated from these external failures, it

risks being taken down with them. So, we need mechanism to control the interactions between dependencies, to prevent cascading failures.

To resolve this issue need just implement Circuit Breaker pattern, which allows wrap critical methods and will trying to call them later.

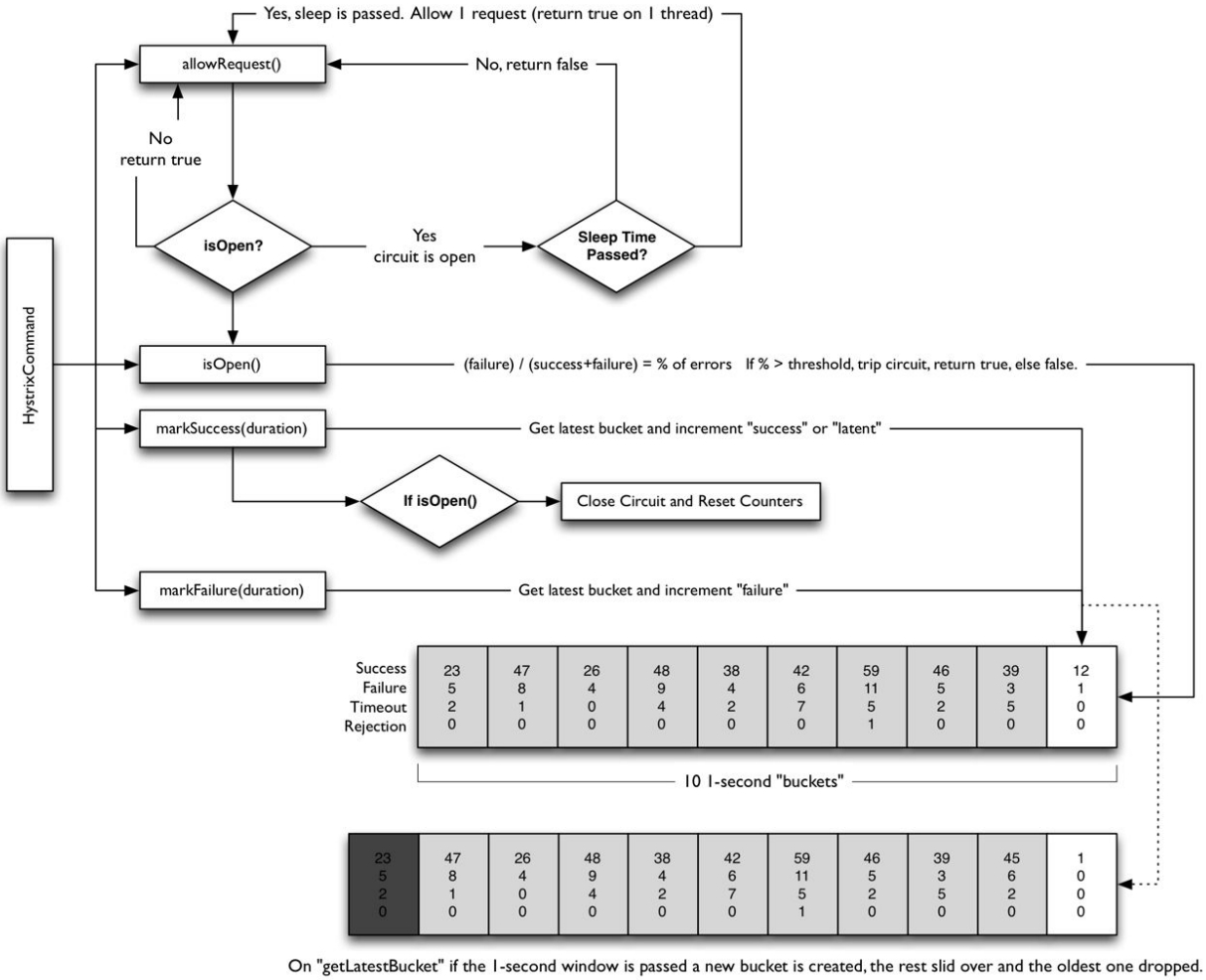
I see implementation like class extension for Actor class, which wraps calls by Actor class methods in begin/rescue block. It's allow catch exceptions and start circuit breaker logic. All failures and success calls will saved in internals metrics.

Also I will implement CircuitBreakerException. All CircuitBreakerException exceptions and Its inheritors will be ignored by circuit breaker try/rescue block.

Api methods:

get_fallback
set_timeout
set_repeat
...

Schema of circuit breaker states switches:



Hystrix Circuit breaker schema, gotten here https://github.com/Netflix/Hystrix/wiki/How-it-Works#circuit-breaker

Roadmap

Week 1 - 2

Research code of Celluloid projects https://github.com/celluloid
And the Hystrix project https://github.com/Netflix/Hystrix

Week 3 - 5

Implementation of Dynamic Pool system

- Create benchmarks of existing static-pool implementation to compare to future results.
- Write test units which fail until this new implementation is complete.
- Begin including and adapting the functionality found in the Java Thread Pool Executor.
- Expand API until all failing tests show 100% coverage, and all pass.
- Perform new benchmarks with dynamic-pools, and compare to previous.

MIDTERM

Week 6 - 9

Implementation of Circuit Breaker

- Create sets of tests to cover major use cases.
- Implement Exception system for Circuit Breaker.
- Implement internal of class wrapper:
    - Timer
    - Success/fail metrics
    - Method wrapper
    - Circuit breaker logic
- Expand API until all failing tests show 100% coverage, and all pass.

Week 10

Tests, Benchmarks, Documentation.

Week 11

Buffer Period.

Week 12 - 13

Holidays

Final Evaluation

If I overtake plan I would like to make benchmarking and optimize some parts of celluloid projects.

References:

[1] Dynamic Pool idea
https://github.com/celluloid/GSoC/wiki/Idea:-Dynamic-Pools

[2] Circuit breaker idea
https://github.com/celluloid/GSoC/wiki/Idea:-Circuit-Breaker

[3] Java Thread Pool Executor
https://docs.oracle.com/javase/7/docs/api/java/util/concurrent/ThreadPoolExecutor.html