Protect your users with Circuit Breakers

Scott Triglia
Europython 2016
Yelp’s Mission:
Connecting people with great local businesses.
Yelp Stats
As of Q1 2016

- 90M
- 102M
- 70%
- 32
Scott Triglia

@scott_triglia

Work with the $$$
Let’s talk Circuit Breakers
Python Process

Calling Code

Local Python Function
def block_request():
    return percentage_failed() > 0.05
Our goals today: introduce a basic circuit breaker
Our goals today: a modular circuit breaker
Our goals today: test it out on several scenarios
the fundamental rule:

your systems will fail

what’s your response?
Nygard’s circuit breaker
Process 1 → Req → SQS Client → Amazon SQS → Task Submission → Worker → Process 2 → Req
Circuit Breaker States:

* **Healthy** (or “closed”)
* **Recovering** (or “half-open”)
* **Unhealthy** (or “open”)
is_healthy = self.assess_if_healthy()
if is_healthy == HEALTHY:
    return self.issue_request()
elif is_healthy == RECOVERING:
    return issue_trial_request()
else:
    raise RequestBlockedError()
Recovery:

* Wait for `recovery_timeout` seconds
* Send a trial request, trust its results
Before a circuit breaker:
Before a circuit breaker:

* Diners wait forever to get food
Before a circuit breaker:
* Diners wait forever to get food
* Kitchen has a growing backlog
Before a circuit breaker:
* Diners wait forever to get food
* Kitchen has a growing backlog
* New diners making things worse
With a circuit breaker:
With a circuit breaker:

* Fewer frustrated users
With a circuit breaker:

* Fewer frustrated users
* Reduced load on the backend
With a circuit breaker:
* Fewer frustrated users
* Reduced load on the backend
* A well defined failure mode
Module 1:

Should our waiters all agree?
New Behavior:
* Clients inform each other
* Processes are no longer independent
* Propagate failure faster
* Requires distributed datastore
* Forces decisions about consistency
Module 2:

What should we do in response?
is_healthy = self.assess_if_healthy()

if is_healthy == HEALTHY:
    return self.issue_request()

elif is_healthy == RECOVERING:
    return issue_trial_request()

else:
    raise RequestBlockedError()
New Behavior:

* Code can check in advance about healthiness of system

* Automatic monitoring!
* Build features on top of system health status

* Requires a single source of truth?
Module 3:

Who decides we’re unhealthy?
def signal_overload(cb):
    if len(jobs) > THRESH:
        cb.mark_unhealthy()
New Behavior:

* CB gets signals from anywhere
* Signal combining logic
* Allows many (many) new signals

* Must combine signals

* Adds complexity to system
Module 4:

How do we recover?
Dark launch:

* Reject but process normally
* Dangerous with side effects

Block User Request  Try to process anyway!
Synthetic:

* Dark launching with fake requests
* Not necessarily representative
New Behavior:

* Traffic determines health
* Removal of recovery timeouts
* Faster(?) recovery
* No timeout tuning required
* Dark launching not always possible
* Synthetic can be unrepresentative
in summary
Your system will fail, have a plan!
The basic CB is better than nothing
Questions to ask:

* Should our waiters all agree?
* How should I deal with unhealthiness?
* Who decides we’re unhealthy?
* How do we recover?
Questions to ask:

* Should our waiters all agree?
* How should I deal with unhealthiness?
* Who decides we’re unhealthy?
* How do we recover?
Questions to ask:

* Should our waiters all agree?
* How should I deal with unhealthiness?
* Who decides we’re unhealthy?
* How do we recover?
Questions to ask:

* Should our waiters all agree?
* How should I deal with unhealthiness?
* Who decides we’re unhealthy?
* How do we recover?
Questions to ask:

* Should our waiters all agree?
* How should I deal with unhealthiness?
* Who decides we’re unhealthy?
* How do we recover?
...and much more!
Much comes down to your use case
Questions?

striglia@yelp.com
@scott_triglia
Can’t we do better than rejecting requests?
How do I safely test out a new circuit breaker?