Resilient Javascript From Front to Back End With Circuit Breakers

Lance Ball
Principal Software Engineer
https://lanceball.com
Twitter: @lanceball
GitHub: @lance

Riviera Dev 2018
Thursday, May 17 2018

Resilience

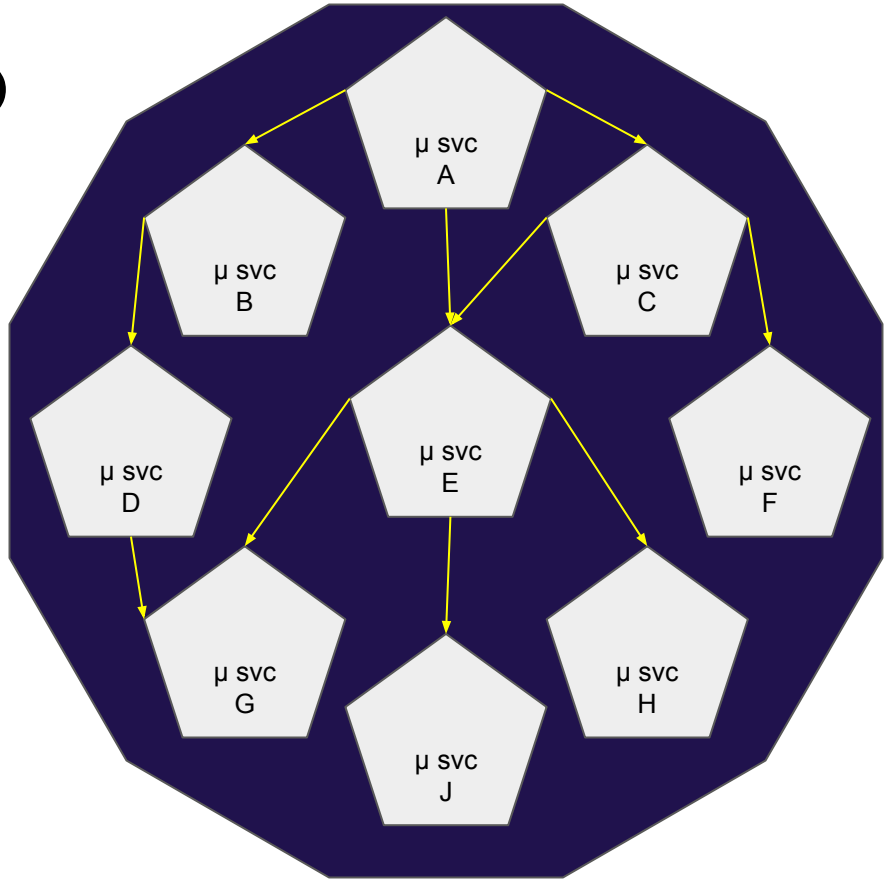
Resiliency is defined as the capability of a system to maintain its functions and structure in the face of internal and external change and to degrade gracefully when it must.

TOWARD INHERENTLY SECURE AND RESILIENT SOCIETIES
Brad Allenby, Jonathan Fink

http://science.sciencemag.org/content/309/5737/1034.full

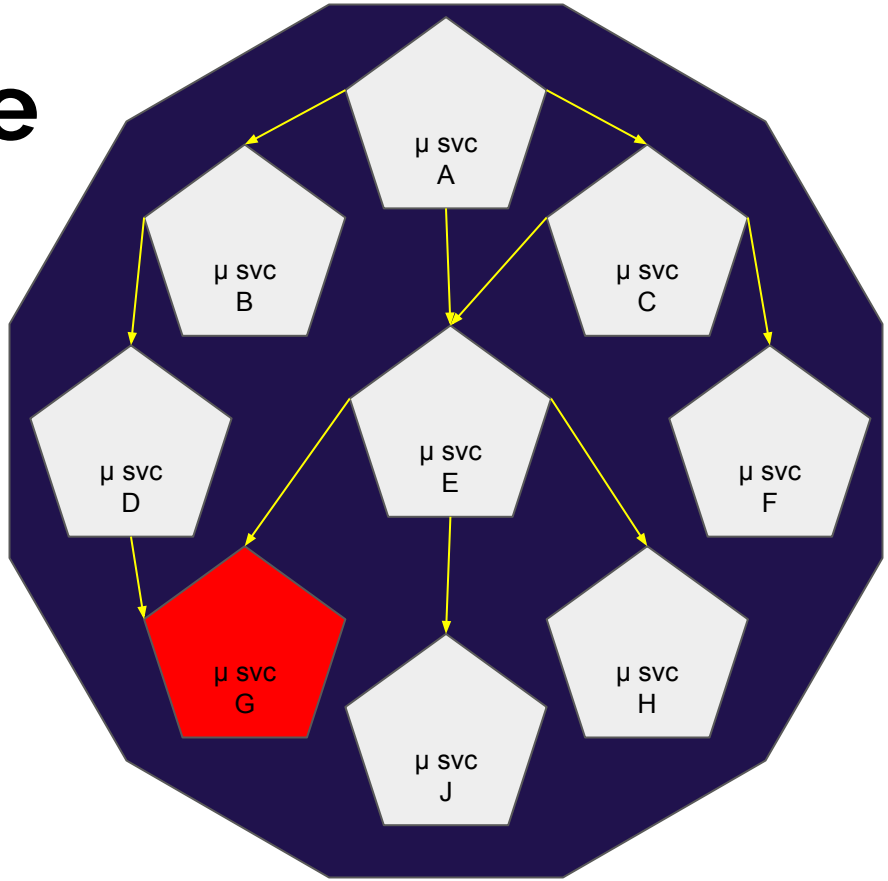
Microservices

My App



Microservices are not a panacea

A Single Failure



μ-Service D

```
function wait (timeout) {
 return new Promise(resolve => {
   setTimeout(resolve, timeout)
 });
}
```

μ-Service D

```
const MAX_ATTEMPTS = 10;
let retryAttempts = 0;

function fetchData (url) {
 return request.get(url)
    .then(formatData)
    .catch(err => {
      if (retryAttempts > MAX_ATTEMPTS) return Promise.reject(err);
      retryAttempts++;
      await wait(500);
      return fetchData(url);
    });
}
```

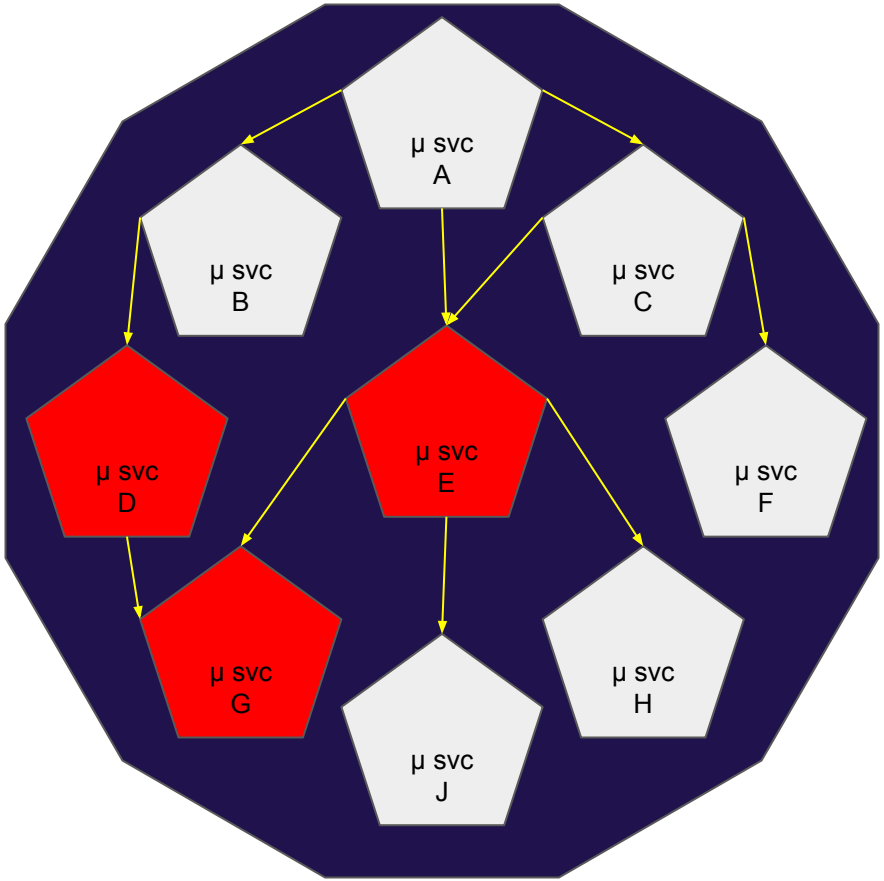
What Happens When We Keep On Trying?

(hint: things get worse)

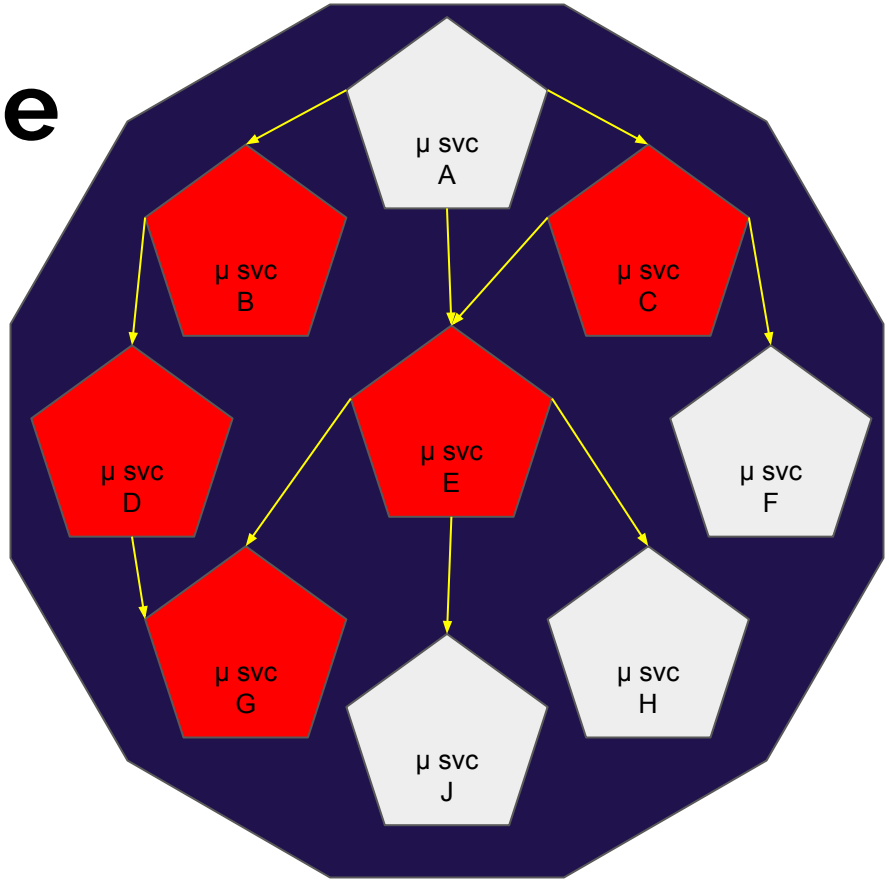
μ-Service G
Causes D and E to Block

So now what?

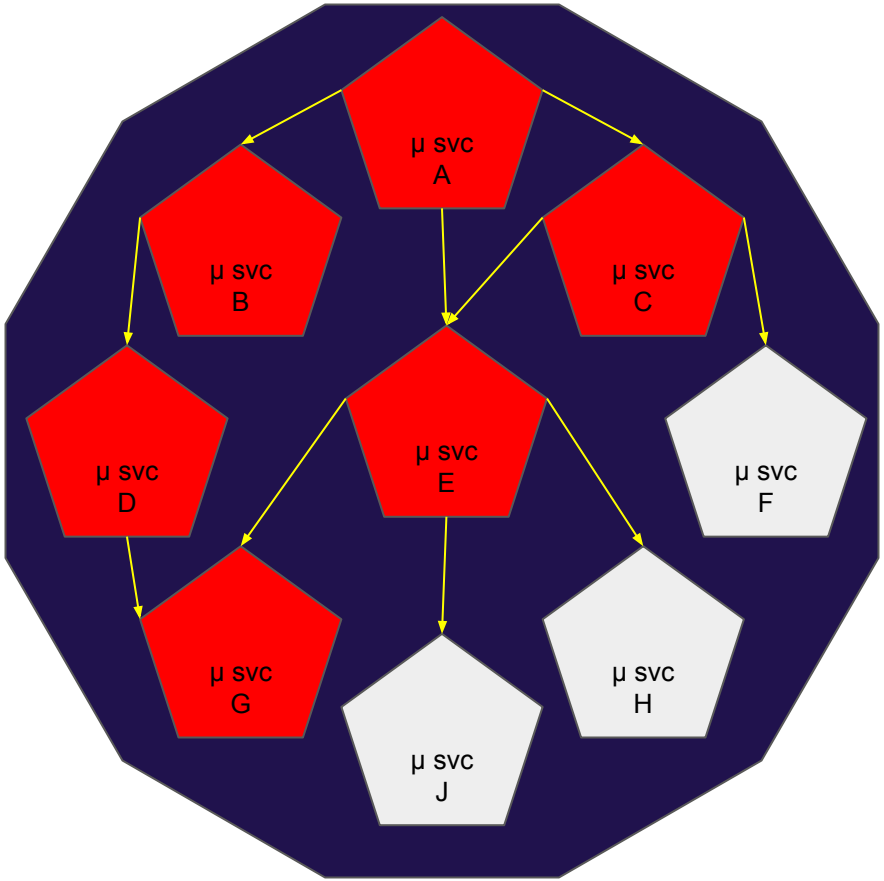
Causes More



Cascade



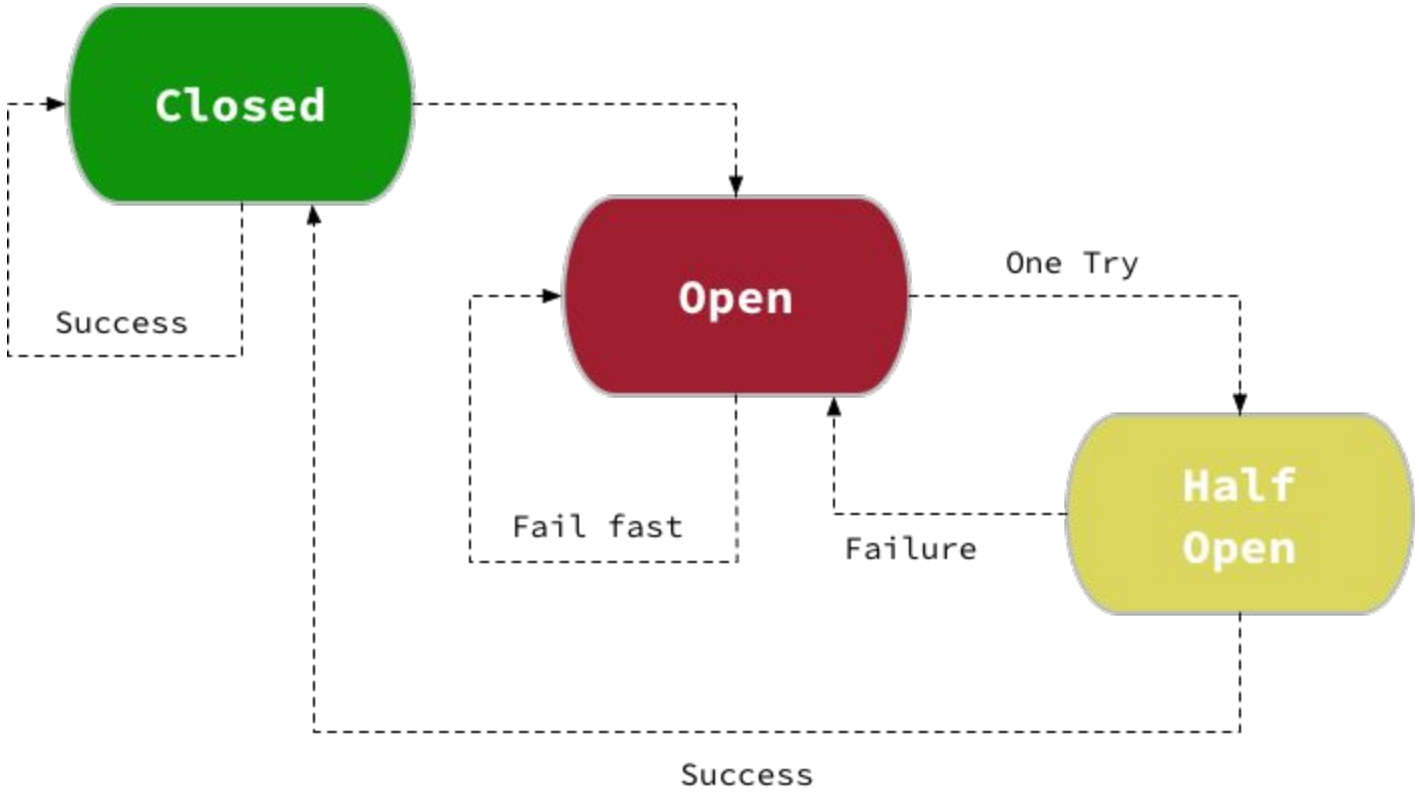
Dead App



Assume that an application connects to a remote service 100 times per second and the service fails. The application developer ***does not want to have the same error reoccur*** constantly. They also want to handle the error ***quickly and gracefully*** without waiting for TCP connection timeout.

Naive Implementations are a Band-Aid

Circuit Breakers

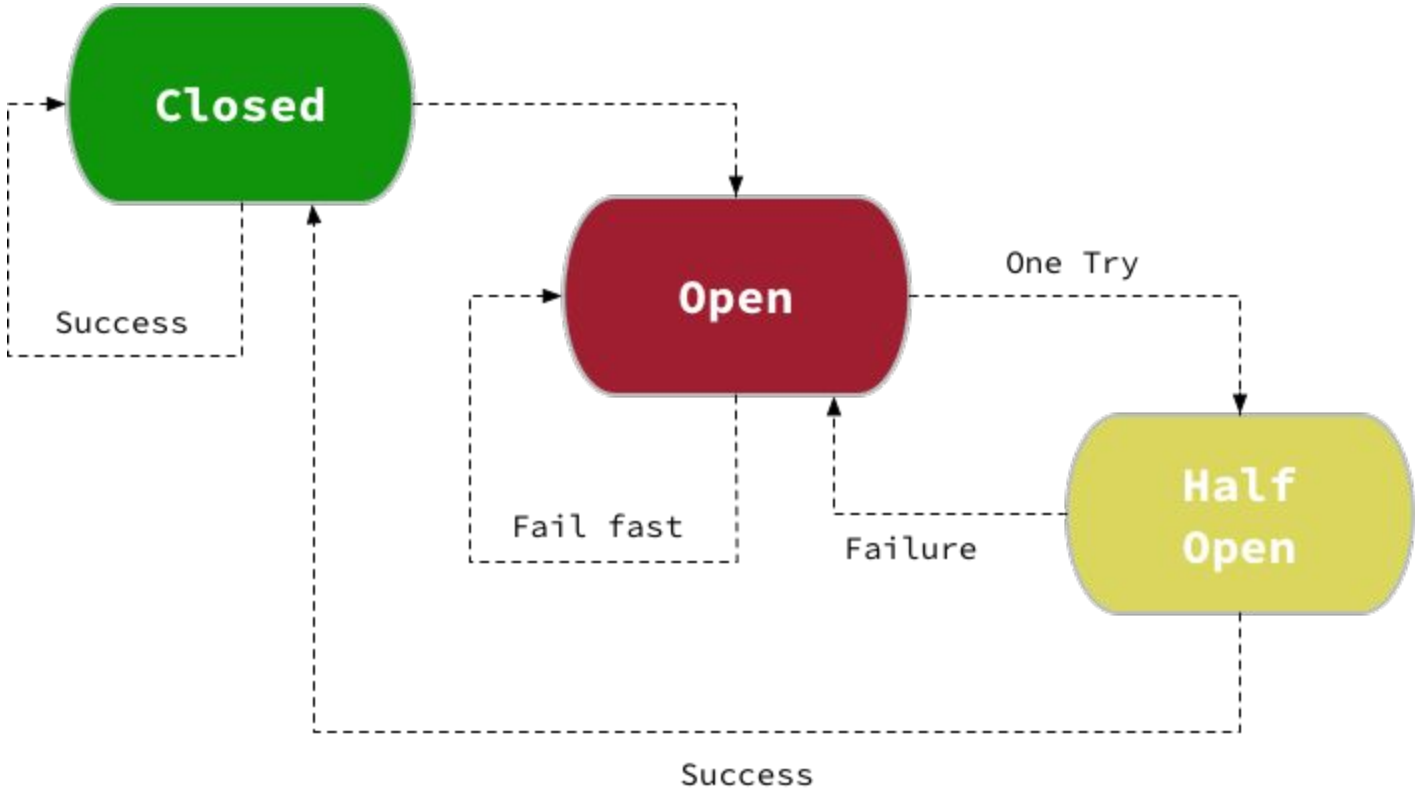

Closed
Success
Open
One Try
Fail fast
Failure
Half Open
Success

```
const CircuitBreaker = require(‘opossum’);

const options = {
 timeout: 1000,
 errorThresholdPercentage: 50,
 resetTimeout: 5000
}

const circuit =
  CircuitBreaker( fetchData('/some/url'), options );
```

```
function fetchData (url) {
 return _ => {
    return request.get(url)
      .then(formatData)
      .catch(err => {
        console.log(err)
      });
   }
}
```

```
circuit.fallback(
  _ => 'Sorry, out of service right now'
);

circuit.on('fallback',
  result => reportFallbackEvent(result));
```

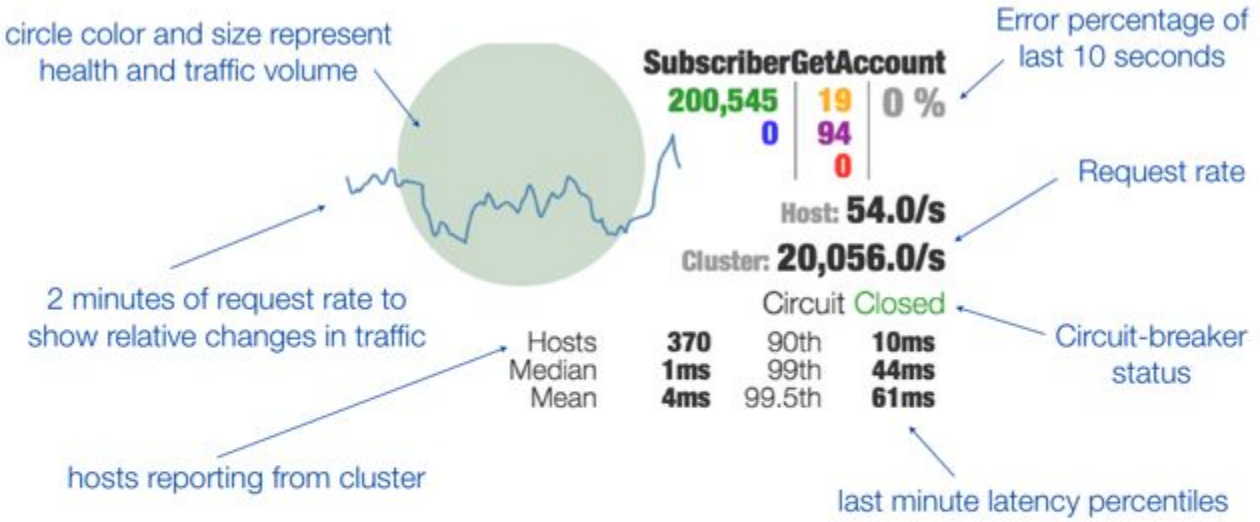
Events


Closed
Success
Open
One Try
Fail fast
Failure
Half Open
Success

Events

- fire
  - When the circuit is fired
- success
  - When the call is successful
- failure
  - When the call fails
- open
  - When the circuit opens
- close
  - When the circuit closes
- halfOpen
  - When the circuit enters half-open state
- fallback
  - When a fallback function is called
- cacheHit
  - A success value is in the cache
- cacheMiss
  - A value was not found in the cache
- timeout
  - When the call times out
- semaphore-locked
  - When resources are used up and no more calls can be made
- health-check-failed
  - When a user-supplied health check function fails
- snapshot
  - When a statistics snapshot is taken

Statistics Snapshots



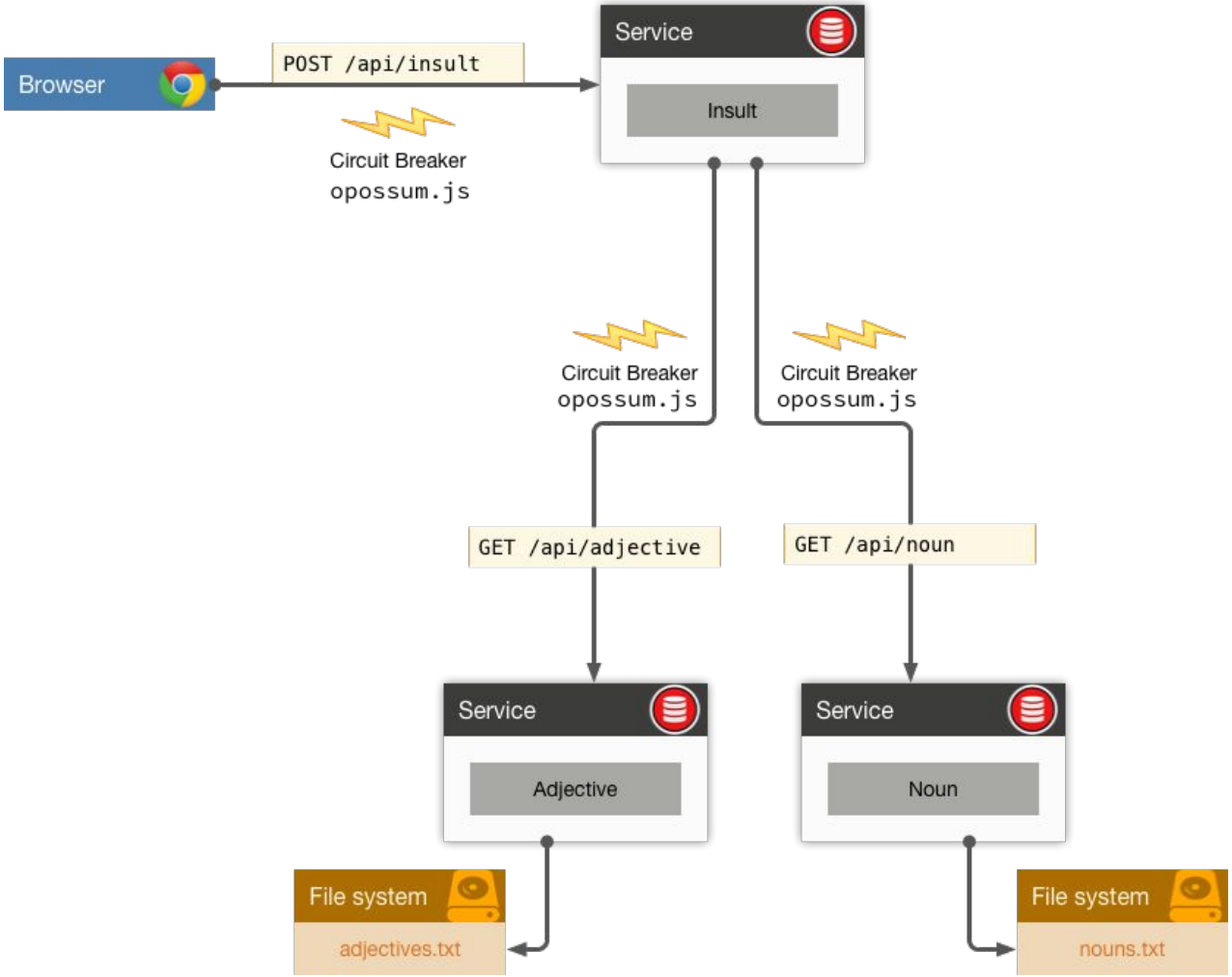
Rolling 10 second counters
with 1 second granularity

Successes	200,545	19	Thread timeouts
Short-circuited (rejected)	0	94	Thread-pool Rejections
		0	Failures/Exceptions

RHOAR Circuit Breaker

Demo Time!

But What About The Front End?


Browser
POST /api/insult
Service
Insult
Circuit Breaker
opossum.js
Circuit Breaker
opossum.js
Circuit Breaker
opossum.js
GET /api/adjective
GET /api/noun
Service
Adjective
Service
Noun
File system
adjectives.txt
File system
nouns.txt

Elizabethan Insults

Moar Demo Time!

```
const insult = circuitBreaker(getOrPostInsult, circuitBreakerOptions);
insult.fallback(_ => {
  return {
    name: 'Server Admin',
    adj1: 'sleep-addled',
    adj2: 'half witted',
    noun: 'bumbershoot'
  };
});

insult.on('failure', console.log);
insult.on('reject', console.log);
insult.on('open', console.log);

$('#invoke').click(e => insult.fire(e).then(updateInsultList));
$('#form-submit').submit(e => insult.fire(e).then(updateInsultList));
$('#clear').click(clearInsultList);
```

Merci Beaucoup!

https://github.com/bucharest-gold/nodejs-circuit-breaker
https://github.com/lance/elizabethan-insults
https://github.com/bucharest-gold/opossum
https://launch.openshift.io