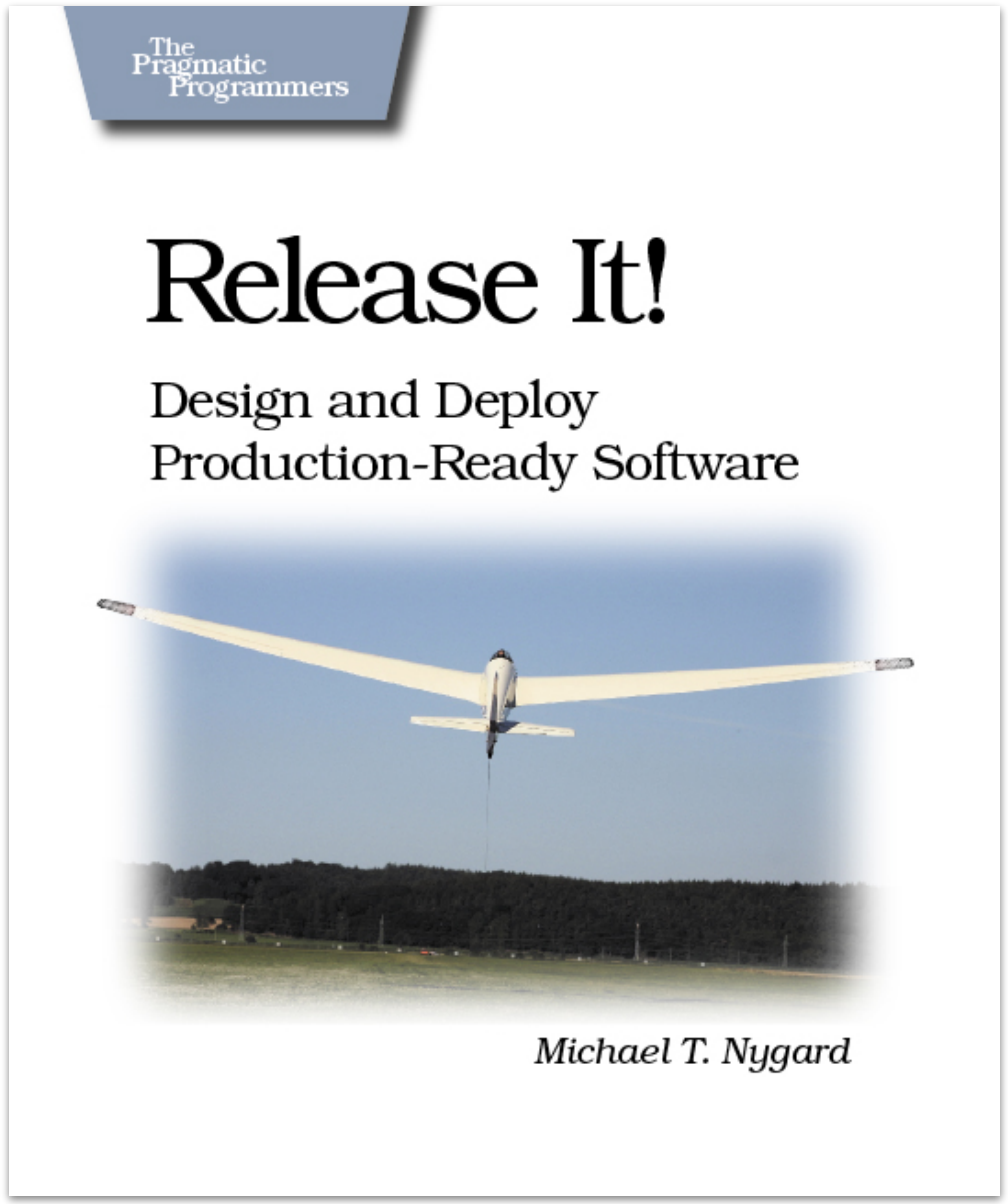
Circuit Breakers
with Netflix Hystrix

- Michael Nygard, 2007
- Describes fault tolerance patterns that became the inspiration for many of the implementations now in use with MicroServices


The Pragmatic Programmers
Release It!
Design and Deploy Production-Ready Software
Michael T. Nygard

Spring Cloud

- **Spring Cloud Contract**: supports and facilitates contract testing
- **Spring Cloud Netflix**: umbrella project offering a number of Netflix services and libraries adapted for Spring:
  - **Hystrix**, Eureka, Ribbon, Feign, Zuul
- **Spring Cloud Config Server**: configuration as a service
- **Spring Cloud Sleuth**: distributed tracing

Motivation

“Applications in complex distributed architectures have dozens of dependencies, each of which will inevitably fail at some point.

If the host application is not isolated from these external failures, it risks being taken down with them.”

HYSTRIX
DEFEND YOUR APP

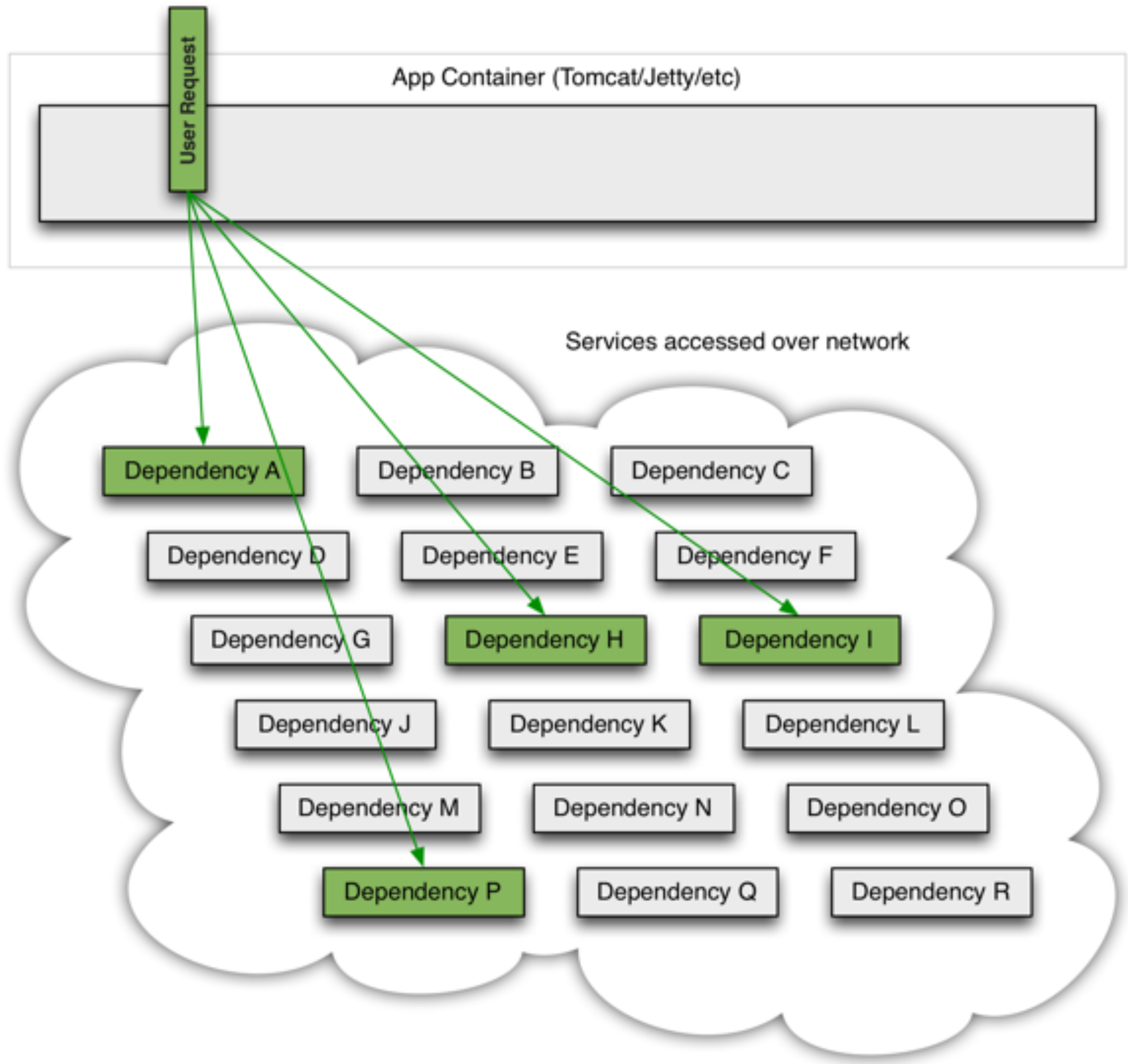
"In a distributed environment, inevitably some of the many service dependencies will fail.

Hystrix is a library that helps you control the interactions between these distributed services by adding latency tolerance and fault tolerance logic.

Hystrix does this by isolating points of access between the services, stopping cascading failures across them, and providing fallback options, all of which improve your system's overall resiliency."

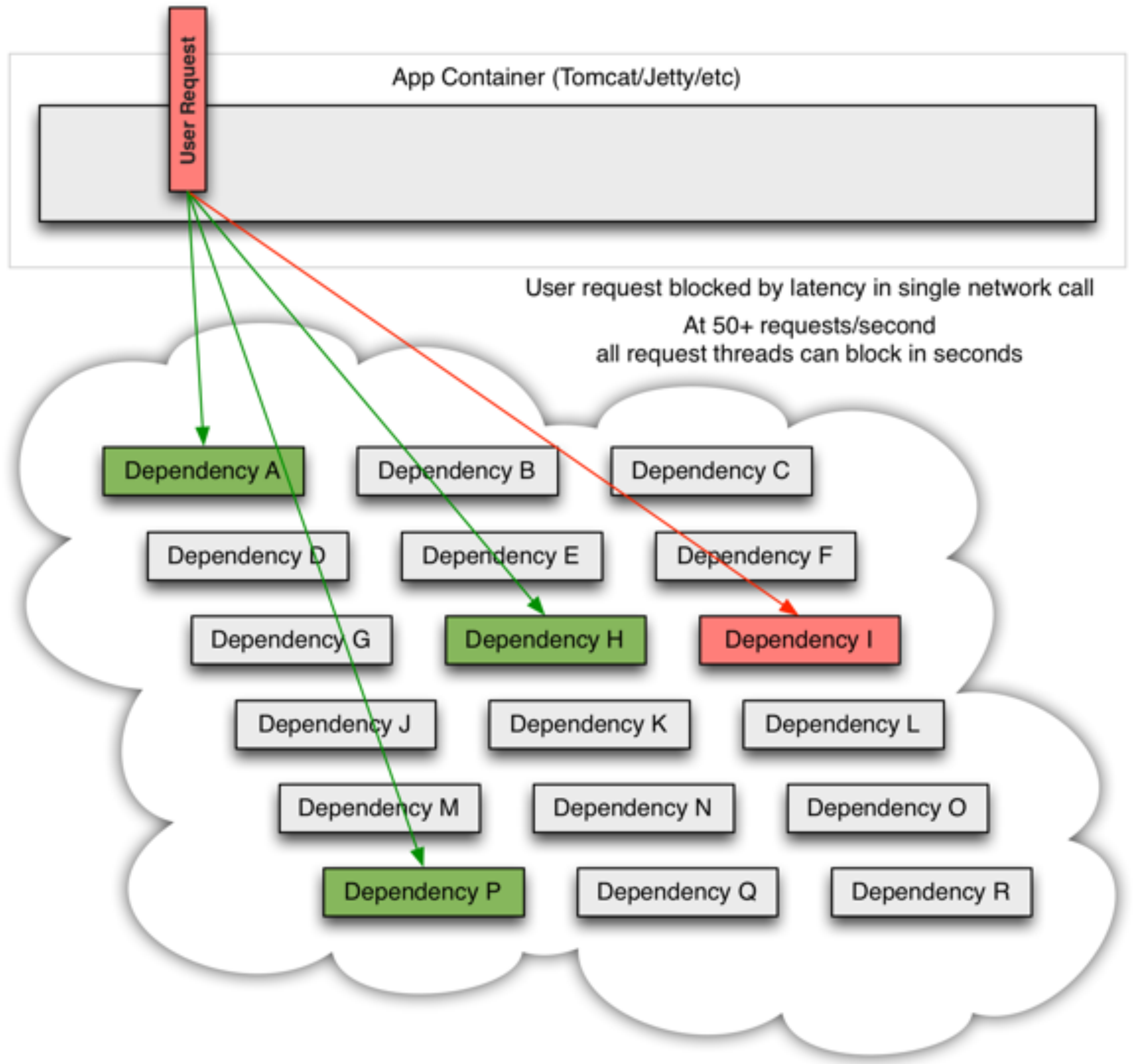
Services Dependency Scenario

- A typical application depending on a number of backing services
- All services are up and behaving normally
- Circuit is *Closed*



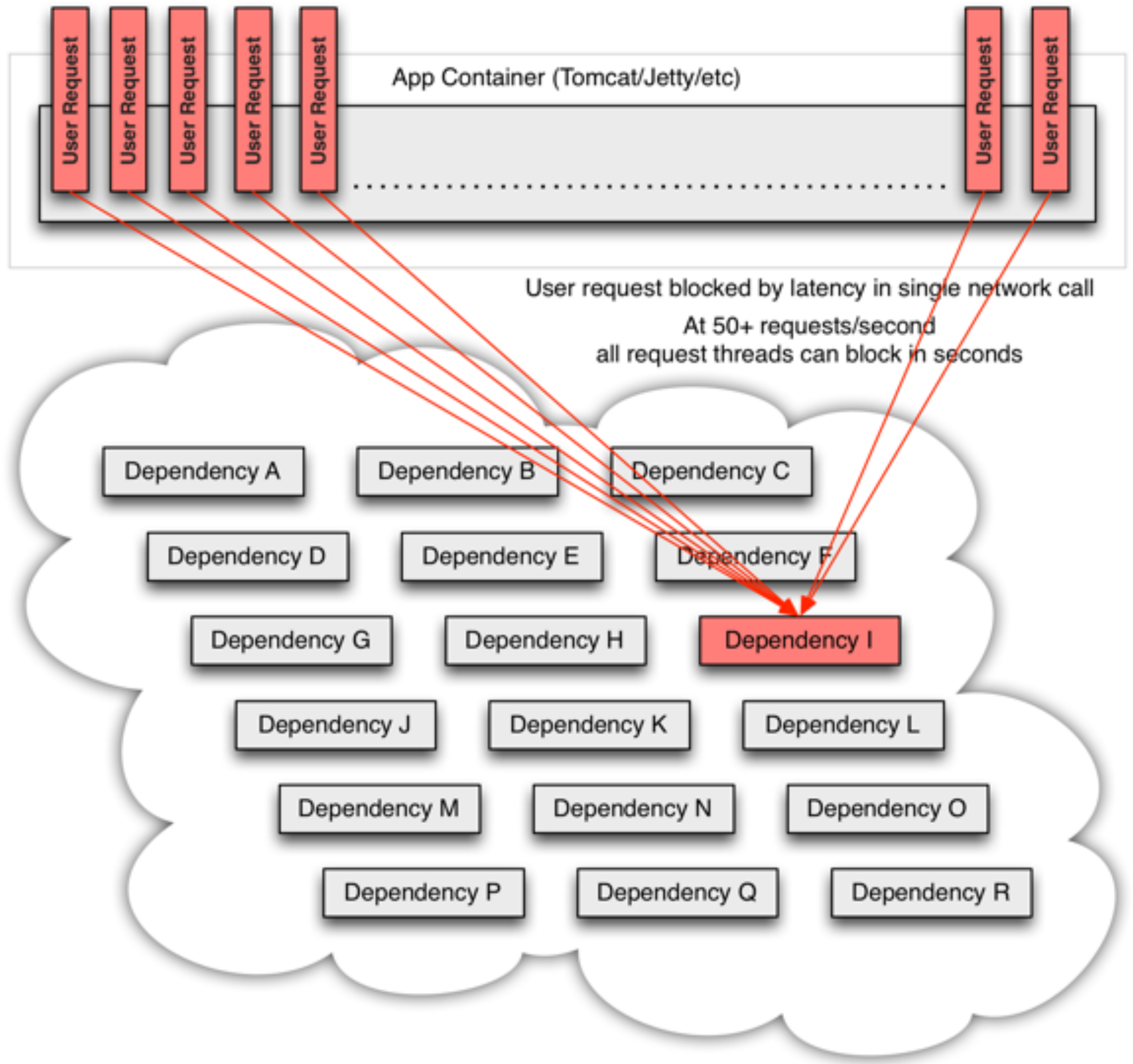
Failing Dependency

- A dependency begins misbehaves
- Response latency increases, tying up thread in calling application



Failure Cascades to Caller

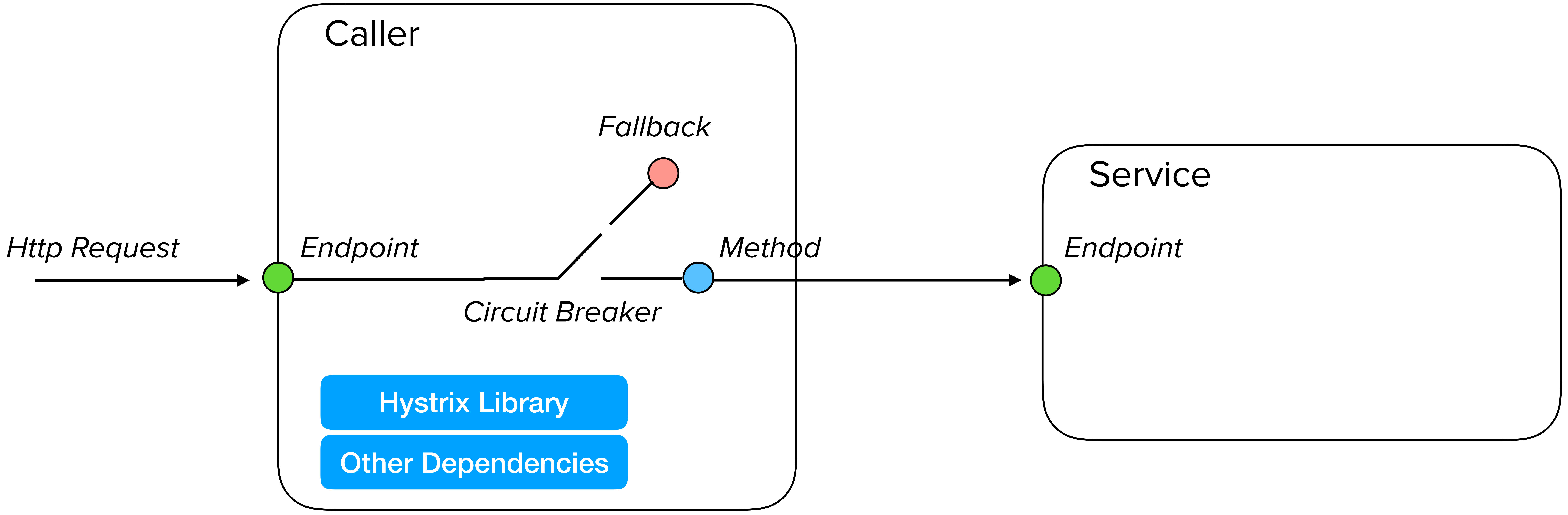
- Calling application’s thread pool is exhausted waiting on misbehaving dependency
- Failure cascades to caller

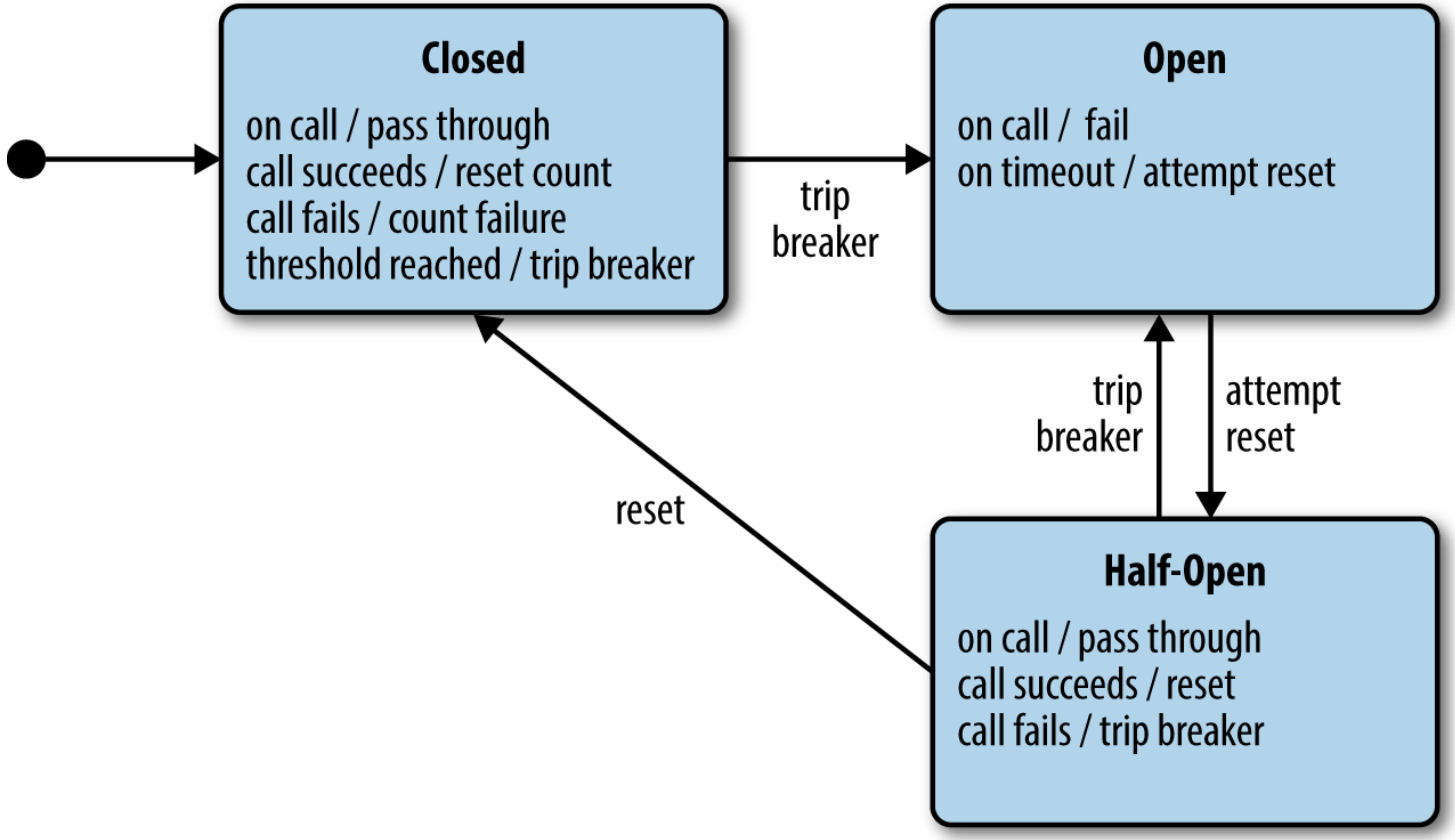


What is it for?

- Give protection from and control over latency and failure from dependencies accessed (typically over the network) via third-party client libraries.
- Stop cascading failures in a complex distributed system.
- Fail fast and rapidly recover.
- Fallback and gracefully degrade when possible.
- Enable near real-time monitoring, alerting, and operational control.

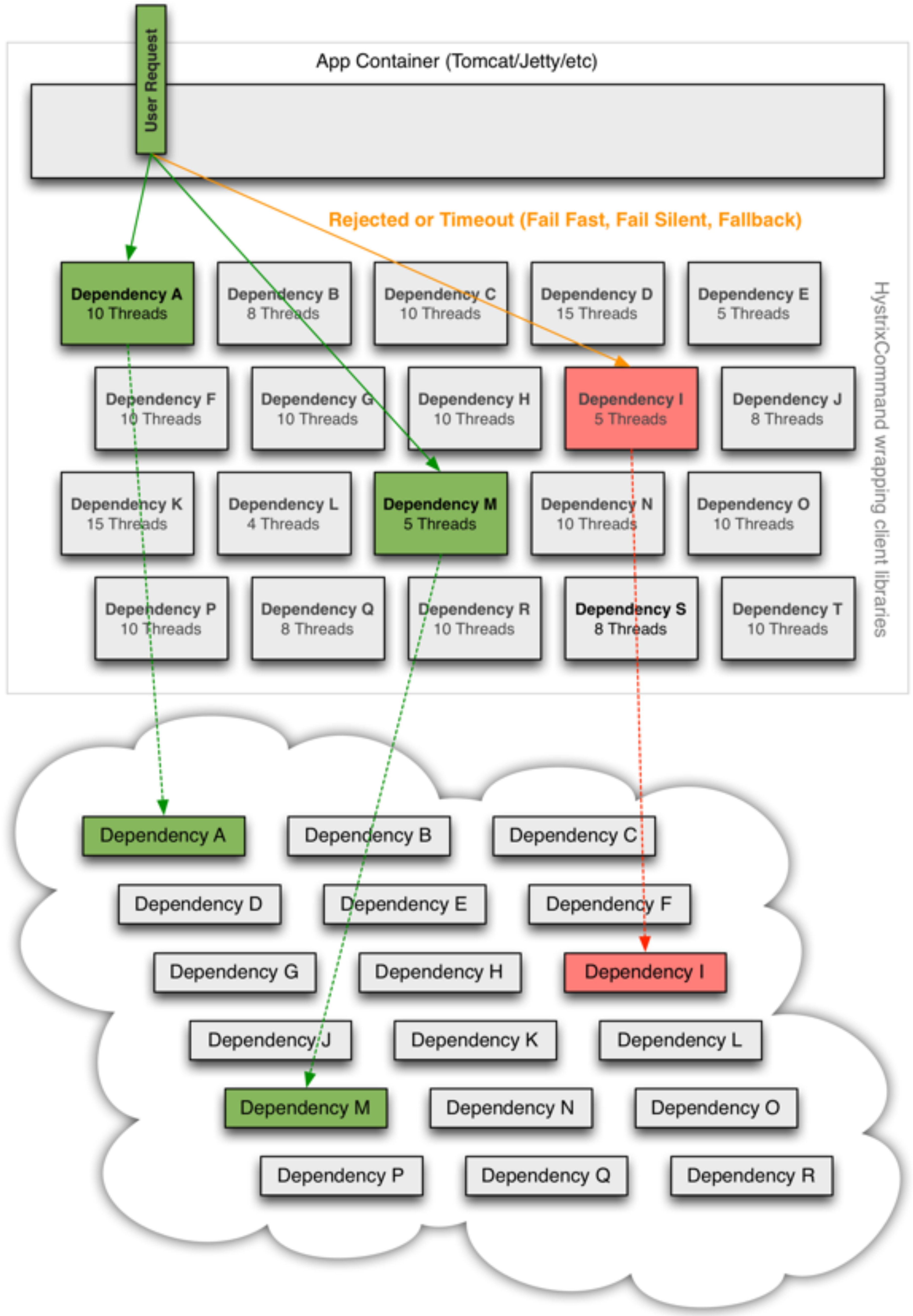
Circuit Breaker isolates calls to other services



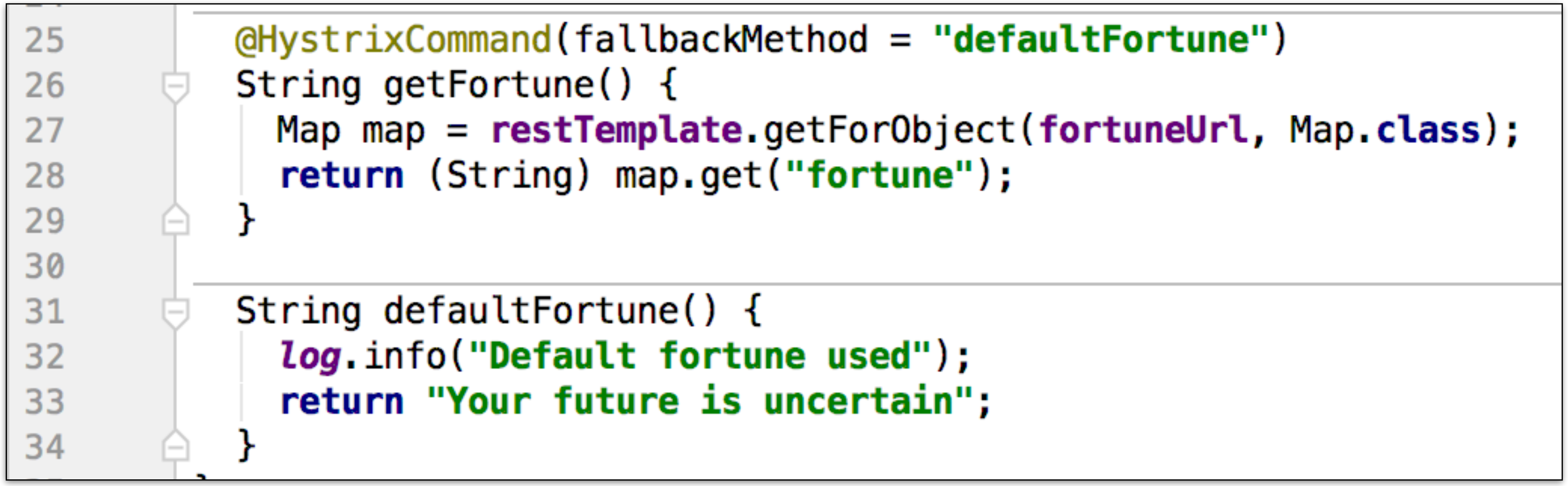

Closed
on call / pass through
call succeeds / reset count
call fails / count failure
threshold reached / trip breaker
trip breaker
Open
on call / fail
on timeout / attempt reset
trip breaker
attempt reset
reset
Half-Open
on call / pass through
call succeeds / reset
call fails / trip breaker

Application Protected with Hystrix

- Application is isolated from a misbehaving backing service
- When backing service health is restored, calling application will automatically reconfigure itself to call it once more



Annotating a service call



```java
@HystrixCommand(fallbackMethod = "defaultFortune")
String getFortune() {
    Map map = restTemplate.getForObject(fortuneUrl, Map.class);
    return (String) map.get("fortune");
}

String defaultFortune() {
    log.info("Default fortune used");
    return "Your future is uncertain";
}
```

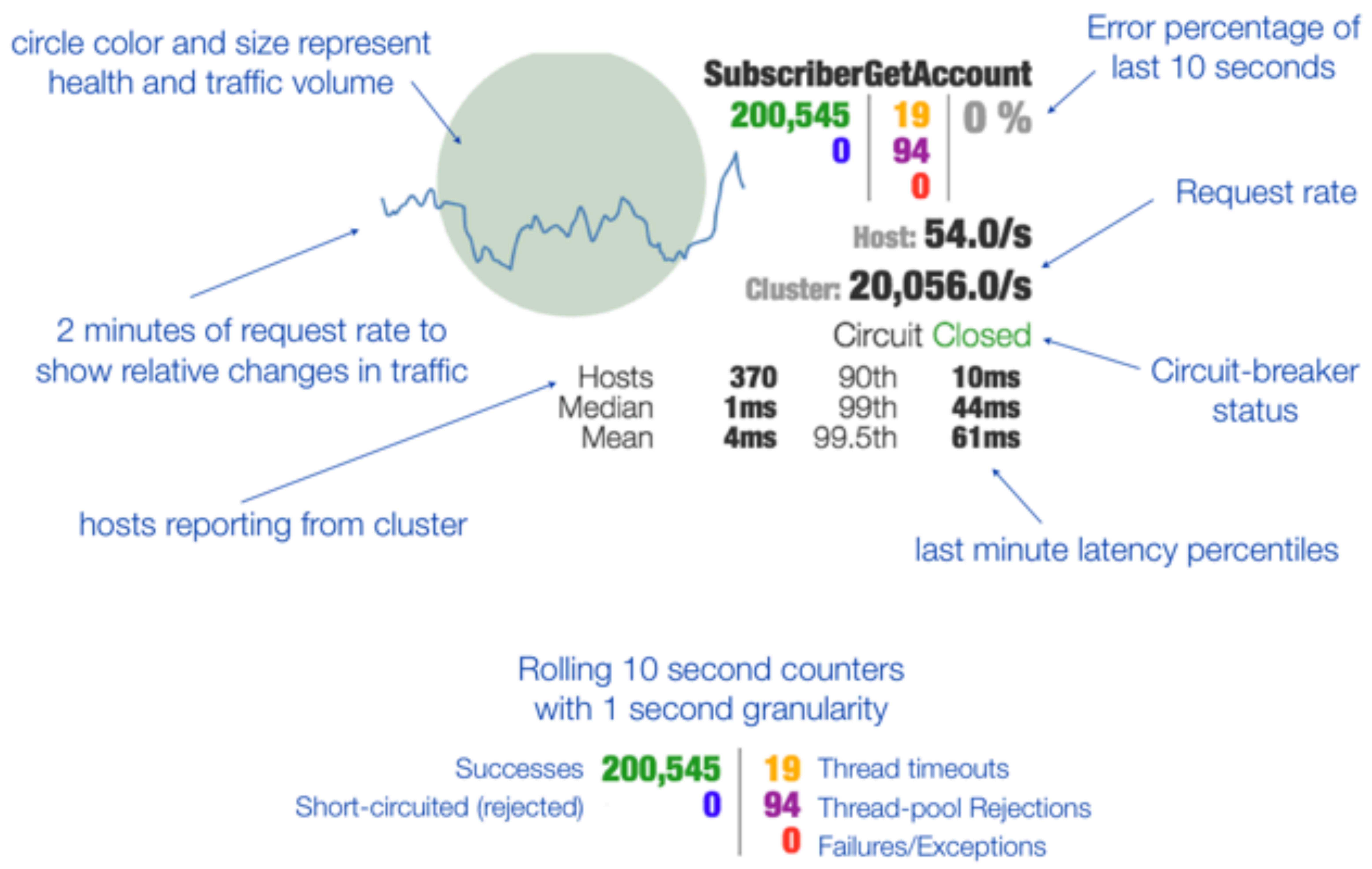
Configuration

	Default
execution.isolation.thread.timeoutInMilliseconds The time in milliseconds after which the caller will observe a timeout and walk away from the command execution	1000 ms
circuitBreaker.requestVolumeThreshold The minimum number of requests in a rolling window that will trip the circuit	20 requests min in a rolling window
circuitBreaker.sleepWindowInMilliseconds The amount of time, after tripping the circuit, to reject requests before allowing attempts again to determine if the circuit should again be closed	5000 ms
circuitBreaker.errorThresholdPercentage The error percentage at or above which the circuit should trip open and start short-circuiting requests to fallback logic	50%
hystrix.threadpool.*HystrixThreadPoolKey*.maximumSize The maximum thread-pool size. This is the maximum amount of concurrency that can be supported without starting to reject HystrixCommands	10

Hystrix Dashboard

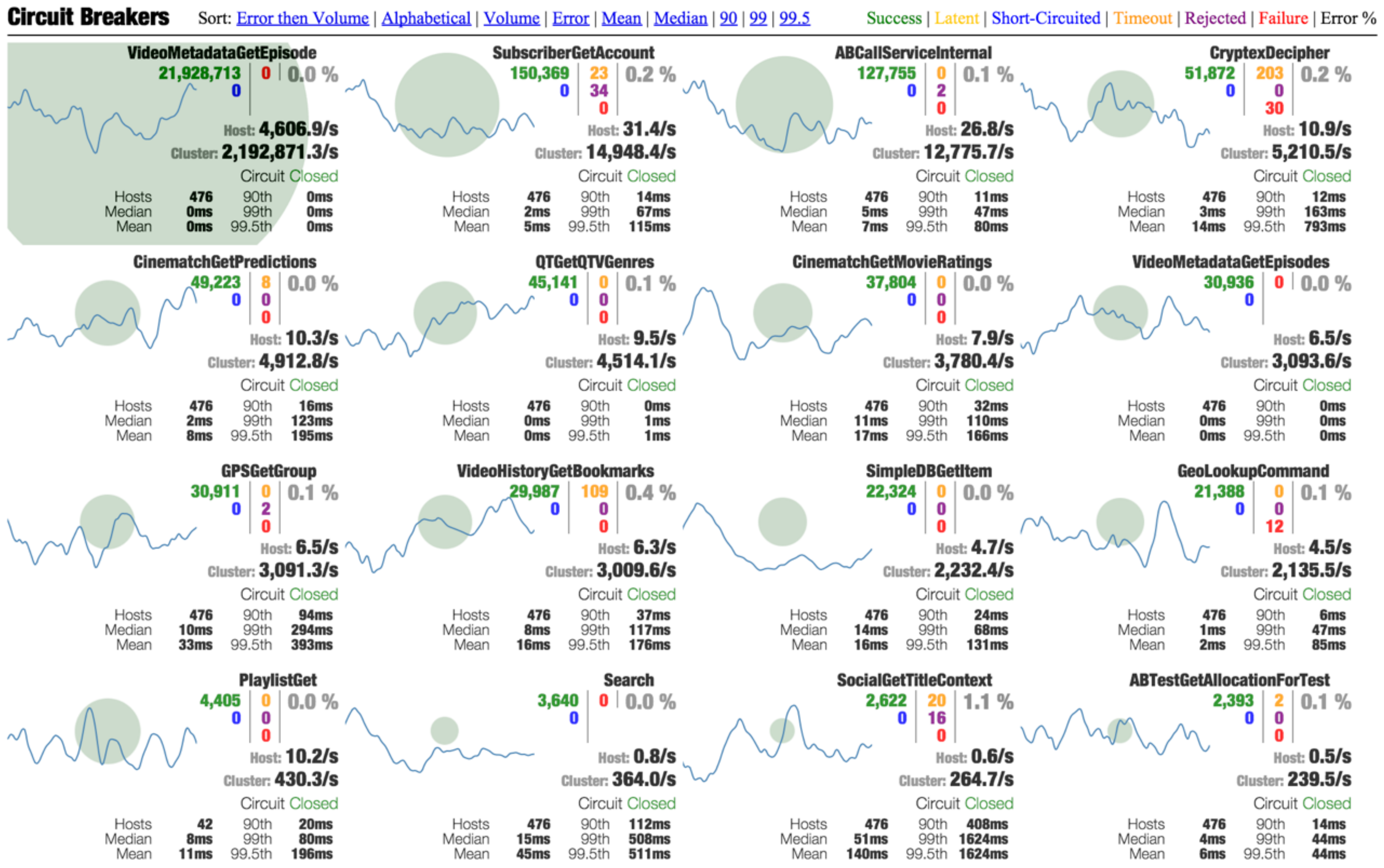
- A standalone application providing visualization of circuit state
- Calling application emits metrics via web socket endpoint /hystrix.stream
- Dashboard application consumes stream and renders metrics visualization for each circuit
- “Reduces time to discover and recover from operational events”

Circuit Breaker Monitoring



circle color and size represent health and traffic volume
SubscriberGetAccount
200,545
0
19
94
0
0 %
Error percentage of last 10 seconds
Request rate
Host: 54.0/s
Cluster: 20,056.0/s
2 minutes of request rate to show relative changes in traffic
Circuit Closed
Circuit-breaker status
Hosts 370
Median 1ms
Mean 4ms
90th 10ms
99th 44ms
99.5th 61ms
hosts reporting from cluster
last minute latency percentiles
Rolling 10 second counters with 1 second granularity
Successes 200,545
Short-circuited (rejected) 0
19 Thread timeouts
94 Thread-pool Rejections
0 Failures/Exceptions

Example Dashboard



Circuit Breakers Sort: Error then Volume | Alphabetical | Volume | Error | Mean | Median | 90 | 99 | 99.5

Success | Latent | Short-Circuited | Timeout | Rejected | Failure | Error %

VideoMetadataGetEpisode
21,928,713 | 0 | 0.0 %
0
Host: 4,606.9/s
Cluster: 2,192,871.3/s
Circuit Closed
Hosts 476 90th 0ms
Median 0ms 99th 0ms
Mean 0ms 99.5th 0ms

SubscriberGetAccount
150,369 | 23 | 0.2 %
0 | 34
0
Host: 31.4/s
Cluster: 14,948.4/s
Circuit Closed
Hosts 476 90th 14ms
Median 2ms 99th 67ms
Mean 5ms 99.5th 115ms

ABCallServiceInternal
127,755 | 0 | 0.1 %
0 | 2
0
Host: 26.8/s
Cluster: 12,775.7/s
Circuit Closed
Hosts 476 90th 11ms
Median 5ms 99th 47ms
Mean 7ms 99.5th 80ms

CryptexDecipher
51,872 | 203 | 0.2 %
0 | 0
30
Host: 10.9/s
Cluster: 5,210.5/s
Circuit Closed
Hosts 476 90th 12ms
Median 3ms 99th 163ms
Mean 14ms 99.5th 793ms

CinematchGetPredictions
49,223 | 8 | 0.0 %
0 | 0
0
Host: 10.3/s
Cluster: 4,912.8/s
Circuit Closed
Hosts 476 90th 16ms
Median 2ms 99th 123ms
Mean 8ms 99.5th 195ms

QTGetQTVGenres
45,141 | 0 | 0.1 %
0 | 0
0
Host: 9.5/s
Cluster: 4,514.1/s
Circuit Closed
Hosts 476 90th 0ms
Median 0ms 99th 1ms
Mean 0ms 99.5th 1ms

CinematchGetMovieRatings
37,804 | 0 | 0.0 %
0 | 0
0
Host: 7.9/s
Cluster: 3,780.4/s
Circuit Closed
Hosts 476 90th 32ms
Median 11ms 99th 110ms
Mean 17ms 99.5th 166ms

VideoMetadataGetEpisodes
30,936 | 0 | 0.0 %
0
Host: 6.5/s
Cluster: 3,093.6/s
Circuit Closed
Hosts 476 90th 0ms
Median 0ms 99th 0ms
Mean 0ms 99.5th 0ms

GPSGetGroup
30,911 | 0 | 0.1 %
0 | 2
0
Host: 6.5/s
Cluster: 3,091.3/s
Circuit Closed
Hosts 476 90th 94ms
Median 10ms 99th 294ms
Mean 33ms 99.5th 393ms

VideoHistoryGetBookmarks
29,987 | 109 | 0.4 %
0 | 0
0
Host: 6.3/s
Cluster: 3,009.6/s
Circuit Closed
Hosts 476 90th 37ms
Median 8ms 99th 117ms
Mean 16ms 99.5th 176ms

SimpleDBGetItem
22,324 | 0 | 0.0 %
0 | 0
0
Host: 4.7/s
Cluster: 2,232.4/s
Circuit Closed
Hosts 476 90th 24ms
Median 14ms 99th 68ms
Mean 16ms 99.5th 131ms

GeoLookupCommand
21,388 | 0 | 0.1 %
0 | 0
12
Host: 4.5/s
Cluster: 2,135.5/s
Circuit Closed
Hosts 476 90th 6ms
Median 1ms 99th 47ms
Mean 2ms 99.5th 85ms

PlaylistGet
4,405 | 0 | 0.0 %
0 | 0
0
Host: 10.2/s
Cluster: 430.3/s
Circuit Closed
Hosts 42 90th 20ms
Median 8ms 99th 80ms
Mean 11ms 99.5th 196ms

Search
3,640 | 0 | 0.0 %
0
Host: 0.8/s
Cluster: 364.0/s
Circuit Closed
Hosts 476 90th 112ms
Median 15ms 99th 508ms
Mean 45ms 99.5th 511ms

SocialGetTitleContext
2,622 | 20 | 1.1 %
0 | 16
0
Host: 0.6/s
Cluster: 264.7/s
Circuit Closed
Hosts 476 90th 408ms
Median 51ms 99th 1624ms
Mean 140ms 99.5th 1624ms

ABTestGetAllocationForTest
2,393 | 2 | 0.1 %
0 | 0
0
Host: 0.5/s
Cluster: 239.5/s
Circuit Closed
Hosts 476 90th 14ms
Median 4ms 99th 44ms
Mean 6ms 99.5th 44ms