Microsoft

Writing resilient microservices with Dapr

Shubham Sharma, Microsoft
@shubham1172

DaprCon Community Day

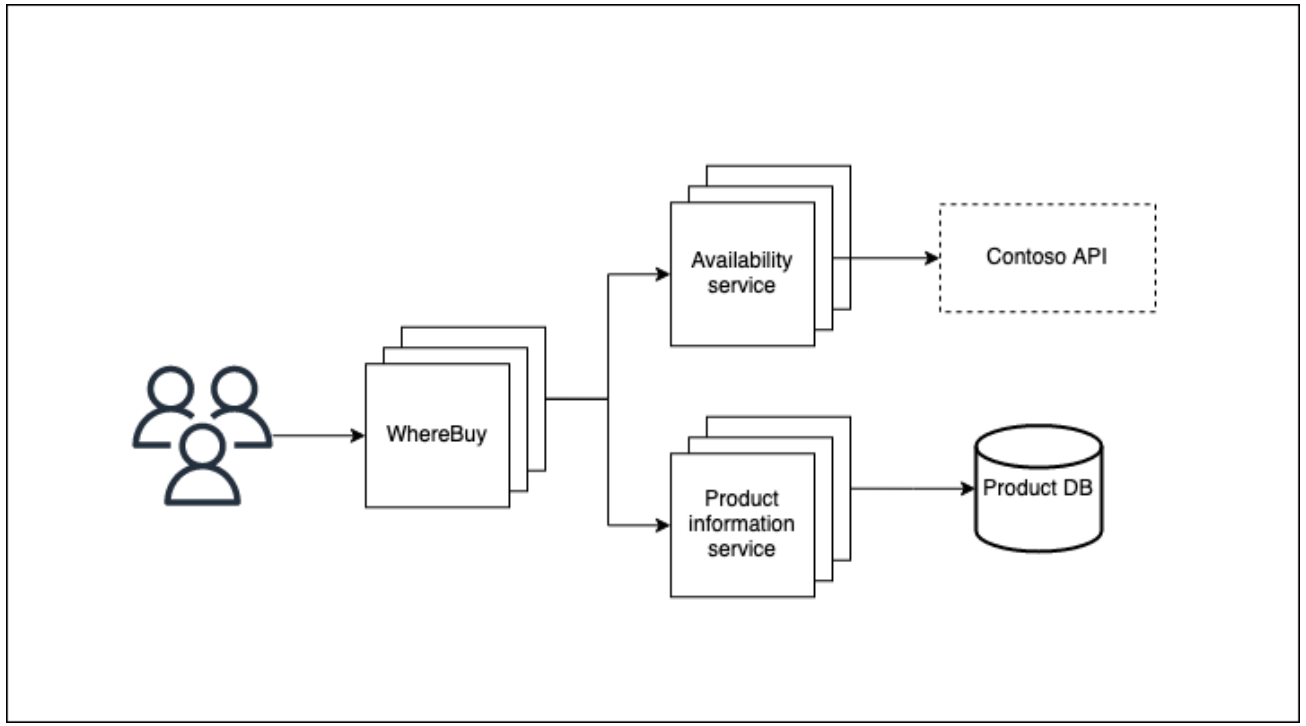
What’s inside

- Chaos in the real world
- Resiliency with Dapr
- Dapr-izing your applications

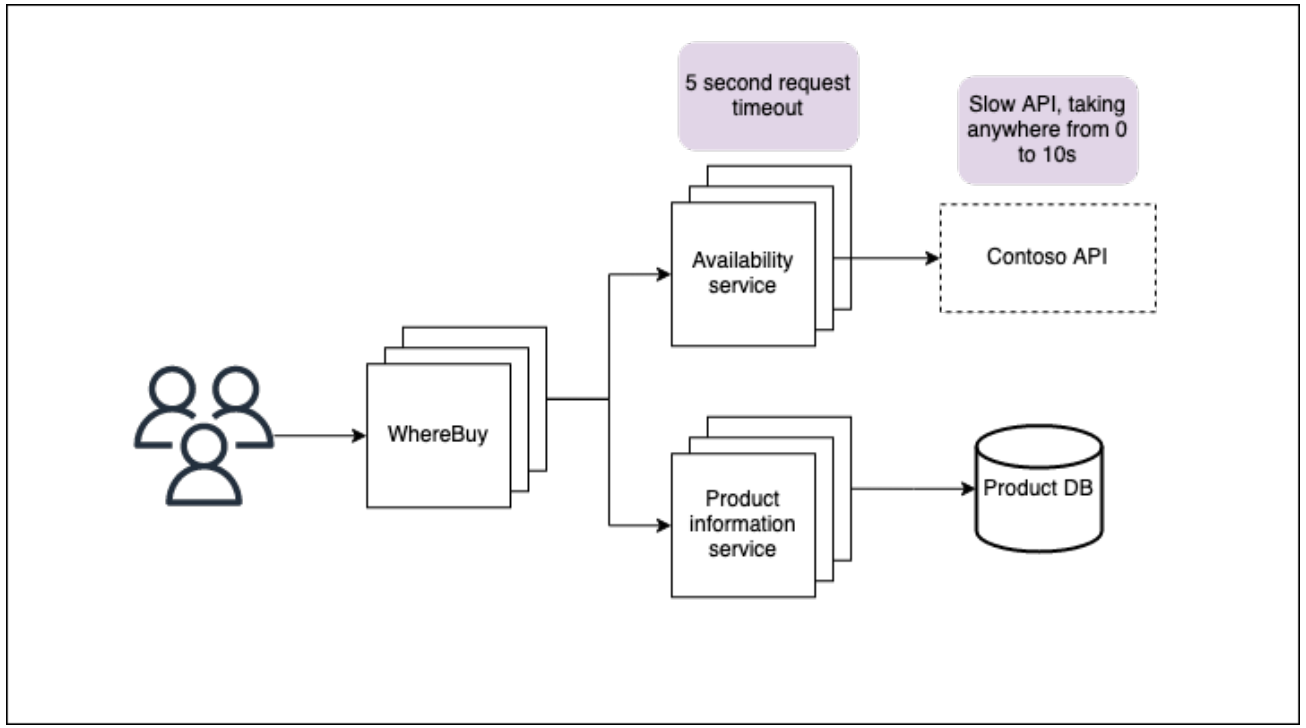
Chaos in the real world

Why microservices need to be resilient?

#wherebuy – low prices, more choices



Scenario A


5 second request timeout
Slow API, taking anywhere from 0 to 10s
Availability service
Contoso API
WhereBuy
Product information service
Product DB

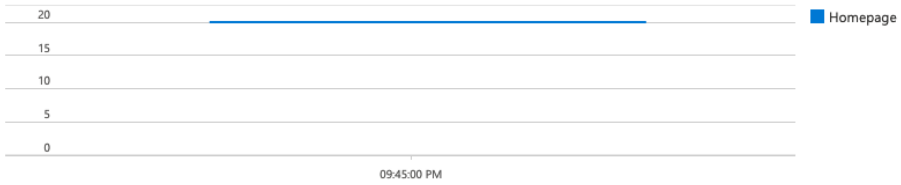
Scenario A

Load	Duration	Response time	Error percentage	Throughput
390	1 min, 7 secs	5.21 secs	40.00 %	5.91 /s
Total requests		90th percentile response time	Aggregate requests which failed	Request rate

Client-side metrics

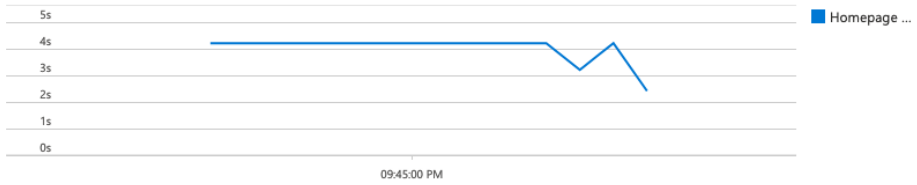
Requests : **Homepage** | Percentile : **90** | Error type : **2 selected** | Time range : **10/6/2022, 9:44:08 PM - 10/6/2022, 9:45:16 PM** | Group by : **5s**

Virtual Users (Max)



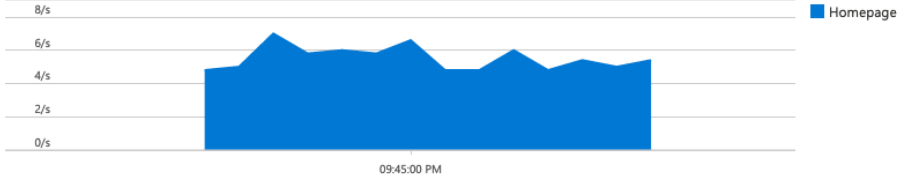
Homepage
20

Response time (successful responses)



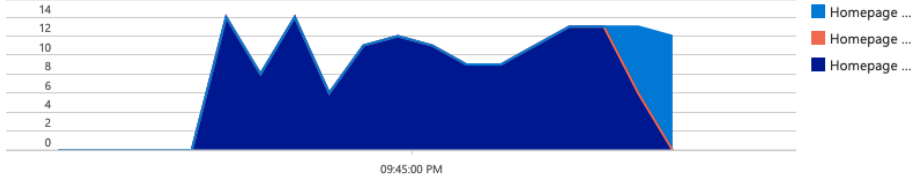
Homepage Pct 90
4.01 s

Requests/sec (Avg)



Homepage
5.51 /s

Errors (total)

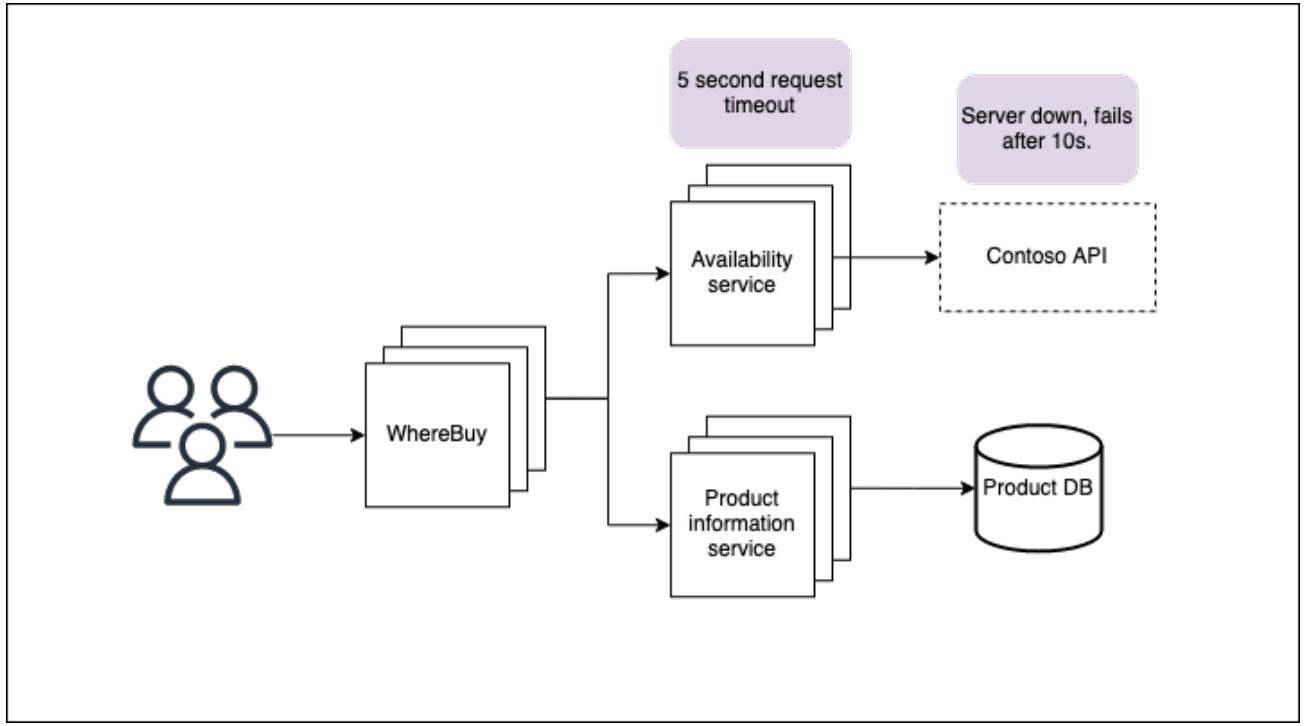


Homepage Error Non HT...
19

Homepage Error Aggregate
0

Homepage Error 500
137

Scenario B


5 second request timeout
Server down, fails after 10s.
Availability service
Contoso API
WhereBuy
Product information service
Product DB

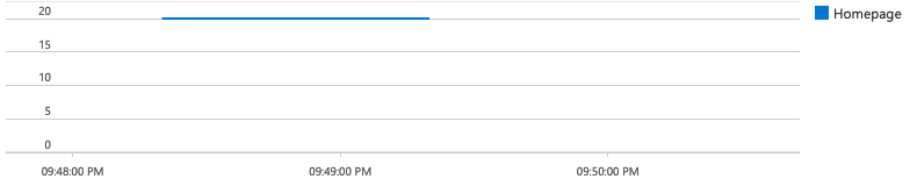
Scenario B

Load	Duration	Response time	Error percentage	Throughput
376	1 min, 1 sec	5.21 secs	100.00 %	6.16 /s
Total requests		90th percentile response time	Aggregate requests which failed	Request rate

Client-side metrics

Requests : **Homepage** | Percentile : **90** | Error type : **2 selected** | Time range : **10/6/2022, 9:47:58 PM - 10/6/2022, 9:49:11 PM** | Group by : **5s**

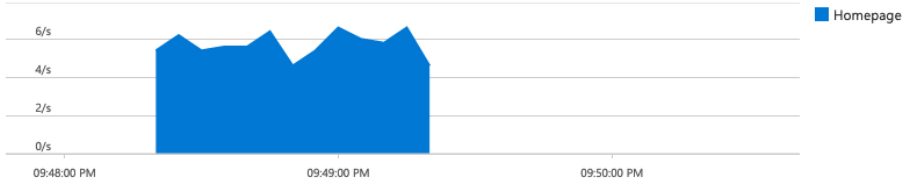
Virtual Users (Max)



Homepage
20

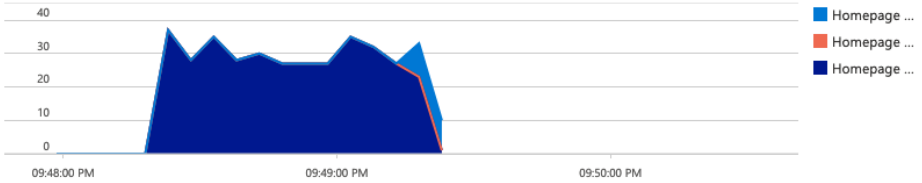
Response time (successful responses)

Requests/sec (Avg)



Homepage
5.71 /s

Errors (total)



Homepage Error Non HT...	Homepage Error Aggregate	Homepage Error 500
19	0	357

Resiliency with Dapr

Defining and applying fault tolerance resiliency policies

Overview

How does a failure impact the overall service health and end users?

Overview

How does a failure impact the overall service health and end users?

Reminder: Failure is inevitable.

Overview

How does a failure impact the overall service health and end users?

Reminder: Failure is inevitable.

Dapr allows defining popular resiliency **policies**, which can be applied to **targets**, and **scoped** to specific applications.

Overview

How does a failure impact the overall service health and end users?

Reminder: Failure is inevitable.

Dapr allows defining popular resiliency **policies**, which can be applied to **targets**, and **scoped** to specific applications.

Declarative and decoupled from application code.

Policies - Timeouts

- Failures can be slow
- Upper-bound for waiting

```yaml
spec:
  policies:
    timeouts:
      general: 5s
      largeResponse: 10s
      reallySlow: 60s
```

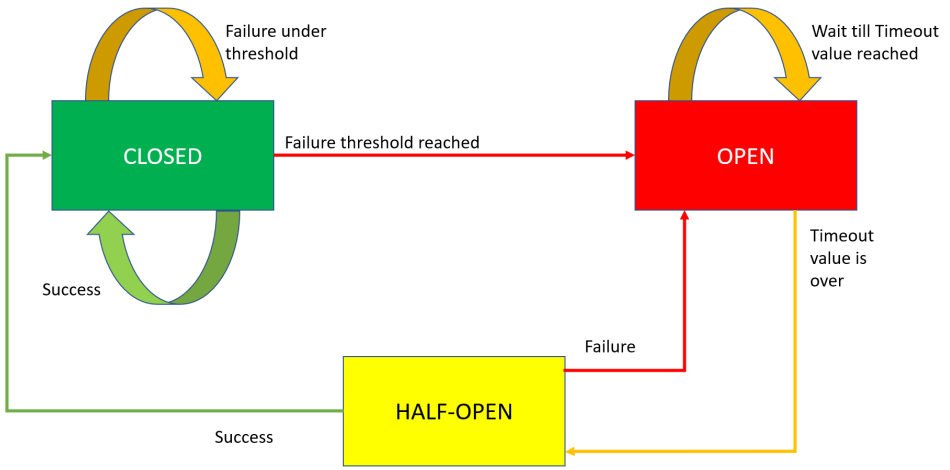
Policies - Retries

Choose between **constant** or **exponential** back-off retry policies

```yaml
spec:
  policies:
    retries:
      retryFinite:
        policy: constant # Default policy
        duration: 5s # Time interval between retries
        maxRetries: 10
      retryForever:
        policy: exponential
        maxInterval: 15s
        maxRetries: -1 # Retry indefinitely (default)
```

Policies – Circuit breakers

Handle elevated failure rates by **shutting off all traffic** to the impacted service when a certain criteria is met.



Source: [Proposal] Resiliency policies across all building blocks #3586 | dapr/dapr | GitHub

```
spec:
  policies:
    circuitBreakers:
      ExternalApiCallerCB:
        maxRequests: 1 # maximum requests when CB is half-open
        interval: 8s # period for clearing internal counts
        timeout: 45s # period of open state before switching to half-open
        trip: consecutiveFailures > 8 # CEL statement to trigger open state
```

Targets

(1) Apps

✅ Retries
✅ Timeouts
✅ Circuit breakers

Target service invocation calls between Dapr apps.

Targets

(1) Apps

✅ Retries
✅ Timeouts
✅ Circuit breakers

Target service invocation calls between Dapr apps.

(2) Components

✅ Retries
✅ Timeouts
✅ Circuit breakers

Target **inbound** (the sidecar calling your app) and **outbound** (calls to the sidecar) operations.

Targets

(1) Apps

✅ Retries
✅ Timeouts
✅ Circuit breakers

Target service invocation calls between Dapr apps.

(2) Components

✅ Retries
✅ Timeouts
✅ Circuit breakers

Target **inbound** (the sidecar calling your app) and **outbound** (calls to the sidecar) operations.

(3) Actors

✅ Retries
✅ Timeouts
✅ Circuit breakers

Circuit breaker can target an actor ID, all actors across the actor type, or both.

Dapr-izing your applications

Resilient microservices out-of-box.

Using Dapr with your application

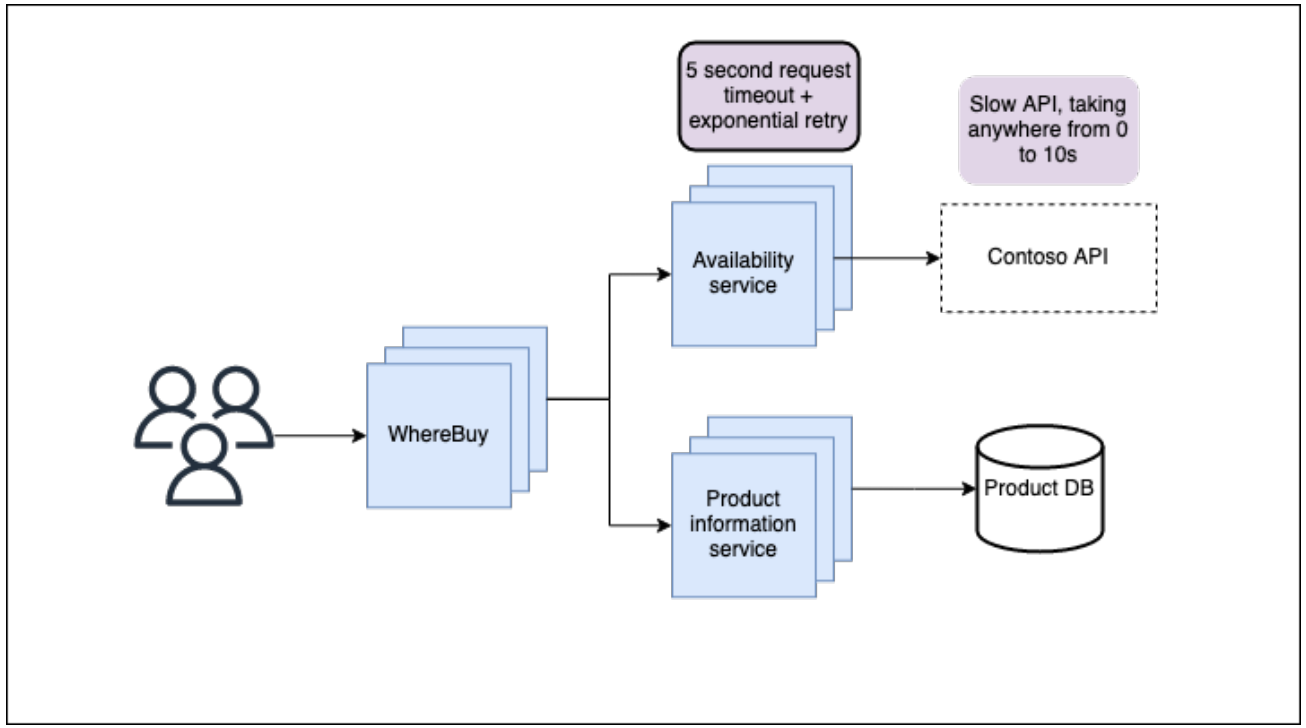
(1) Configure Dapr annotations

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: availabilityapp
  labels:
    app: availability
spec:
  template:
    metadata:
      # Just add these to enable Dapr!
      annotations:
        dapr.io/enabled: "true"
        dapr.io/app-id: "availabilityapp"
        dapr.io/enable-api-logging: "true"
        dapr.io/log-level: "debug"
...
```

(2) Invoke services using Dapr

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontendapp
  labels:
    app: frontend
...
    spec:
      containers:
        - name: frontend
          image: ghcr.io/shubham1172/daprcon/frontend:latest
          env:
            # Use these when Dapr is disabled
            # - name: WHEREBUY_AVAILABILITY_API_URL
            #   value: http://availability/check
            # - name: WHEREBUY_PRODUCT_API_URL
            #   value: http://product/get
            # Use these when Dapr is enabled
            - name: WHEREBUY_AVAILABILITY_API_URL
              value: http://localhost:3500/v1.0/invoke/availabilityapp/method/check
            - name: WHEREBUY_PRODUCT_API_URL
              value: http://localhost:3500/v1.0/invoke/productapp/method/get
...
```

Scenario A – with Dapr



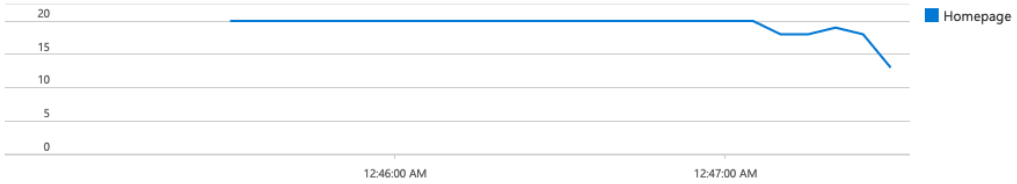
Scenario A – with Dapr

Load	Duration	Response time	Error percentage	Throughput
448	2 mins	11.40 secs	0.67 %	3.76 /s
Total requests		90th percentile response time	Aggregate requests which failed	Request rate

Client-side metrics

Requests : **Homepage** | Percentile : **90** | Error type : **500** | Time range : **10/7/2022, 12:45:00 AM - 10/7/2022, 12:47:14 AM** | Group by : **5s**

Virtual Users (Max)

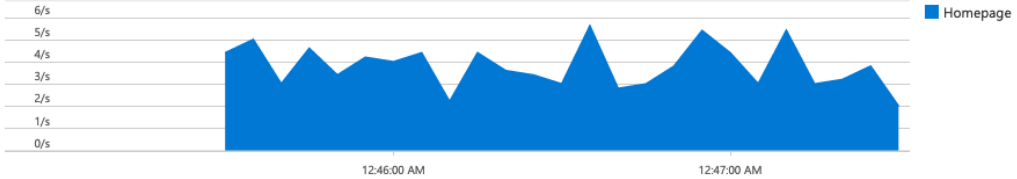


Homepage
20

Response time (successful responses)

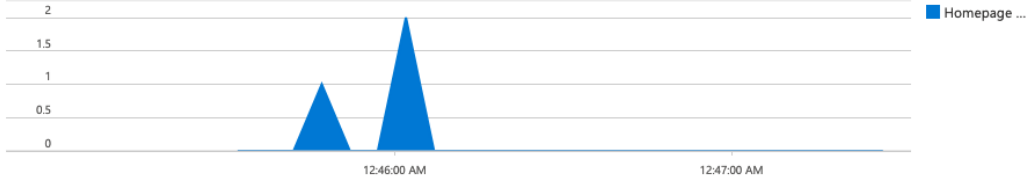
Homepage Pct 90
10.51 s

Requests/sec (Avg)



Homepage
3.8 /s

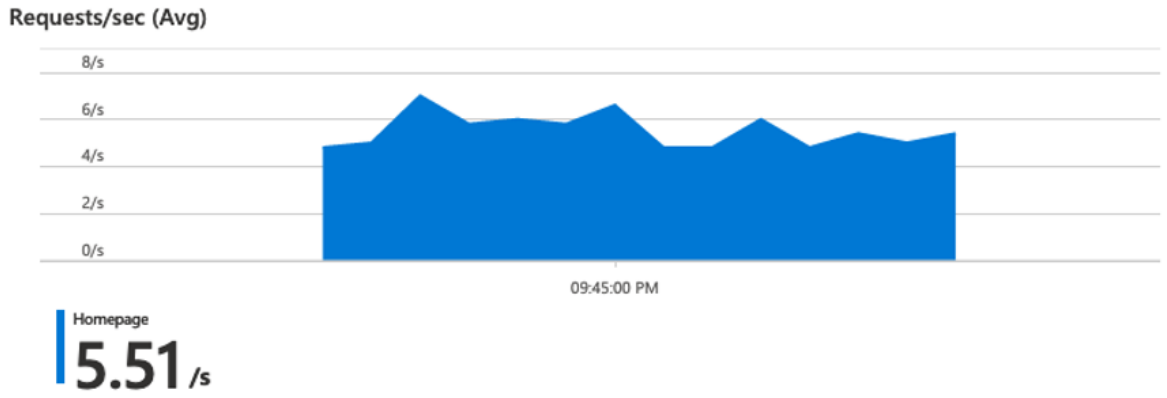
Errors (total)

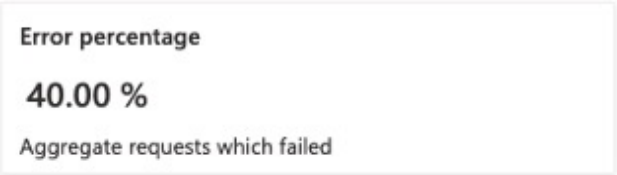


Homepage Error 500
3

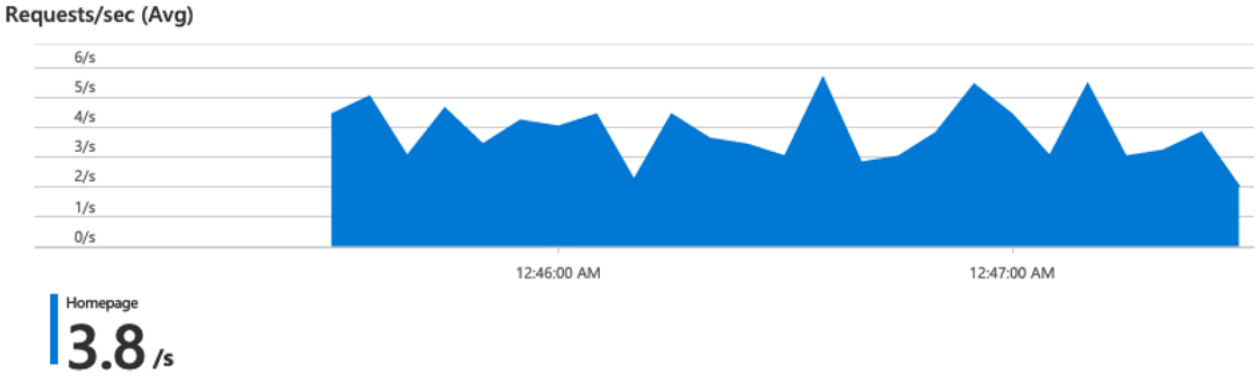
Scenario A – comparison

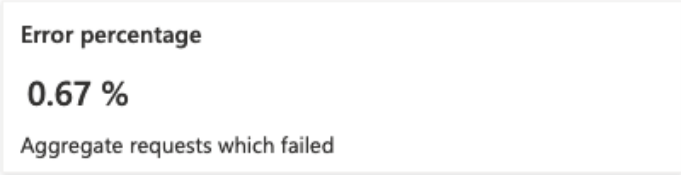
Without Dapr



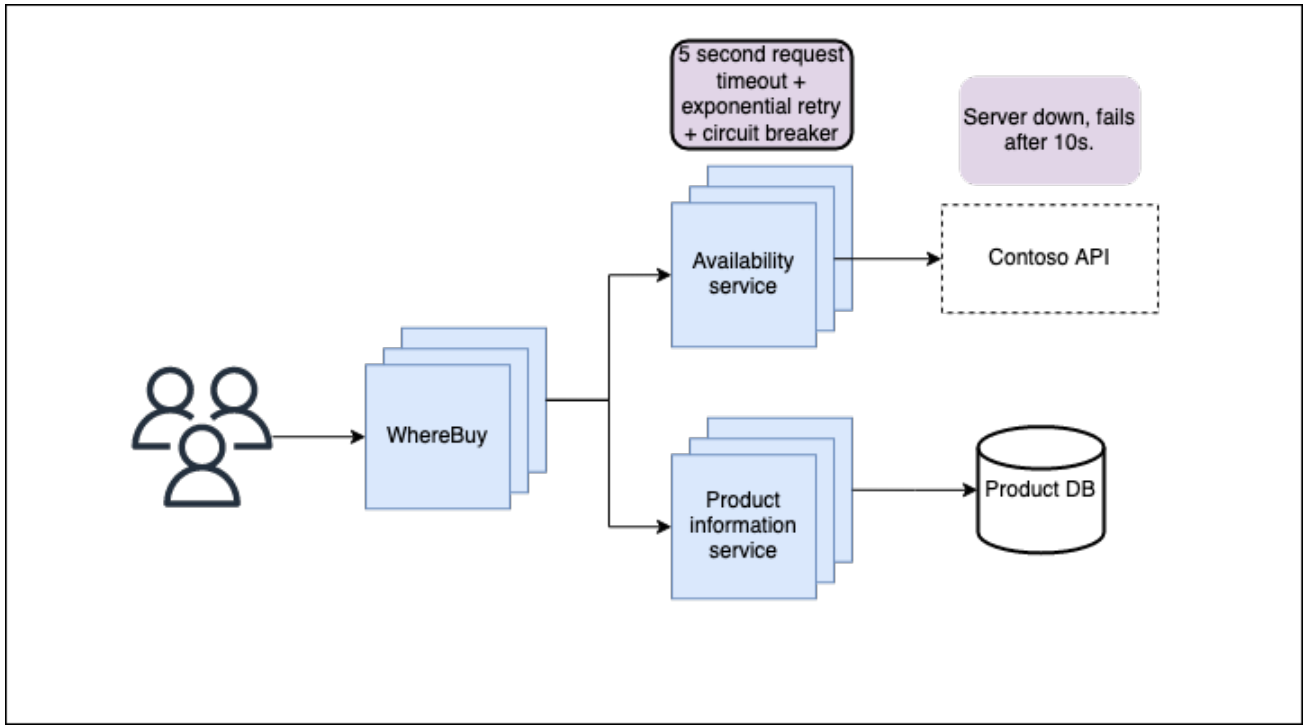


With Dapr





Scenario B – with Dapr



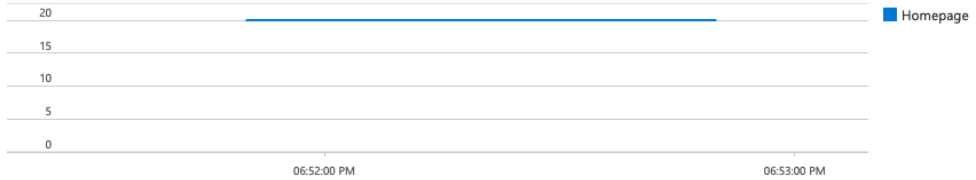
Scenario B – with Dapr

Load	Duration	Response time	Error percentage	Throughput
5677	1 min, 3 secs	205.00 ms	100.00 %	91.56 /s
Total requests		90th percentile response time	Aggregate requests which failed	Request rate

Client-side metrics

Requests : **Homepage** | Percentile : **90** | Error type : **2 selected** | Time range : **10/7/2022, 6:51:27 PM - 10/7/2022, 6:52:34 PM** | Group by : **5s**

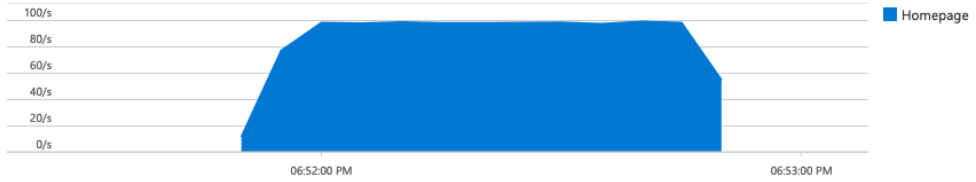
Virtual Users (Max)



Homepage
20

Response time (successful responses)

Requests/sec (Avg)



Homepage
86.17 /s

Errors (total)

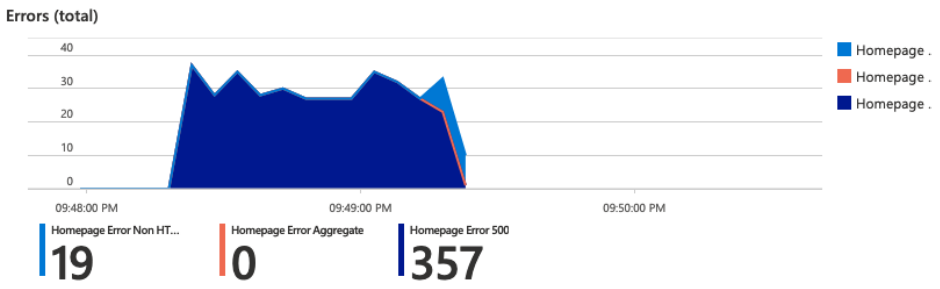
Homepage Error Non HT...
19

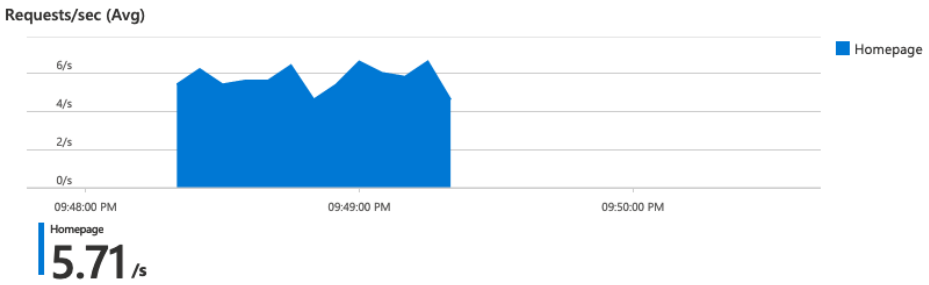
Homepage Error Aggregate
0

Homepage Error 500
5.66 K

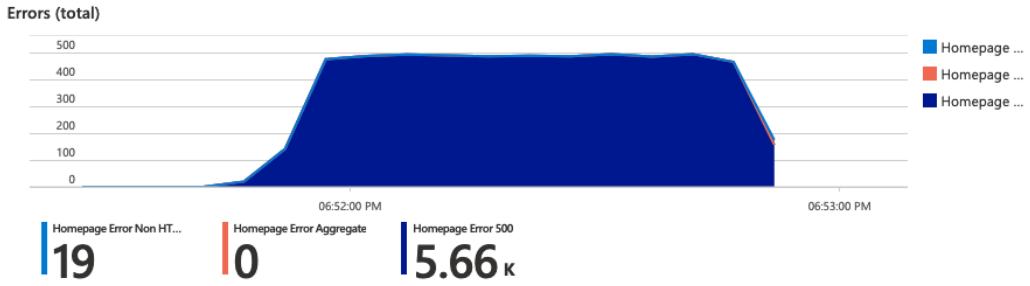
Scenario B – comparison

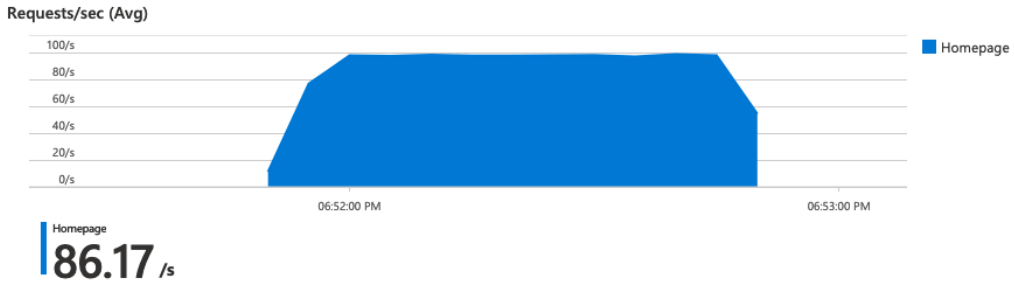
Without Dapr





With Dapr





Resources

Documentation

Overview | Resiliency | Dapr Docs

Dapr arguments and annotations for daprd, CLI, and Kubernetes | Dapr Docs

[Proposal] Resiliency policies across all building blocks #3586
[Original proposal on Dapr GitHub repository]

Code

shubham1172/daprcon-resiliency
[Code from this talk]

Microsoft

Thank you