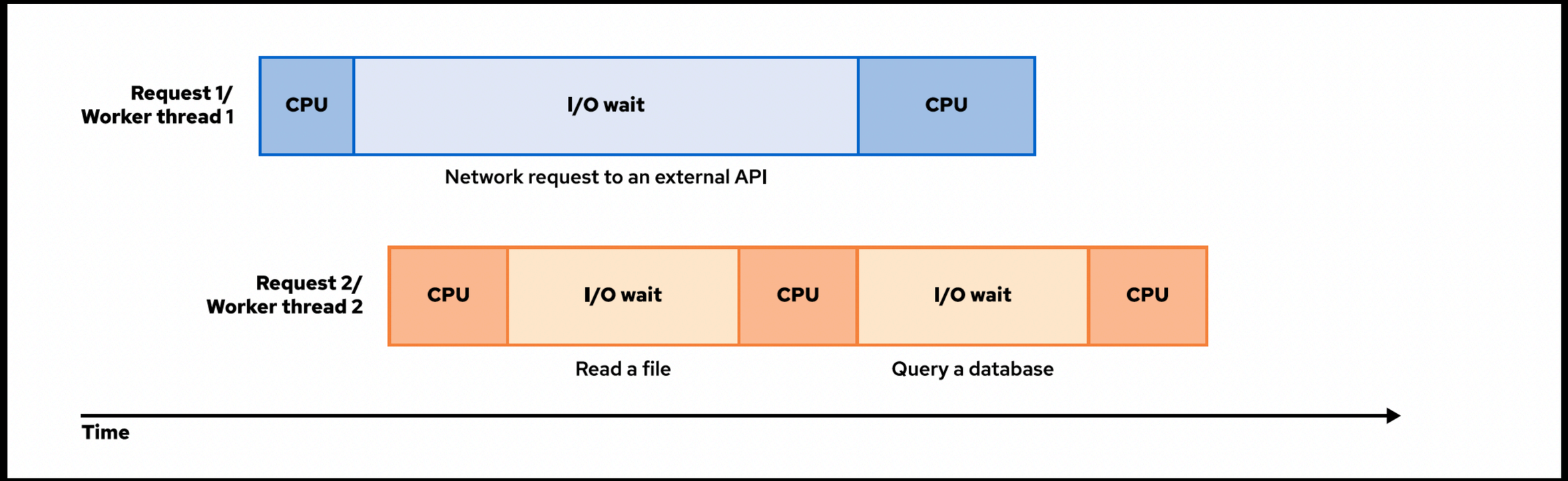
Desarrollando microservicios

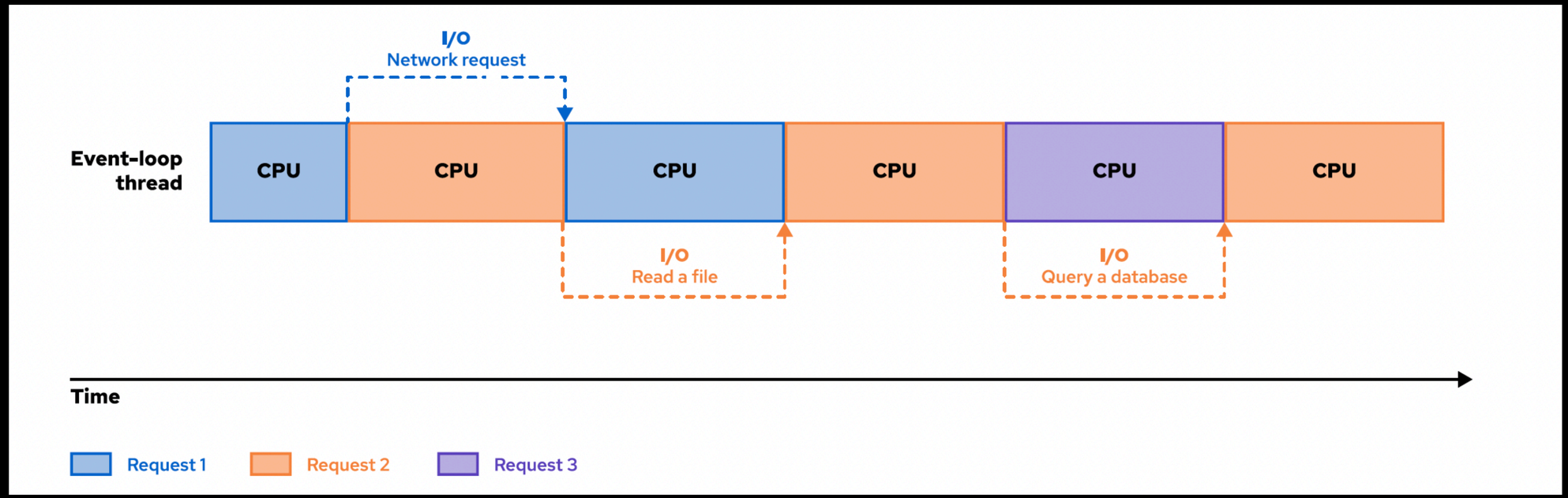
Reactivos y Asíncronos

JoeDayz | 2024

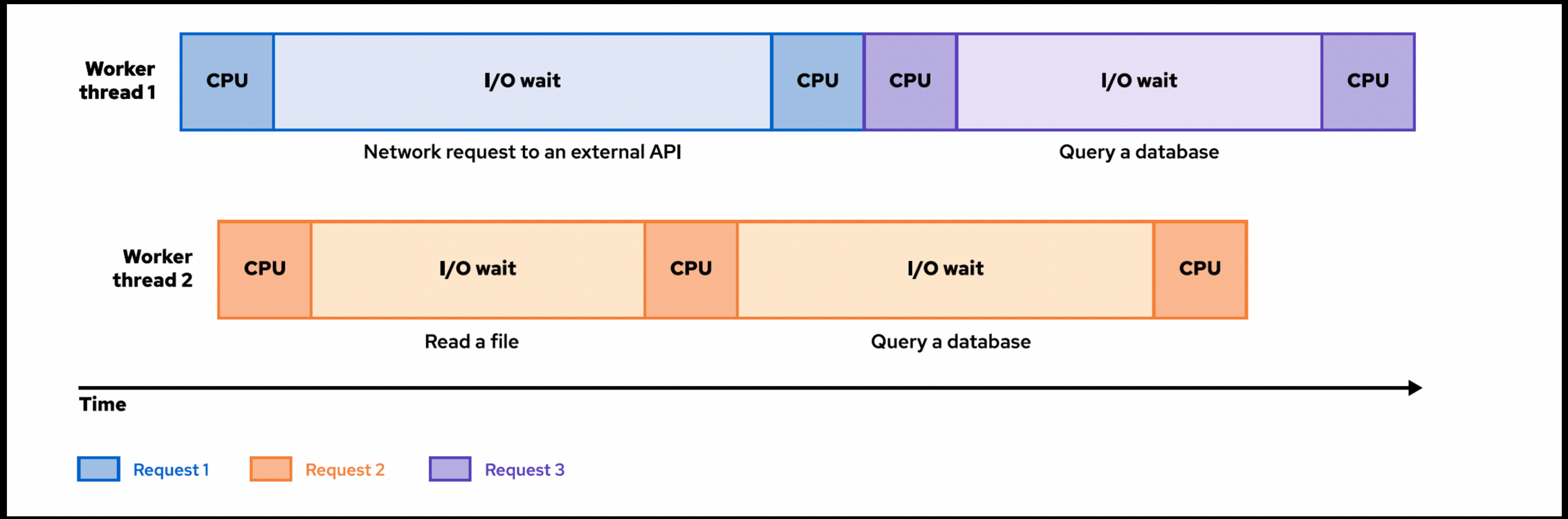
Blocking I/O


Request 1/
Worker thread 1
CPU
I/O wait
CPU
Network request to an external API
Request 2/
Worker thread 2
CPU
I/O wait
CPU
I/O wait
CPU
Read a file
Query a database
Time

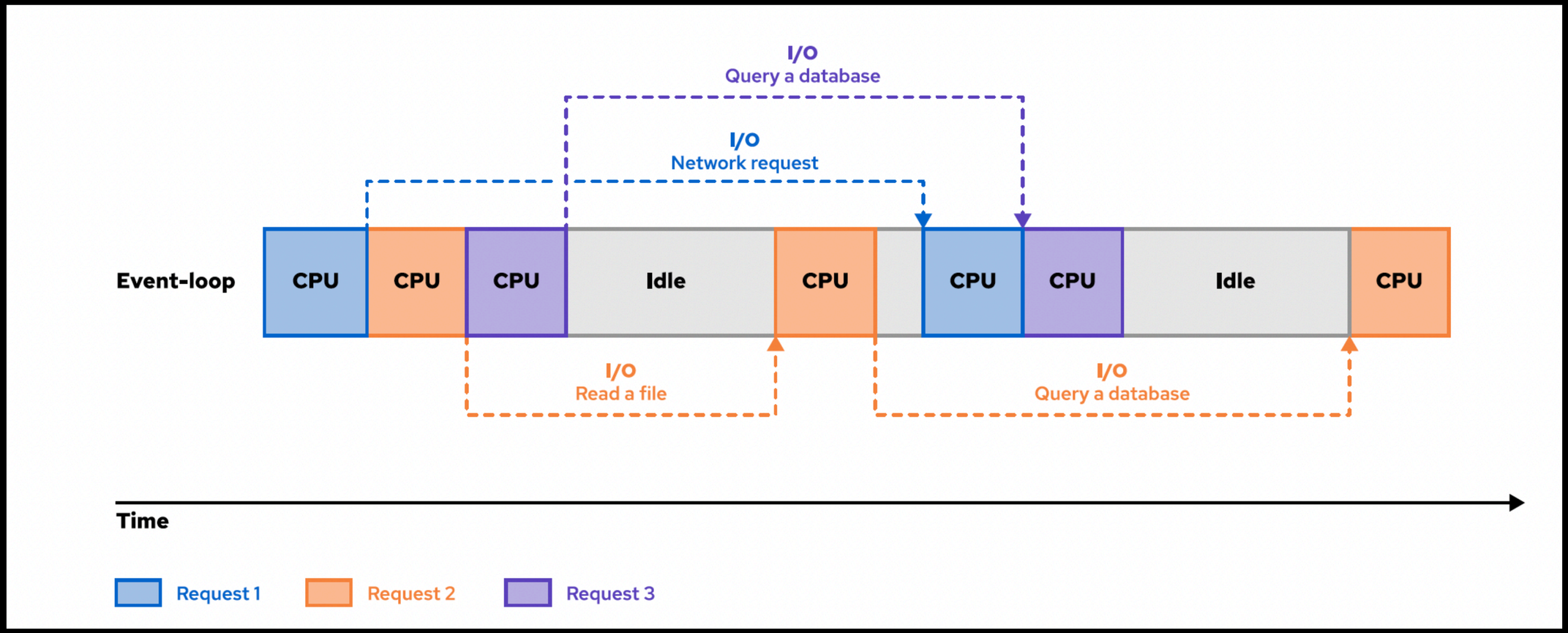
NO Blocking I/O



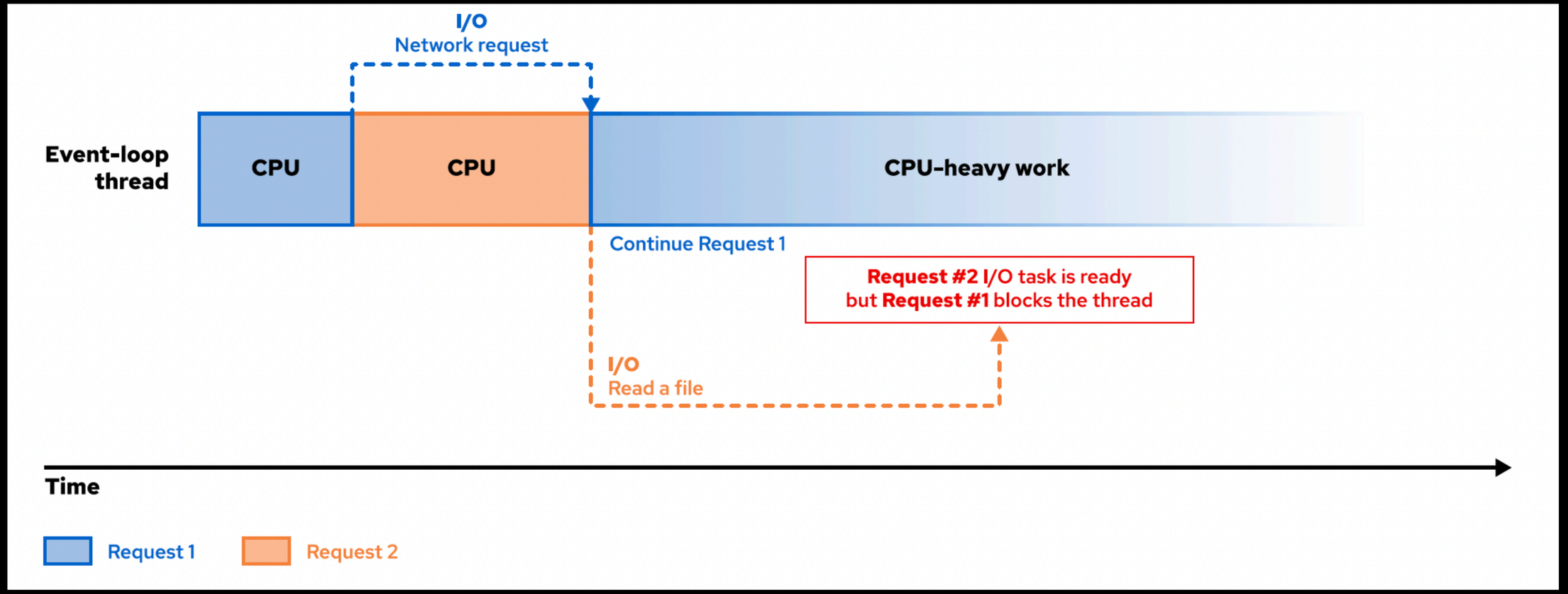
2 Worker Threads atendiendo 3 Requests



2 Event Loop Threads atendiendo 3 Requests



No bloquees el Event Loop



Extensiones Reactivas

```
<dependency>
   <groupId>io.quarkus</groupId>
   <artifactId>quarkus-resteasy-reactive</artifactId>
</dependency>
```

```
<dependency>
   <groupId>io.quarkus</groupId>
   <artifactId>quarkus-resteasy-reactive</artifactId>
</dependency>
```

Important

The RESTEasy Reactive and classic extensions cannot coexist in the same project. When adding any of the RESTEasy Reactive extensions, remove the classic ones, such as `quarkus-resteasy` and `quarkus-resteasy-jackson`.

RESTEasy reactivo

¿Cómo se da cuenta que debe aplicar NO Blocking?

- `io.smallrye.mutiny.Uni`
- `io.smallrye.mutiny.Multi`
- `java.util.concurrent.CompletionStage`
- `org.reactivestreams.Publisher`

Quarkus lo va a ejecutar blocking o no blocking?

```
@GET
@Path( "/cart" )
public ShoppingCart getCart() {
   return storage.findCart();
}
```

```
INFO  [io.qua.htt.access-log] (executor-thread-X) ...
```

Aquí tenemos un endpoint no blocking

```
@GET
@Path( "/cart" )
public Uni<ShoppingCart> getCart() {
   Uni<ShoppingCart> cartStream = storage.findCartAsync();
   return cartStream;
}
```

```
INFO [io.qua.htt.access-log] (vert.x-eventloop-thread-X) ...
```

io.smallrye.common.annotation.Blocking

Indicates that a method is blocking. RESTEasy Reactive runs the method in a worker thread.

io.smallrye.common.annotation.NonBlocking

Indicates that a method is non-blocking. RESTEasy Reactive runs the method in an eventloop thread.

```
@GET
@Path( "/recommendations" )
public Uni<Set<Recommendations>> getRecommendations() {
  // This operation is computationally expensive
  computeRecommendations();

  Uni<Set<Recommendations>> recommendations = fetchRecommendations();

  return recommendations;
}
```

Note

If Vert.x detects that you are blocking the event-loop, then it throws an exception.

```
@GET
@Path( "/recommendations" )
@Blocking
public Uni<Set<Recommendations>> getRecommendations() {
   ...implementation omitted...
}
```

Extensiones Reactivas

```
<dependency>
    <groupId>io.quarkus</groupId>
    <artifactId>quarkus-rest-client-reactive</artifactId>
</dependency>
```

```
<dependency>
    <groupId>io.quarkus</groupId>
    <artifactId>quarkus-rest-client-reactive-jackson</artifactId>
</dependency>
```

Ejemplo:

```
@GET
@Path( "/users/{userId}" )
User getUser( @PathParam( "userId" ) final Long userId );
```

```
@GET
@Path( "/users/{userId}" )
Uni<User> getUser( @PathParam( "userId" ) final Long userId );
```

Recursos

Reactividad

- https://www.reactivemanifesto.org/
- https://quarkus.io/version/2.13/guides/quarkus-reactive-architecture
- https://quarkus.io/version/2.13/guides/resteasy-reactive
- https://smallrye.io/smallrye-mutiny/1.7.0/