Distributed Tracing in Practice

Priit Potter | Plumbr

What is Plumbr?

- Real User Monitoring +
  Application Performance Monitoring
- Tracks all user interactions in the front-end
- Detects bottlenecks and errors in the back-end and maps to real user experience

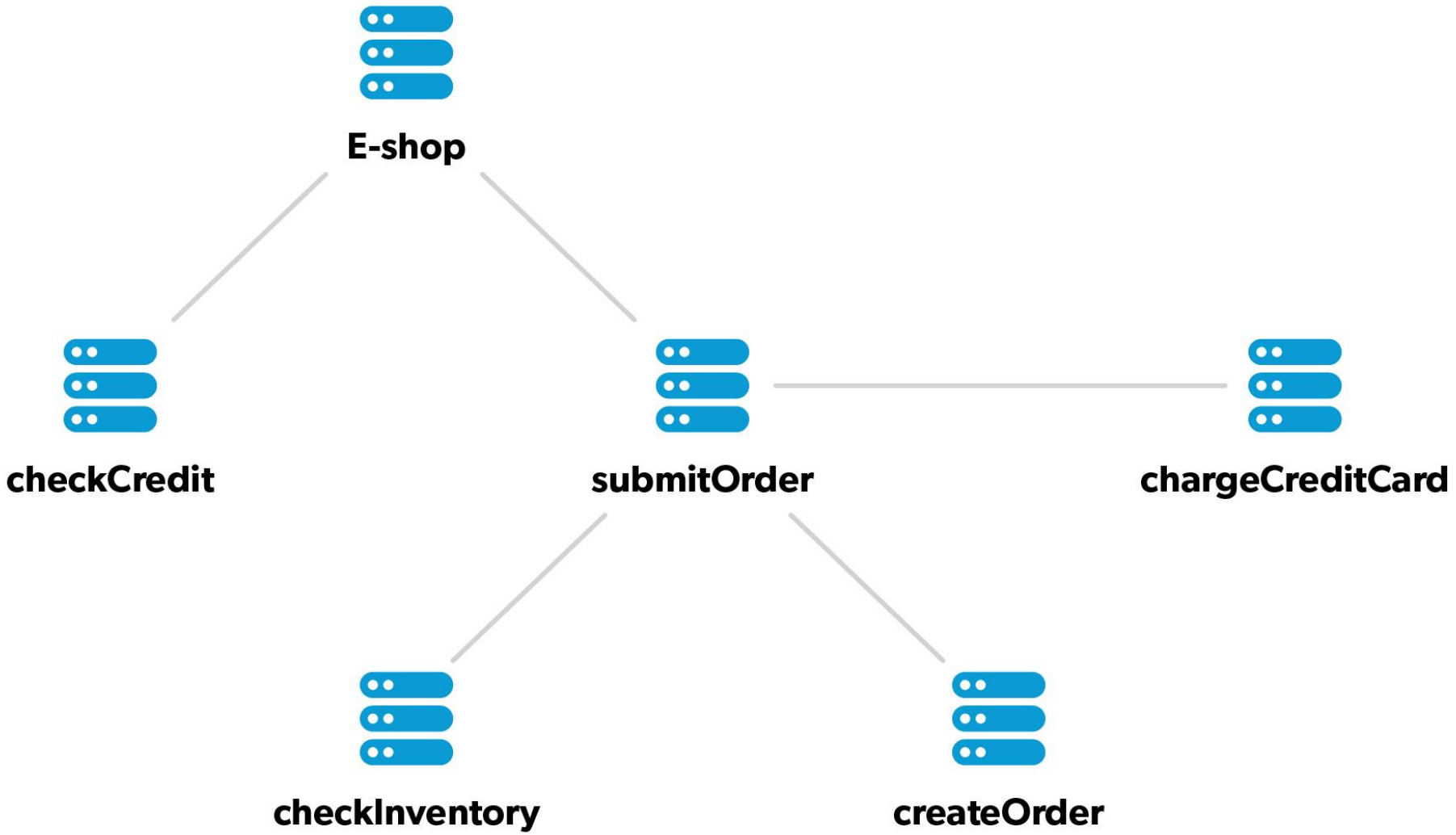
Distributed tracing, also called distributed request tracing, is a method used to profile and monitor applications, especially those built using a microservices architecture.

Source: Opentracing manual

Monolithic vs Microservices

Monolithic

Microservices

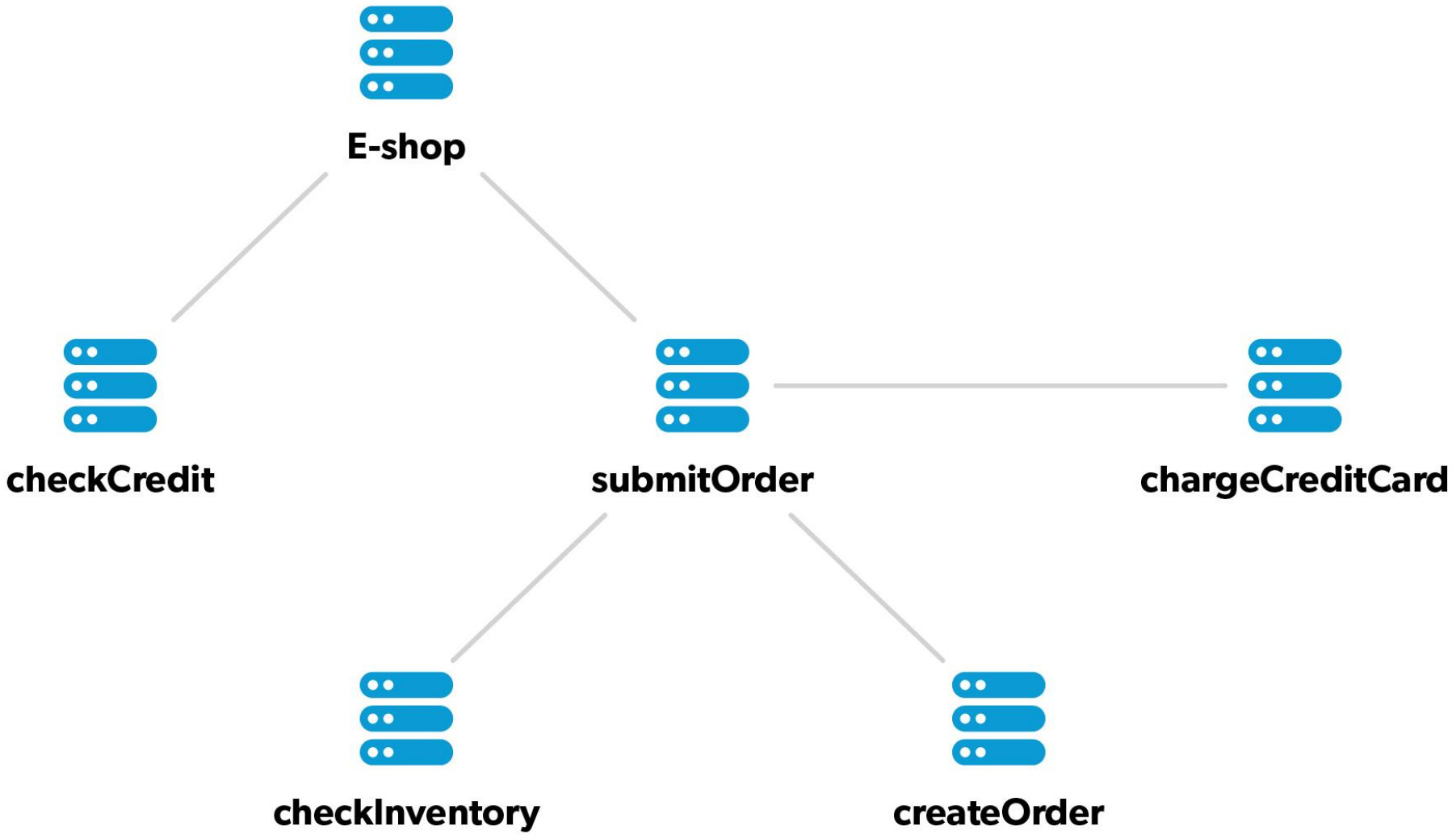

E-shop
checkCredit
submitOrder
chargeCreditCard
checkInventory
createOrder

From: **John The Customer**

To: **support@example.com**

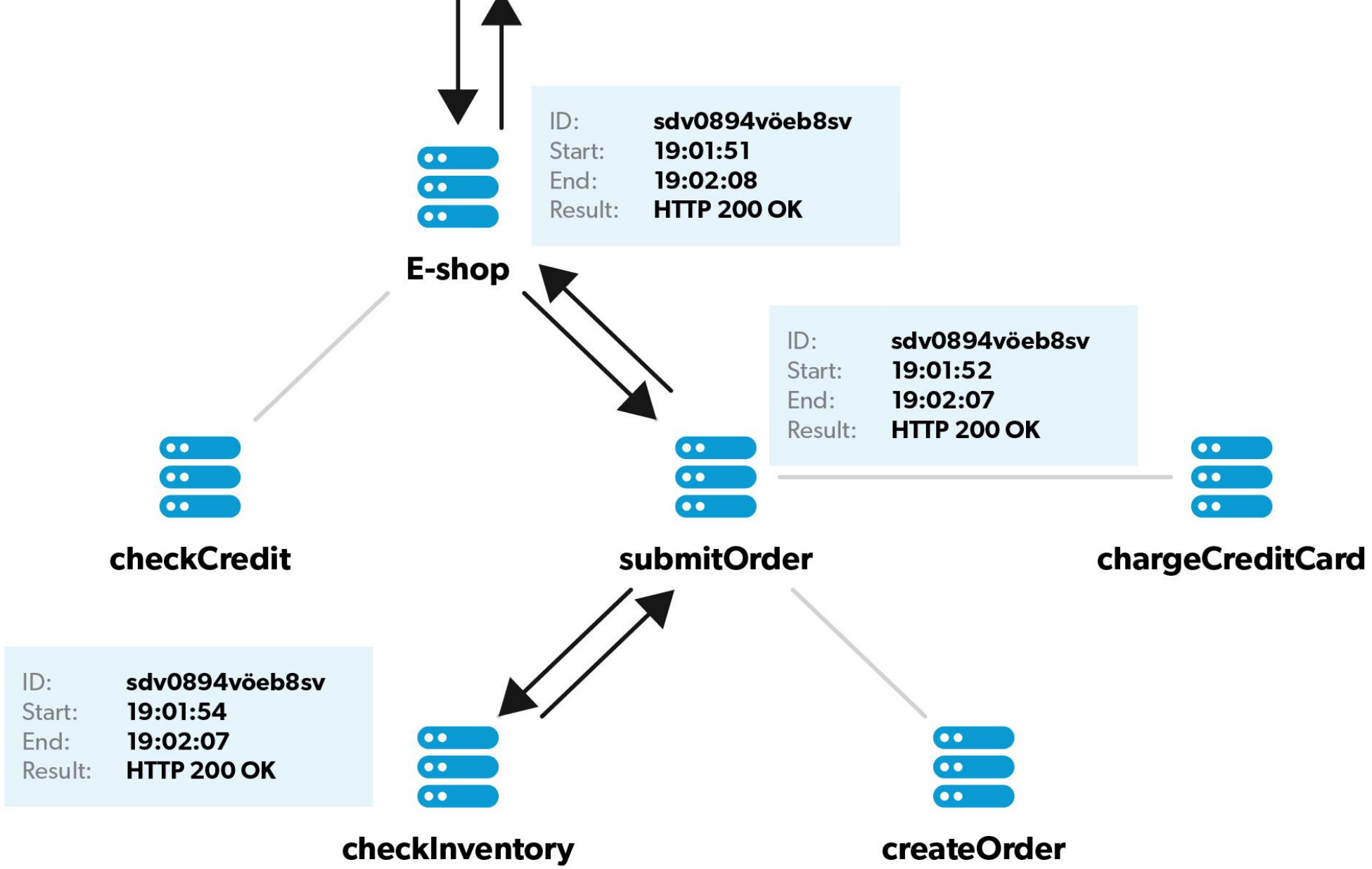
Subject: **Cannot complete checkout**

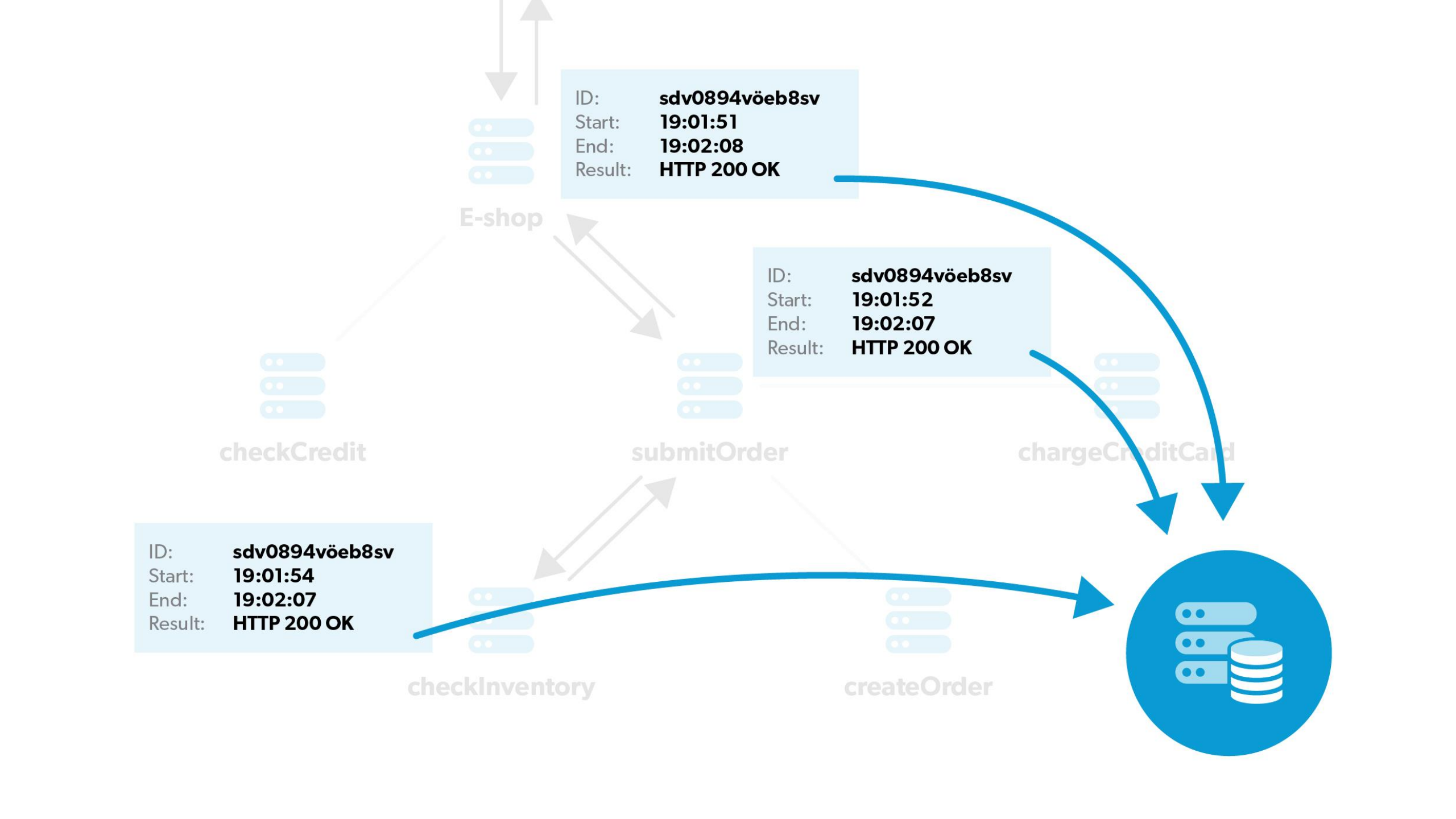
I just tried to complete my order #32828, but was unable to finish the checkout process. Your app stalled for 20 seconds and then gave me an error.


E-shop
checkCredit
submitOrder
chargeCreditCard
checkInventory
createOrder

The 4 rules of distributed tracing

- Every request gets a unique ID
- The ID is passed along to all microservices handling the request
- Every microservice registers the duration and outcome
- Monitoring data is sent to a central server

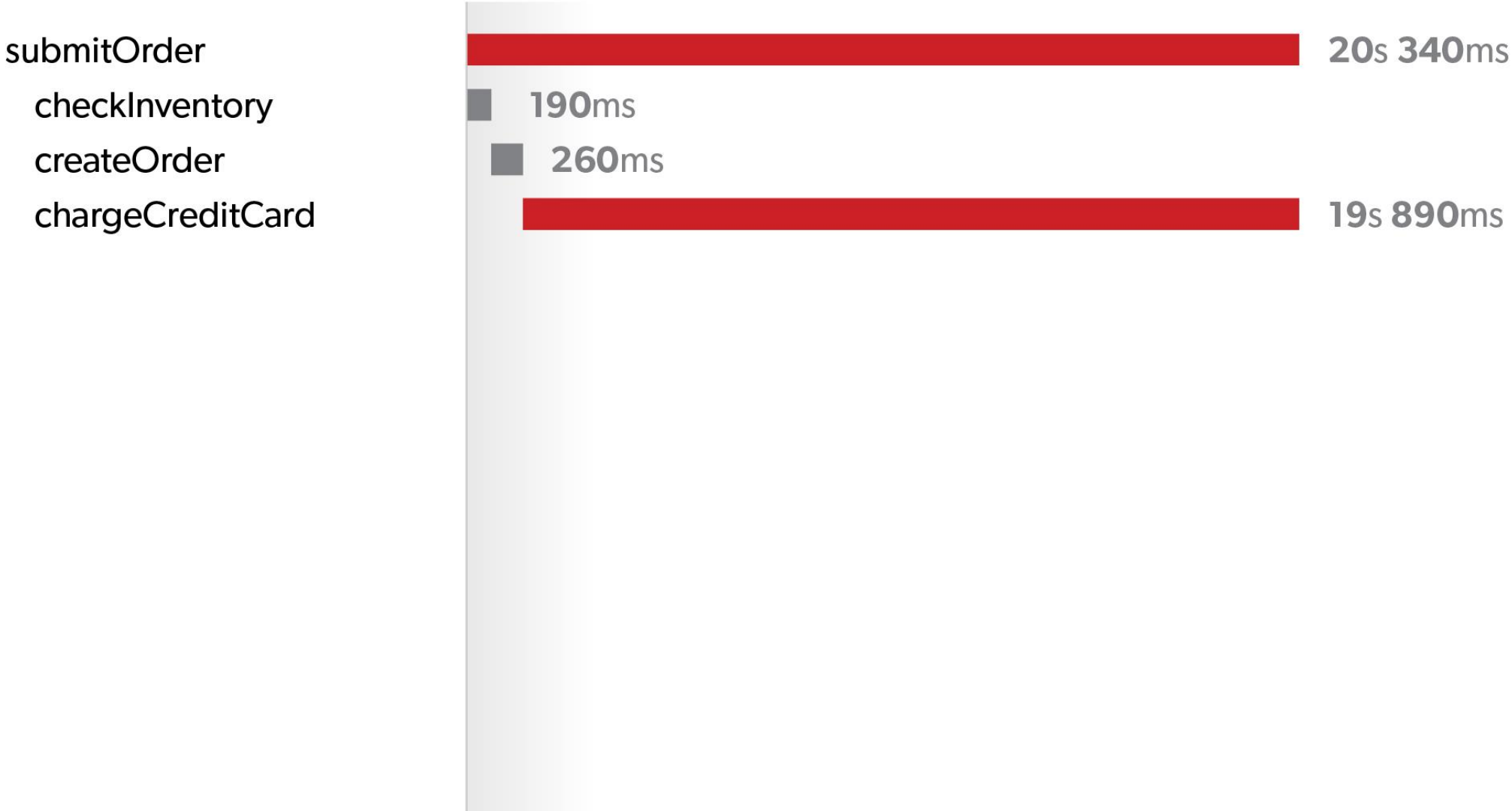

ID: sdv0894vöeb8sv
Start: 19:01:51
End: 19:02:08
Result: HTTP 200 OK
E-shop
ID: sdv0894vöeb8sv
Start: 19:01:52
End: 19:02:07
Result: HTTP 200 OK
checkCredit
submitOrder
chargeCreditCard
ID: sdv0894vöeb8sv
Start: 19:01:54
End: 19:02:07
Result: HTTP 200 OK
checkInventory
createOrder


ID: sdv0894vöeb8sv
Start: 19:01:51
End: 19:02:08
Result: HTTP 200 OK
E-shop
ID: sdv0894vöeb8sv
Start: 19:01:52
End: 19:02:07
Result: HTTP 200 OK
checkCredit
submitOrder
chargeCreditCard
ID: sdv0894vöeb8sv
Start: 19:01:54
End: 19:02:07
Result: HTTP 200 OK
checkInventory
createOrder

Example #1: response to our support ticket

Service: **submitOrder**

Start: 2019-06-17 **19:01** Duration: **20s 340ms**

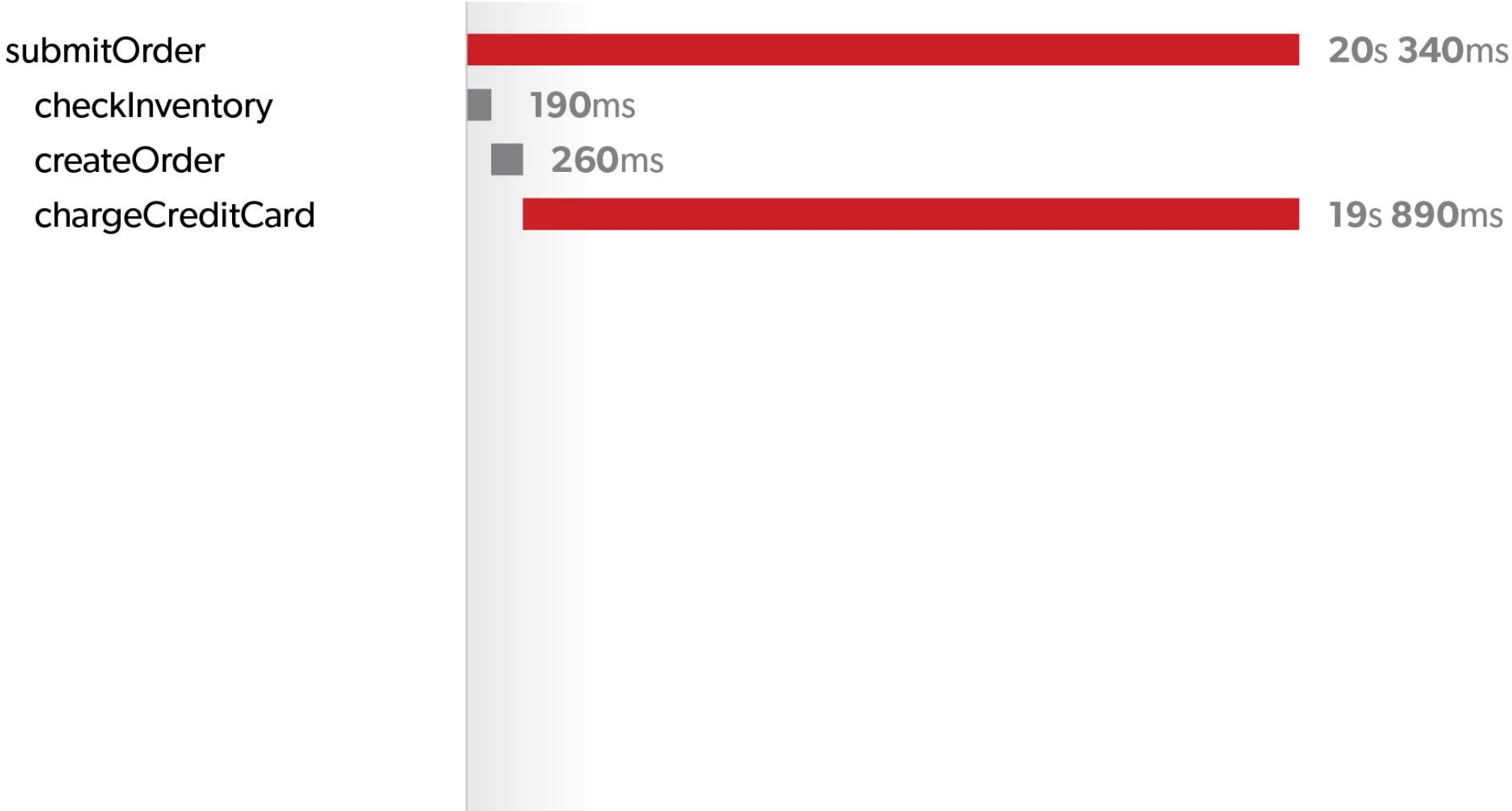


Lesson #1: a distributed application calls for distributed tracing :)

Example #2: smarter response to our support ticket

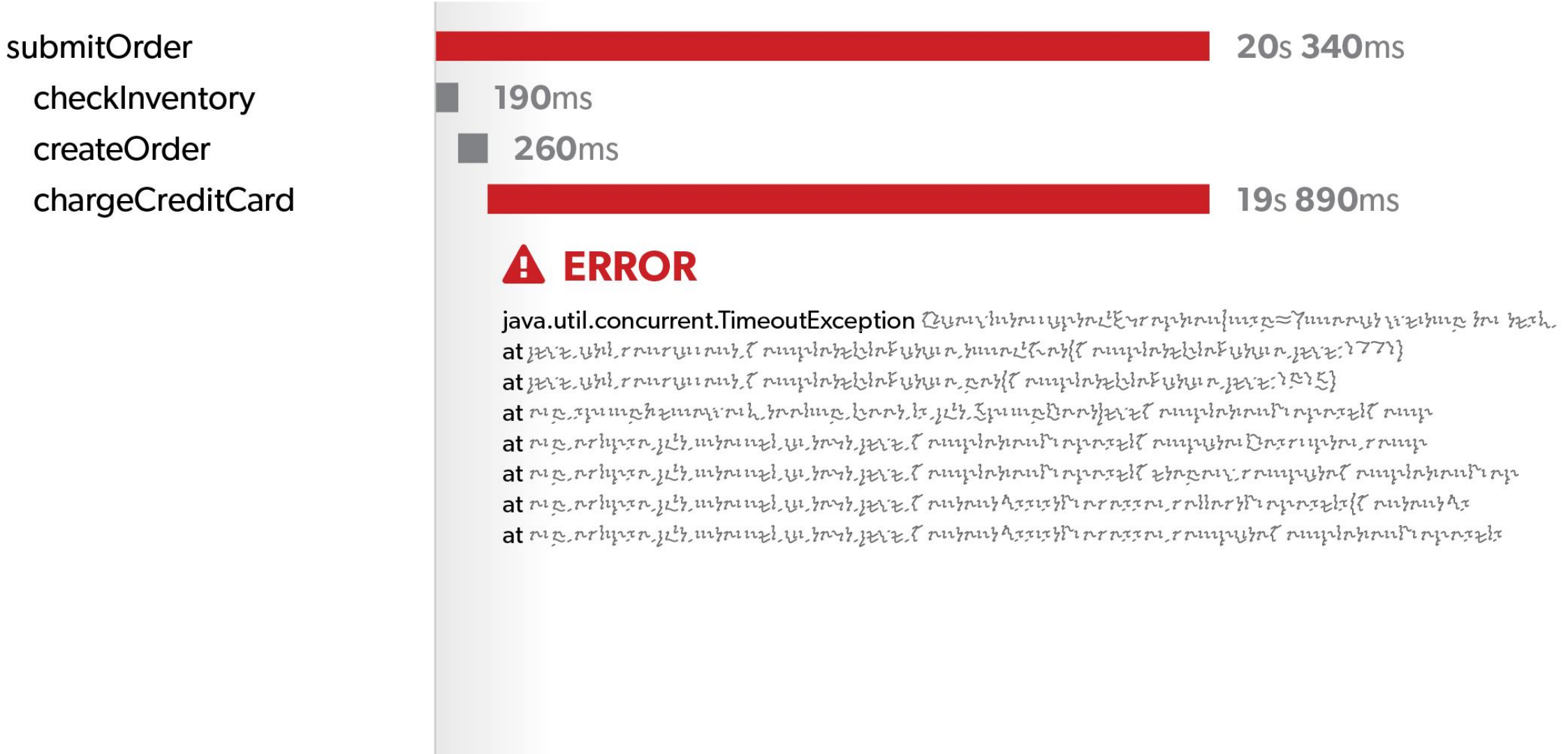
Service: **submitOrder**

Start: 2019-06-17 **19:01** Duration: **20s 340ms**



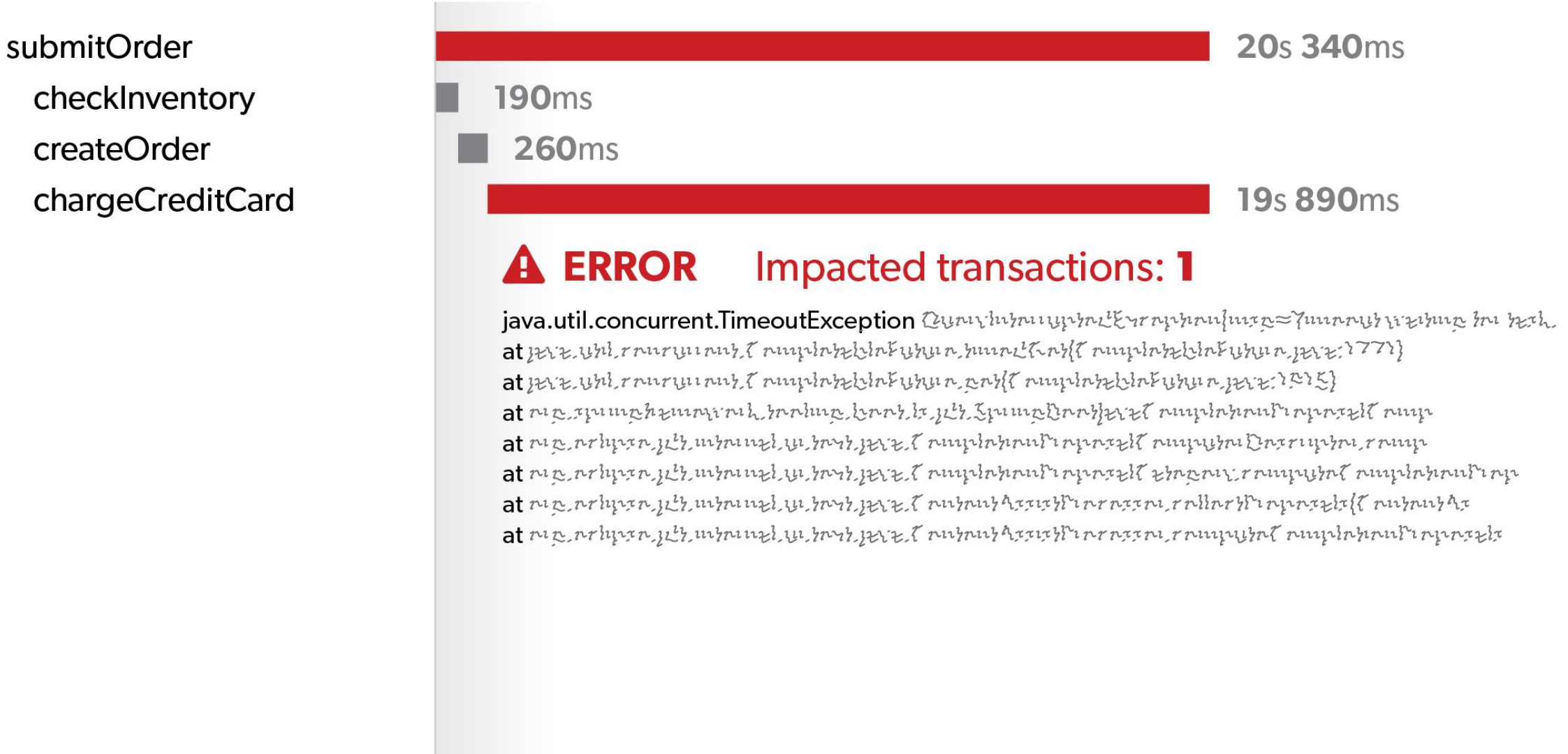
Service: **submitOrder**

Start: 2019-06-17 **19:01** Duration: **20s 340ms**



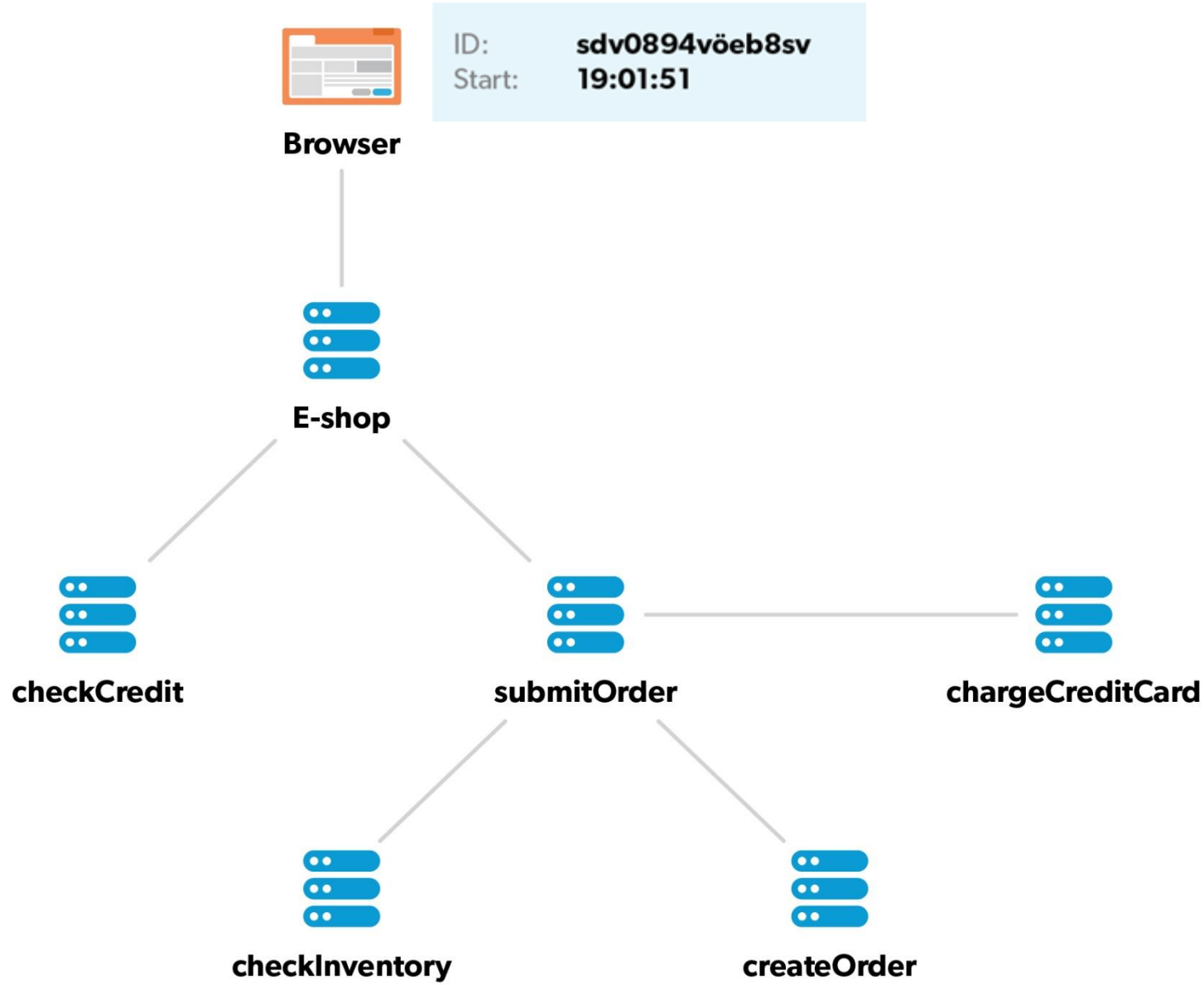
Service: **submitOrder**

Start: 2019-06-17 **19:01** Duration: **20s 340ms**



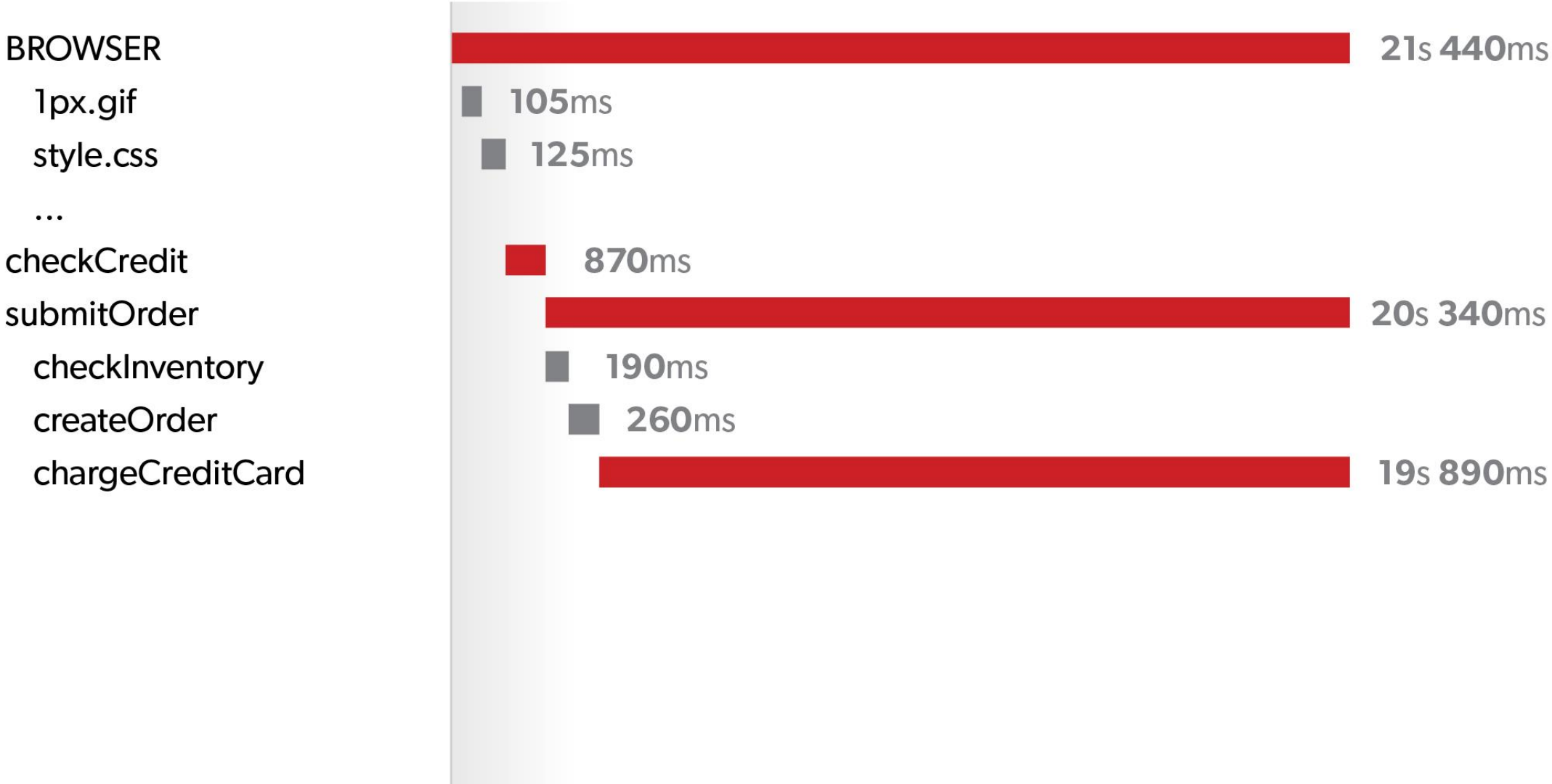
Lesson #2: for faster impact analysis, add error/bottleneck information to traces

Example #3: the smartest response to our support ticket


ID: sdv0894vöeb8sv
Start: 19:01:51
Browser
E-shop
checkCredit
submitOrder
chargeCreditCard
checkInventory
createOrder

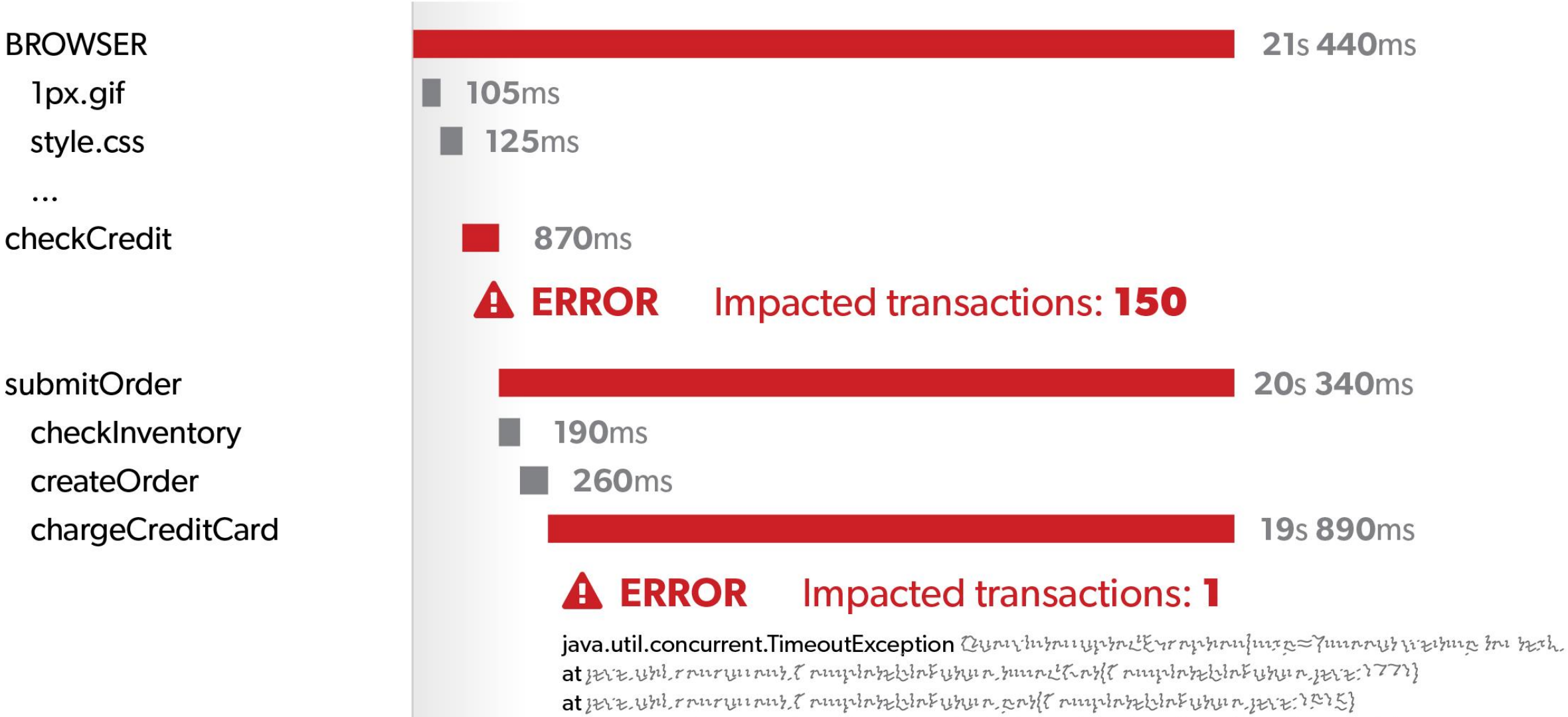
Service: **submitOrder**

Start: 2019-06-17 **19:01** Duration: **21s 440ms**



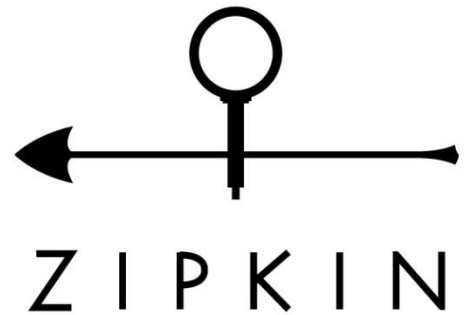
Service: **submitOrder**

Start: 2019-06-17 **19:01** Duration: **21s 440ms**



Lesson #3: for most efficient impact analysis, include frontend traces too


OPENTRACING


ZIPKIN

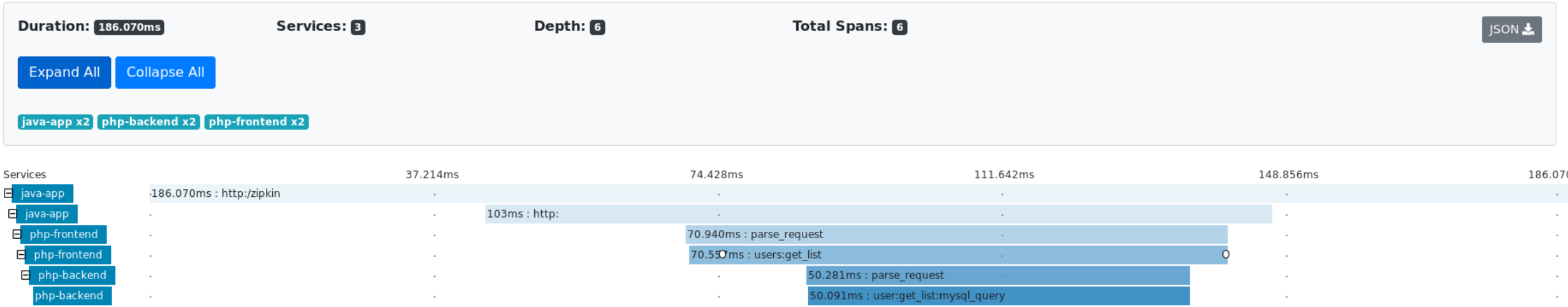

JAEGER


OpenCensus

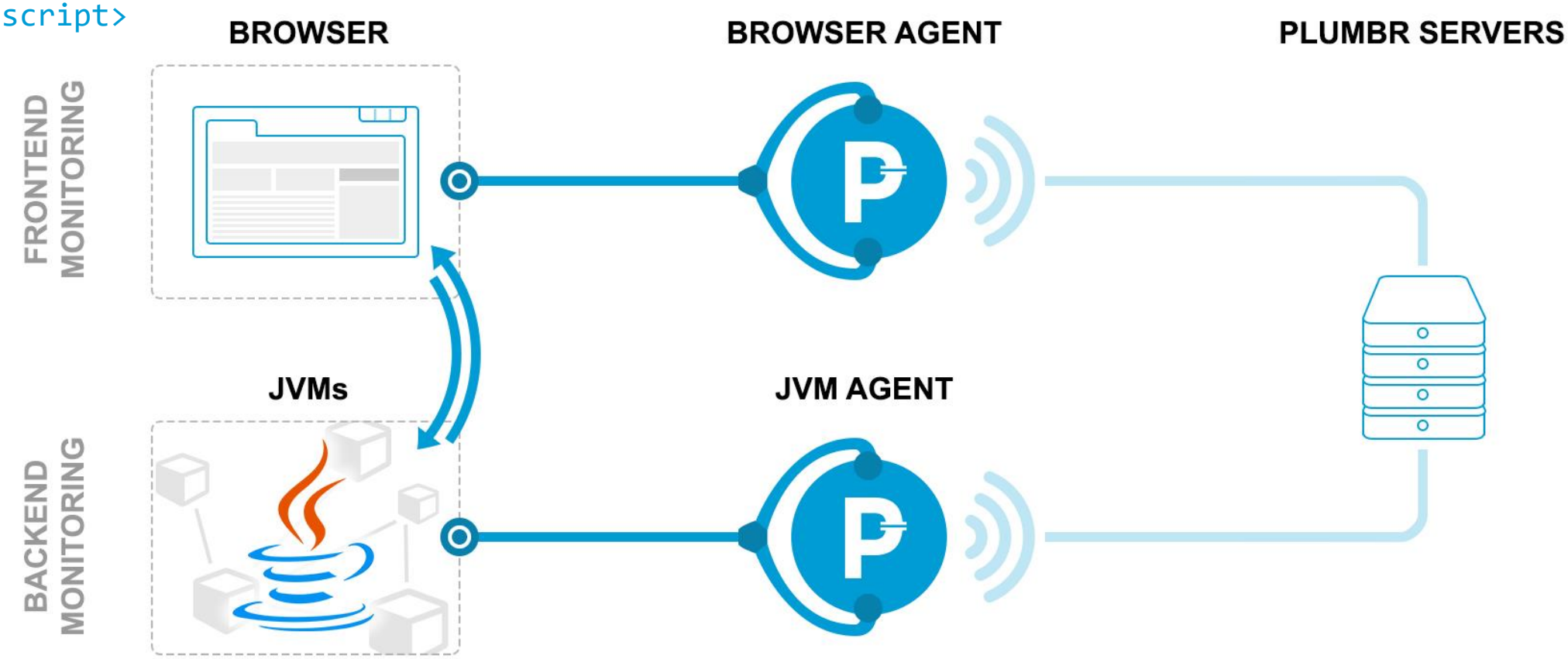
Zipkin PHP example

```
$tracing = create_tracing('php-frontend', '127.0.0.1');
$tracer = $tracing->getTracer();
$request = \Symfony\Component\HttpFoundation\Request::createFromGlobals();

/* Extract the context from HTTP headers */
$carrier = array_map(function ($header) {
      return $header[0];
}, $request->headers->all());
$extractor = $tracing->getPropagation()->getExtractor(new Map());
$extractedContext = $extractor($carrier);

/* Create a span and set its attributes */
$span = $tracer->newChild($extractedContext);
$span->start(Timestamp\now());
$span->setName('parse_request');
$span->setKind(Zipkin\Kind\SERVER);
```

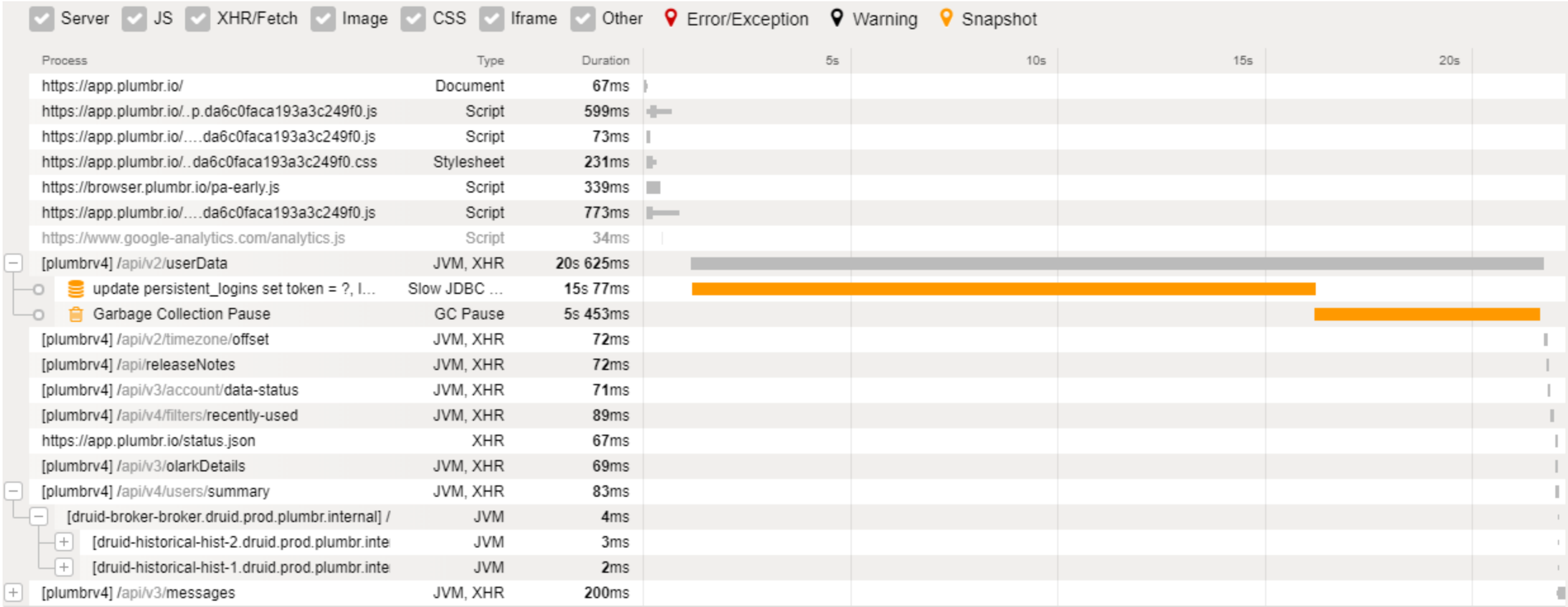
Zipkin PHP example



```
<script src="https://browser.plumbr.io/pa.js" crossorigin="anonymous"
  data-plumbr='{
    "accountId":"k8g0ta8m6e7gsb2afhq6qm5efh",
    "appName":"REPLACE_WITH_YOUR_APPLICATION_NAME",
    "serverUrl":"https://bdr.plumbr.io"
  }'>
</script>
```




```
-javaagent:path-to/plumbr.jar
```


Server
JS
XHR/Fetch
Image
CSS
Iframe
Other
Error/Exception
Warning
Snapshot
Process
Type
Duration
5s
10s
15s
20s
https://app.plumbr.io/ Document 67ms
https://app.plumbr.io/..p.da6c0faca193a3c249f0.js Script 599ms
https://app.plumbr.io/....da6c0faca193a3c249f0.js Script 73ms
https://app.plumbr.io/..da6c0faca193a3c249f0.css Stylesheet 231ms
https://browser.plumbr.io/pa-early.js Script 339ms
https://app.plumbr.io/....da6c0faca193a3c249f0.js Script 773ms
https://www.google-analytics.com/analytics.js Script 34ms
[plumbrv4] /api/v2/userData JVM, XHR 20s 625ms
update persistent_logins set token = ?, l... Slow JDBC ... 15s 77ms
Garbage Collection Pause GC Pause 5s 453ms
[plumbrv4] /api/v2/timezone/offset JVM, XHR 72ms
[plumbrv4] /api/releaseNotes JVM, XHR 72ms
[plumbrv4] /api/v3/account/data-status JVM, XHR 71ms
[plumbrv4] /api/v4/filters/recently-used JVM, XHR 89ms
https://app.plumbr.io/status.json XHR 67ms
[plumbrv4] /api/v3/olarkDetails JVM, XHR 69ms
[plumbrv4] /api/v4/users/summary JVM, XHR 83ms
[druid-broker-broker.druid.prod.plumbr.internal] / JVM 4ms
[druid-historical-hist-2.druid.prod.plumbr.inte JVM 3ms
[druid-historical-hist-1.druid.prod.plumbr.inte JVM 2ms
[plumbrv4] /api/v3/messages JVM, XHR 200ms