APM Agents
Node.JS vs Java

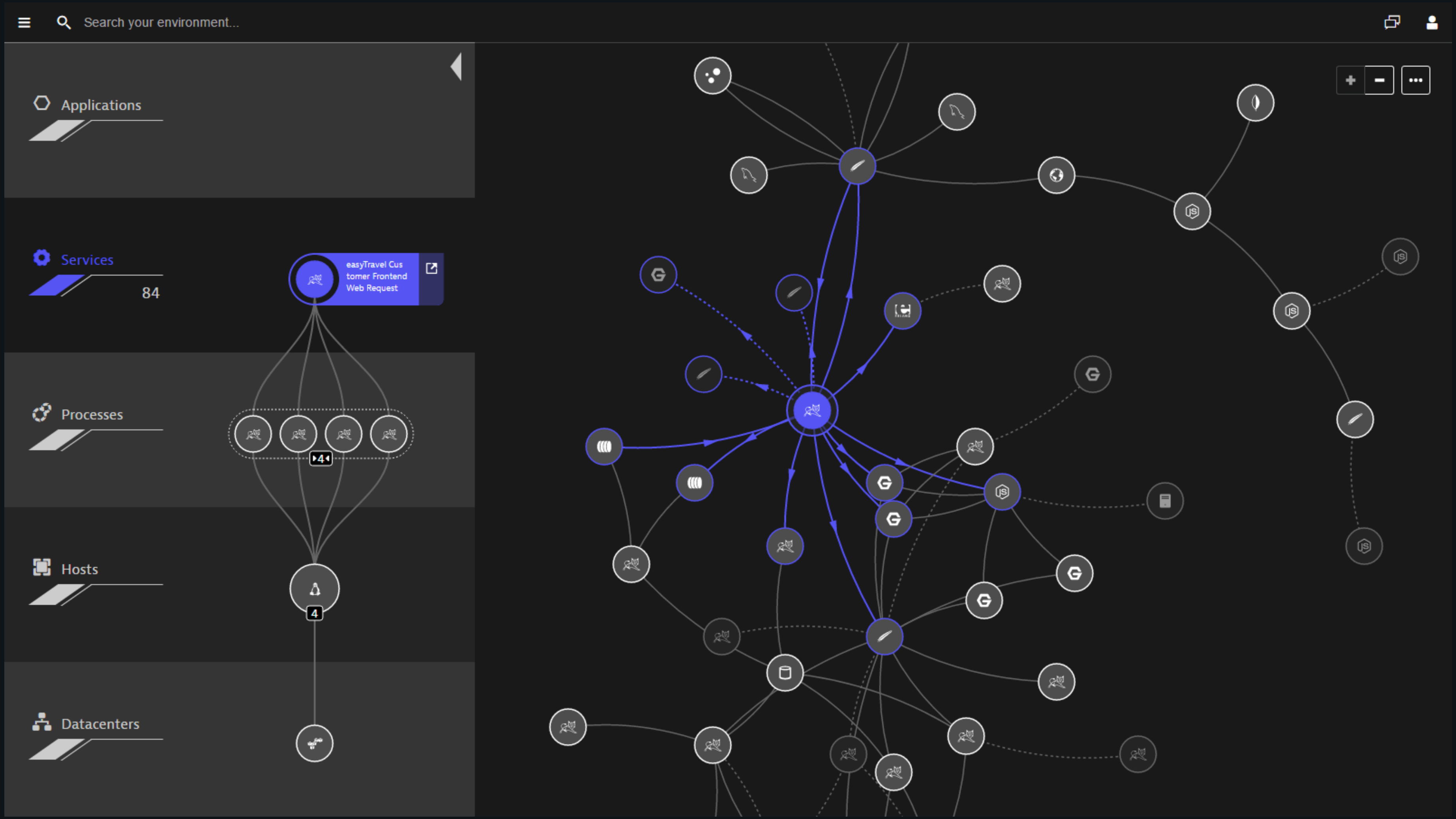
Compare Java and Node.JS tooling related to APM Agents

dynatrace

Introduction – APM

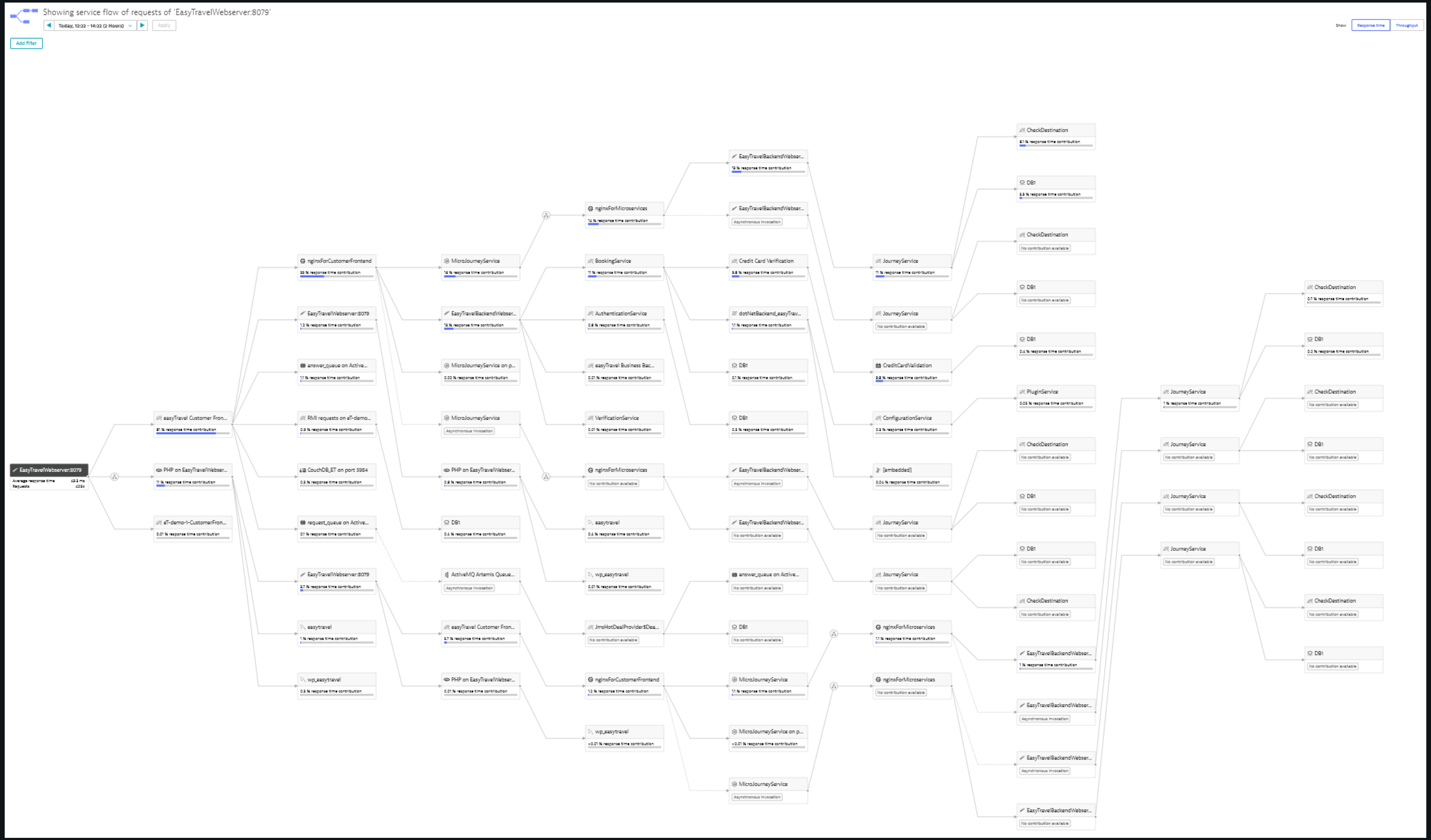
- Agents are injected into customer applications in production
- Agents continuously report data
  - Metrics like CPU/Memory usage, GC time, Event Loop timing,…
  - Transactions (focus on topology, duration and CPU time)
  - References to source code (stack traces, linked with transactions / background)
  - Low resolution CPU Profiling (hotspots on transactions / background)
- Agents add/extract tags to link transactions between processes (“Trace Context”)
- Agent shall not change application behavior
- Servers correlate agent data and visualize it (e.g. topology, detected problems,…)

Infrastructure / Topology



Search your environment...
Applications
Services
84
easyTravel Customer Frontend Web Request
Processes
4
Hosts
4
Datacenters

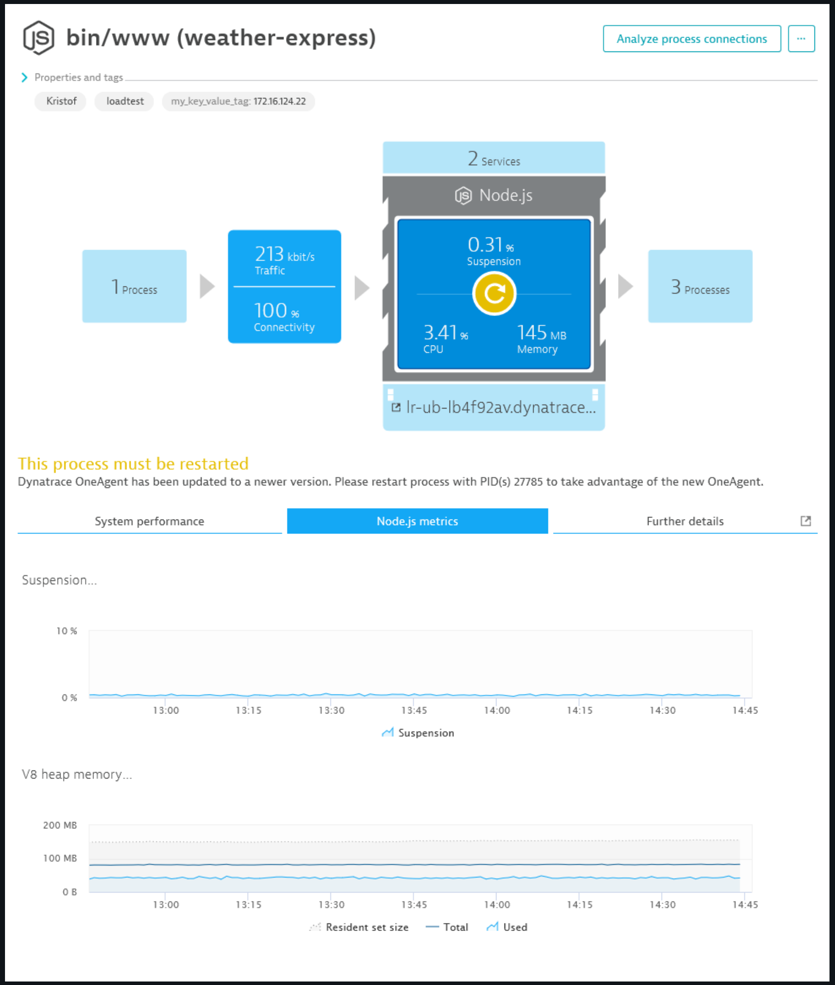
Service Flow



Showing service flow of requests of 'EasyTravelWebserver:8079'

Today, 12:22 - 14:22 (2 Hours)
Apply
Add filter
Show
Response time
Throughput
EasyTravelWebserver:8079
Average response time 49.3 ms
Requests 405k
easyTravel Customer Fron...
81 % response time contribution
PHP on EasyTravelWebser...
11 % response time contribution
eT-demo-1-CustomerFron...
0.01 % response time contribution
nginxForCustomerFrontend
33 % response time contribution
EasyTravelWebserver:8079
1.2 % response time contribution
answer_queue on Active...
1.1 % response time contribution
RMI requests on eT-demo...
0.9 % response time contribution
CouchDB_ET on port 5984
0.3 % response time contribution
request_queue on Active...
0.1 % response time contribution
EasyTravelWebserver:8079
3.7 % response time contribution
easytravel
1 % response time contribution
wp_easytravel
0.3 % response time contribution
MicroJourneyService
16 % response time contribution
EasyTravelBackendWebser...
13 % response time contribution
MicroJourneyService on p...
0.02 % response time contribution
MicroJourneyService
Asynchronous Invocation
PHP on EasyTravelWebser...
0.8 % response time contribution
DB1
0.4 % response time contribution
ActiveMQ Artemis Queue...
Asynchronous Invocation
easyTravel Customer Fron...
3.7 % response time contribution
PHP on EasyTravelWebser...
0.01 % response time contribution
nginxForMicroservices
14 % response time contribution
BookingService
11 % response time contribution
AuthenticationService
0.8 % response time contribution
easyTravel Business Bac...
0.01 % response time contribution
VerificationService
0.01 % response time contribution
nginxForMicroservices
No contribution available
easytravel
0.4 % response time contribution
wp_easytravel
0.01 % response time contribution
JmsHotDealProvider$Dea...
No contribution available
nginxForCustomerFrontend
1.3 % response time contribution
wp_easytravel
<0.01 % response time contribution
EasyTravelBackendWebser...
13 % response time contribution
EasyTravelBackendWebser...
Asynchronous Invocation
Credit Card Verification
9.8 % response time contribution
dotNetBackend_easyTrav...
1.1 % response time contribution
DB1
0.1 % response time contribution
DB1
0.5 % response time contribution
EasyTravelBackendWebser...
Asynchronous Invocation
EasyTravelBackendWebser...
No contribution available
answer_queue on Active...
No contribution available
DB1
No contribution available
MicroJourneyService
1.1 % response time contribution
MicroJourneyService on p...
<0.01 % response time contribution
MicroJourneyService
Asynchronous Invocation
JourneyService
11 % response time contribution
JourneyService
No contribution available
CreditCardValidation
9.8 % response time contribution
ConfigurationService
0.3 % response time contribution
[embedded]
0.04 % response time contribution
JourneyService
No contribution available
JourneyService
No contribution available
nginxForMicroservices
1.1 % response time contribution
nginxForMicroservices
No contribution available
CheckDestination
8.1 % response time contribution
DB1
2.3 % response time contribution
CheckDestination
No contribution available
DB1
No contribution available
DB1
0.4 % response time contribution
PluginService
0.05 % response time contribution
CheckDestination
No contribution available
DB1
No contribution available
DB1
No contribution available
CheckDestination
No contribution available
EasyTravelBackendWebser...
1 % response time contribution
EasyTravelBackendWebser...
Asynchronous Invocation
EasyTravelBackendWebser...
Asynchronous Invocation
EasyTravelBackendWebser...
No contribution available
JourneyService
1 % response time contribution
JourneyService
No contribution available
JourneyService
No contribution available
JourneyService
No contribution available
CheckDestination
0.7 % response time contribution
DB1
0.2 % response time contribution
CheckDestination
No contribution available
DB1
No contribution available
CheckDestination
No contribution available
DB1
No contribution available
CheckDestination
No contribution available
DB1
No contribution available

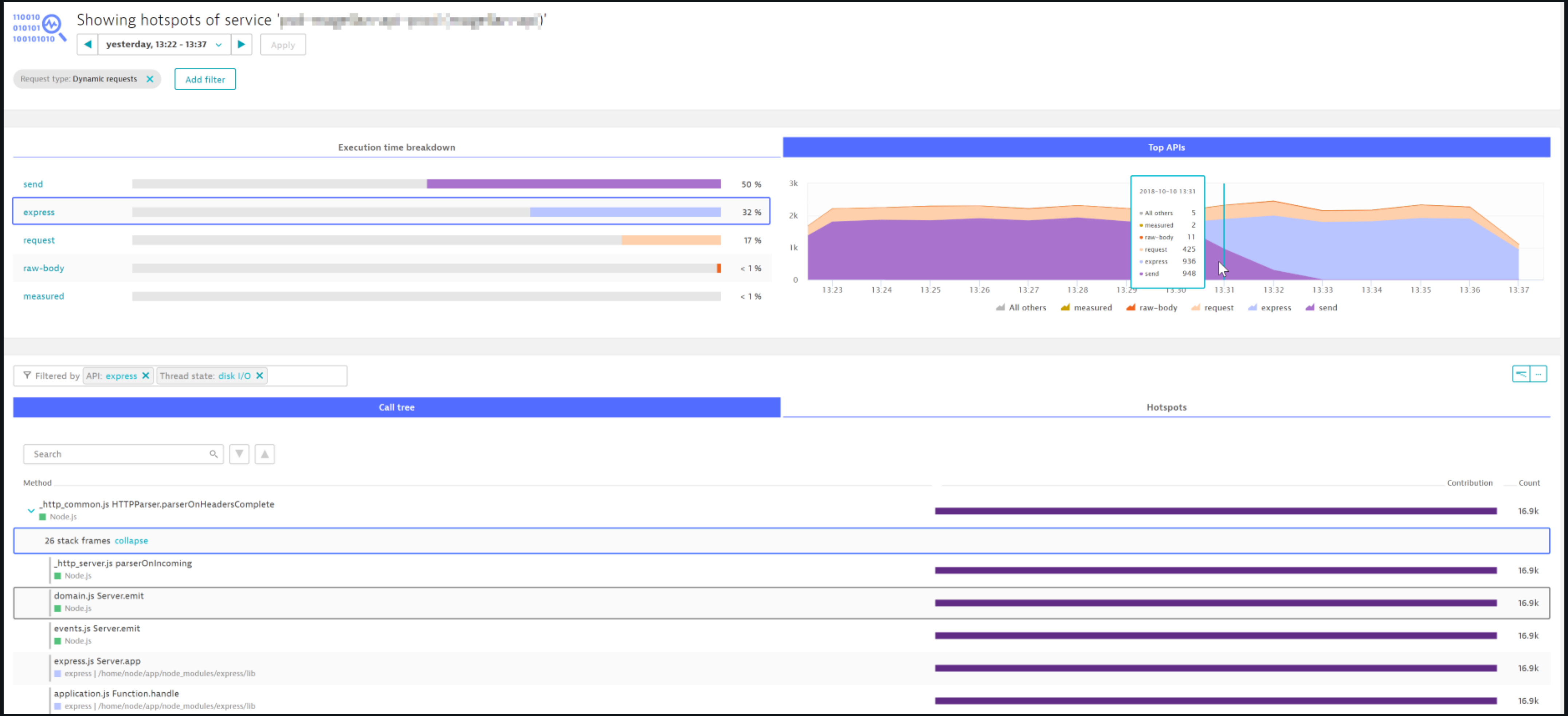
Process Details including Metrics



This process must be restarted

Dynatrace OneAgent has been updated to a newer version. Please restart process with PID(s) 27785 to take advantage of the new OneAgent.

CPU Hotspots (Service / Background)



Showing hotspots of service '[redacted])'

◀ yesterday, 13:22 - 13:37 ⌄ ▶ Apply

Request type: Dynamic requests ✕ Add filter

Execution time breakdown

API	Share
send	50 %
express	32 %
request	17 %
raw-body	< 1 %
measured	< 1 %

Top APIs

2018-10-10 13:31

API	Value
All others	5
measured	2
raw-body	11
request	425
express	936
send	948

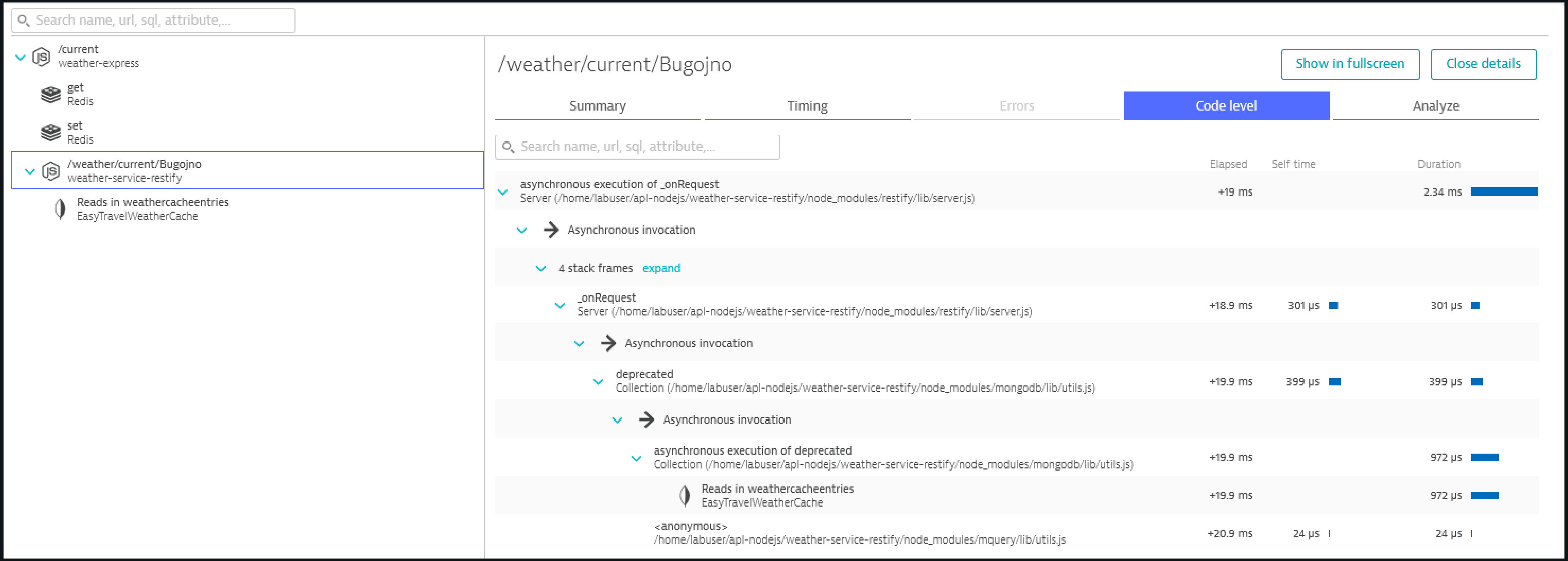
Filtered by API: express ✕ Thread state: disk I/O ✕

Call tree | Hotspots

Search

Method	Contribution	Count
_http_common.js HTTPParser.parserOnHeadersComplete Node.js		16.9k
26 stack frames collapse		
_http_server.js parserOnIncoming Node.js		16.9k
domain.js Server.emit Node.js		16.9k
events.js Server.emit Node.js		16.9k
express.js Server.app express | /home/node/app/node_modules/express/lib		16.9k
application.js Function.handle express | /home/node/app/node_modules/express/lib		16.9k

Transactions with code level visibility


Search name, url, sql, attribute,...
Show in fullscreen
Close details
Summary
Timing
Errors
Code level
Analyze
Search name, url, sql, attribute,...

/current
weather-express

get
Redis

set
Redis

/weather/current/Bugojno
weather-service-restify

Reads in weathercacheentries
EasyTravelWeatherCache

/weather/current/Bugojno

	Elapsed	Self time	Duration
asynchronous execution of _onRequest Server (/home/labuser/apl-nodejs/weather-service-restify/node_modules/restify/lib/server.js)	+19 ms		2.34 ms
Asynchronous invocation			
4 stack frames expand			
_onRequest Server (/home/labuser/apl-nodejs/weather-service-restify/node_modules/restify/lib/server.js)	+18.9 ms	301 µs	301 µs
Asynchronous invocation			
deprecated Collection (/home/labuser/apl-nodejs/weather-service-restify/node_modules/mongodb/lib/utils.js)	+19.9 ms	399 µs	399 µs
Asynchronous invocation			
asynchronous execution of deprecated Collection (/home/labuser/apl-nodejs/weather-service-restify/node_modules/mongodb/lib/utils.js)	+19.9 ms		972 µs
Reads in weathercacheentries EasyTravelWeatherCache	+19.9 ms		972 µs
<anonymous> /home/labuser/apl-nodejs/weather-service-restify/node_modules/mquery/lib/utils.js	+20.9 ms	24 µs	24 µs

Agent initialization/instrumentation

Node.JS	Java
No dedicated APIs to load an agent and instrument code	Standardized tooling API (JVMTI)
Loaded via `require()` as *first* statement in application (or via `-r` command line option)	Dedicated option `(-agentpath)` to load and initialize agent
Monkey Patch exports by intercepting `module._load()`	Modify Byte Code during class loading (JVMTI hook)
Internals/implementation details needed	Internals/implementation details needed
No generic database/metric interfaces	Generic interfaces for databases (JDBC) and metrics (JMX)
Use of public APIs to get e.g. metrics (GC, …)	Use of public APIs to get e.g. metrics (GC, …)
Native ABI stable within a major Node.JS version No diagnostics/debugging/… APIs in N-API	ABI stable since Java 5 (but extended since then)

Monkey Patching vs Byte Code modifications

Node.JS	Java
Monkey Patching is frequently used - also by a lot user space modules	Byte Code instrumentation is usually not used by application programmers
No access to internal core modules No access to file local functions Non configurable properties avoid monkey patching Private symbols, private fields, … Exported functions may be cached inside the file (“partial” instrumentation)	Full access to all loaded classes including inner classes, private members,… Even hot modification is possible
Agent and application frames intermixed on call stack	Agent frames on call stack only as leaf

Code Level visibility / hotspot detection / Profiling

Node.JS	Java
No API to get a single JS call stack from a different thread	API available to capture a stack trace from any thread Allows to add context to stack traces (e.g. transaction) Allows to monitor overhead cause by agent
CPU Profiler included (start/stop, result is a tree)	Profiler can be implemented based on above API
Anonymous functions are very common but have mostly no context when seen on a call stack Inferred name is hard to get and sometimes misleading	Anonymous functions (lambdas) are always part of a class and usually implement an interface
Async IO hides a lot of the transactional context (e.g. no info about Cpu cycles used inside LibUv) Use of `nextTick`/… spreads CPU time/wait time Async Hooks available for tracking (at least in core)	IO often synchronous therefore stack sampling shows where time is spent no framework similar to Async Hooks (instrumentation of threading APIs needed – part of JVM)

Memory Profiling, Stack Traces

Node.JS	Java
Complete Memory Profiler included (Sampling Heap Profiler intended for production use)	JVMTI API to install a sync callback for allocations after every ~x Bytes (since Java 11)
No API to iterate function objects on stacks (strict mode) Function identities are lost as name/script/line can’t be correlated to the concreate function instance	Iterate function/classes objects on stacks

Summary

- Java offers standardized and more generic low level APIs with focus on tool development
- Node.JS offers ready to use functionality for specific use cases
- Strongly typed nature of Java offers benefits to identify/map functions in source code
- Sync nature of a lot Java APIs results in easier monitoring/correlation
- Production readiness (Stability/Overhead) not clearly documented for some Node.JS features
- Node.JS core is much smaller then Java, a lot userland code with duplications (e.g. Promise libs) and independent versioning

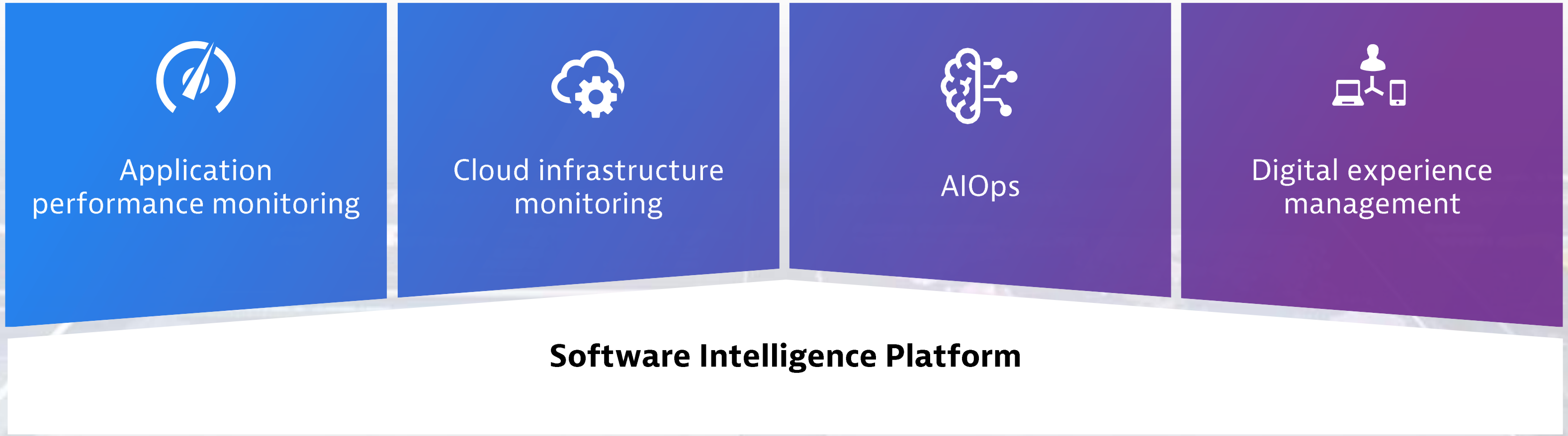
Questions ???

Software intelligence for the enterprise cloud

dynatrace

Software intelligence built for the enterprise cloud

Go beyond APM with the Dynatrace all-in-one platform



Application performance monitoring

Cloud infrastructure monitoring

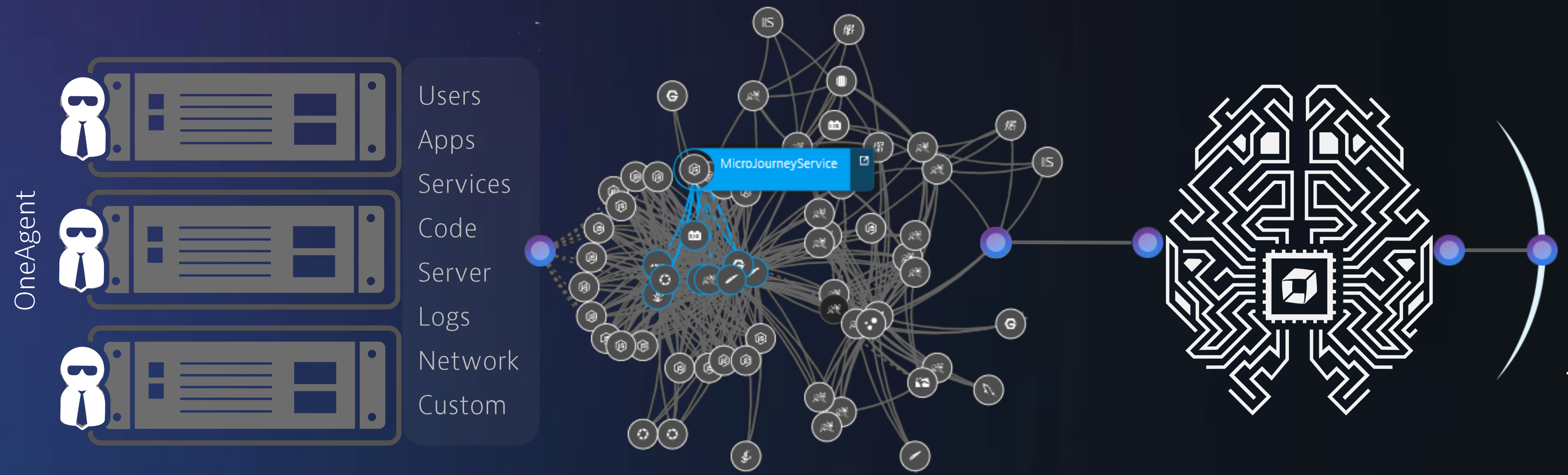
AIOps

Digital experience management

Software Intelligence Platform

Better data makes Dynatrace A.I. and massive automation possible

High fidelity data | Mapped end-to-end | Deterministic AI | Answers + Action



Automated problem detection

Business impact determined

Root cause explained

No alert storms

Trigger self healing

Completely automated

Dynatrace - Software intelligence built for the enterprise cloud

Application performance management

Microservices & containers	Code-level	Database monitoring
Lifecycle analytics	Transaction tracing	Monolith and mainframe

Cloud infrastructure monitoring

Cloud monitoring	Virtualization monitoring	Network monitoring
Container monitoring	Server monitoring	Log monitoring

AIOps

Continuous auto-discovery	Anomaly detection	Prediction
Automatic topology	Root cause analysis	Third party data & metrics

Digital experience management

Real user monitoring	Mobile app monitoring	RUM for SaaS vendors
Session replay	Synthetic monitoring	Digital experience insights

All-in-one software intelligence platform.

Think you know Dynatrace? Think again.

550+ engineers in R&D make it all possible

66% more than nearest competitor

New releases every 2 weeks

26 releases per year

A trusted leader

8 year leader in Gartner MQ

#1 market share leader

Trusted by 72 out of the Fortune 100

OpenShift Primed Partner

Pivotal Technology partner

Docker Certified

AWS Advanced Technology Partner

Azure Partner

Get ready to be amazed in 5 minutes or less

dynatrace.com/trial

dynatrace

dynatrace.com