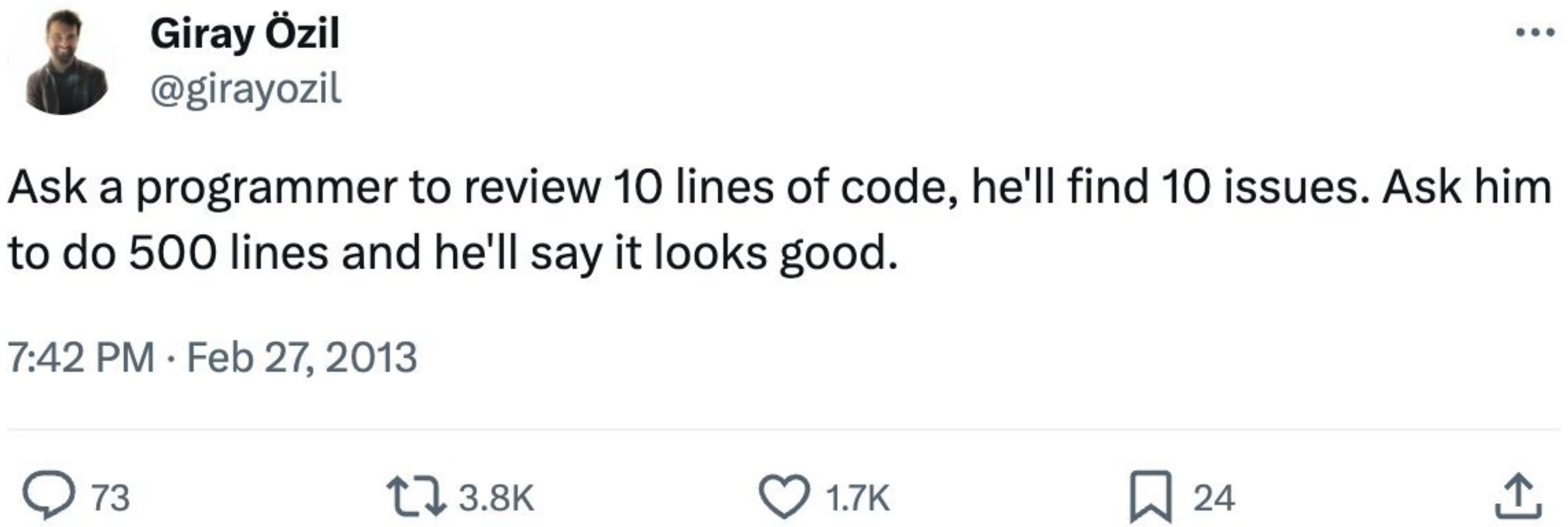
The Power of Continuous Profiling

By Alla Igityan

Observability



Giray Özil
@girayozil

Ask a programmer to review 10 lines of code, he'll find 10 issues. Ask him to do 500 lines and he'll say it looks good.

7:42 PM · Feb 27, 2013

73 3.8K 1.7K 24

heap live size

wall time

thrown exceptions

memory

class load

cpu

file i/o

heap live objects

lock

socket i/o

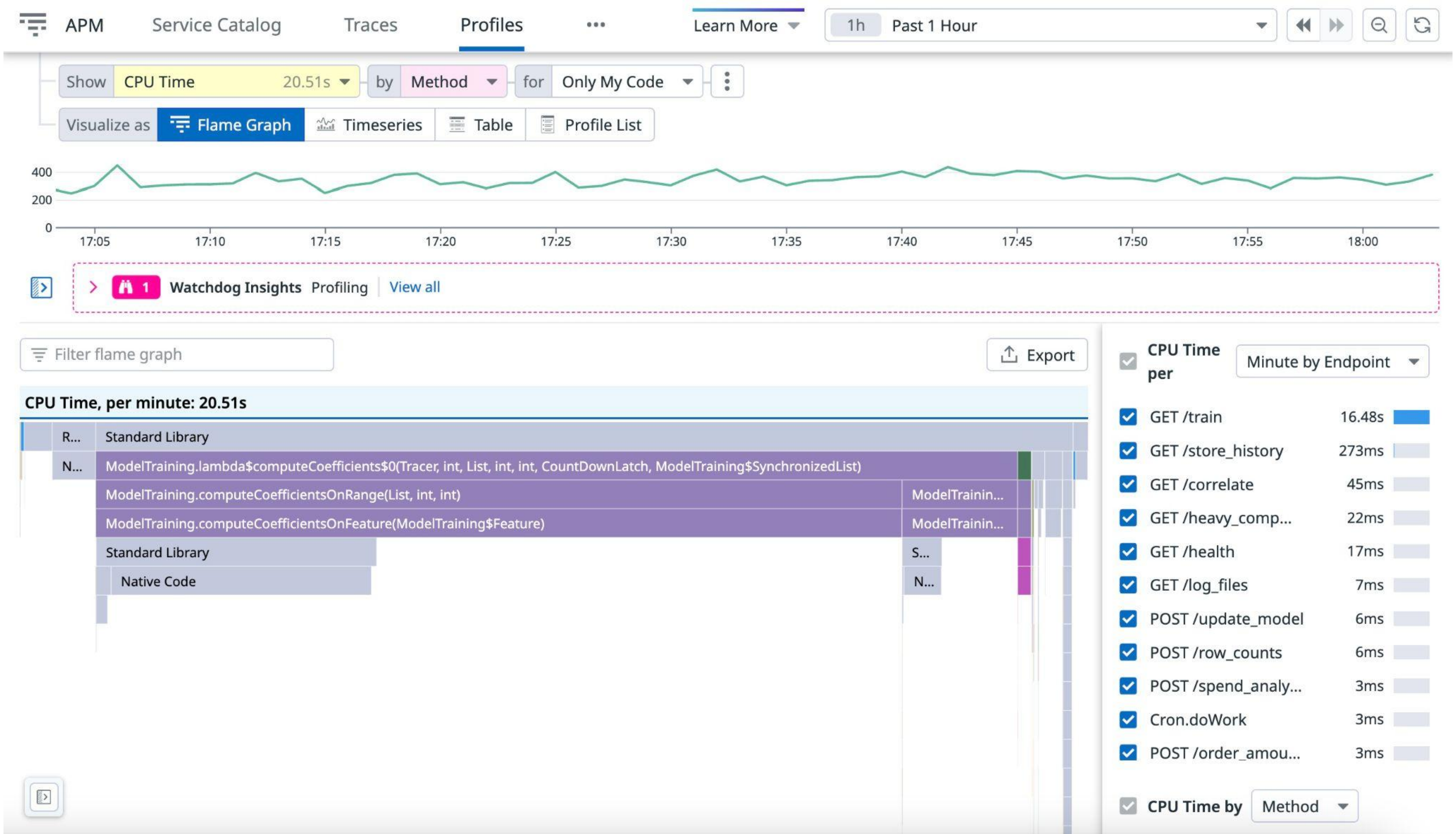
Overhead

Known Knowns

Known Unknowns

Unknown Unknowns

Continuous Profiling



Show CPU Time 20.51s by Method for Only My Code

CPU Time, per minute: 20.51s

R... Standard Library

N... ModelTraining.lambda$computeCoefficients$0(Tracer, int, List, int, int, CountDownLatch, ModelTraining$SynchronizedList)

ModelTraining.computeCoefficientsOnRange(List, int, int) ModelTrainin...

ModelTraining.computeCoefficientsOnFeature(ModelTraining$Feature) ModelTrainin...

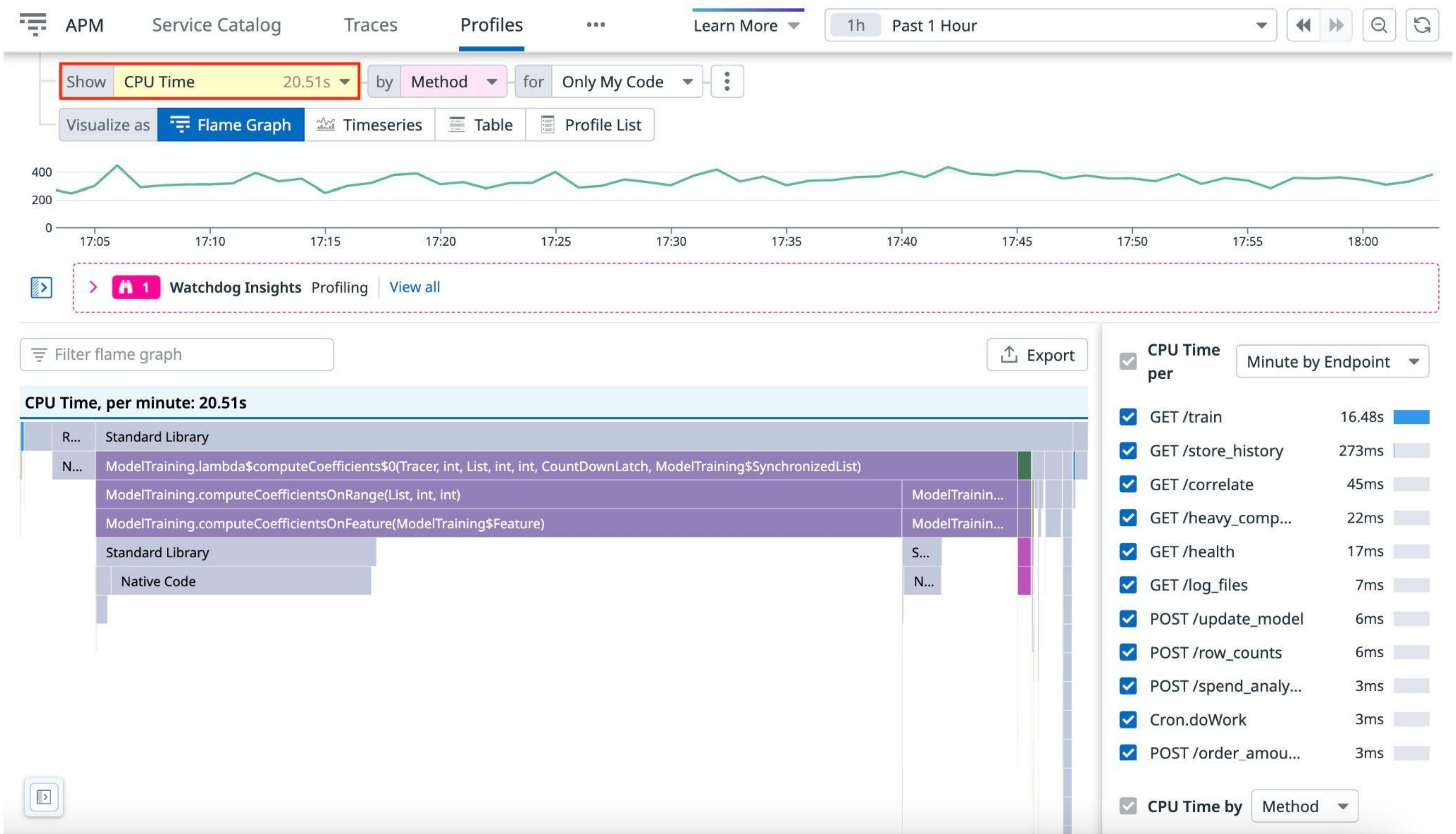
Standard Library S...

Native Code N...

CPU Time per Minute by Endpoint

✓	GET /train	16.48s
✓	GET /store_history	273ms
✓	GET /correlate	45ms
✓	GET /heavy_comp...	22ms
✓	GET /health	17ms
✓	GET /log_files	7ms
✓	POST /update_model	6ms
✓	POST /row_counts	6ms
✓	POST /spend_analy...	3ms
✓	Cron.doWork	3ms
✓	POST /order_amou...	3ms

CPU Time by Method



Show CPU Time 20.51s by Method for Only My Code

Watchdog Insights 1 Profiling | View all

CPU Time, per minute: 20.51s

R... Standard Library

N... ModelTraining.lambda$computeCoefficients$0(Tracer, int, List, int, int, CountDownLatch, ModelTraining$SynchronizedList)

ModelTraining.computeCoefficientsOnRange(List, int, int) | ModelTrainin...

ModelTraining.computeCoefficientsOnFeature(ModelTraining$Feature) | ModelTrainin...

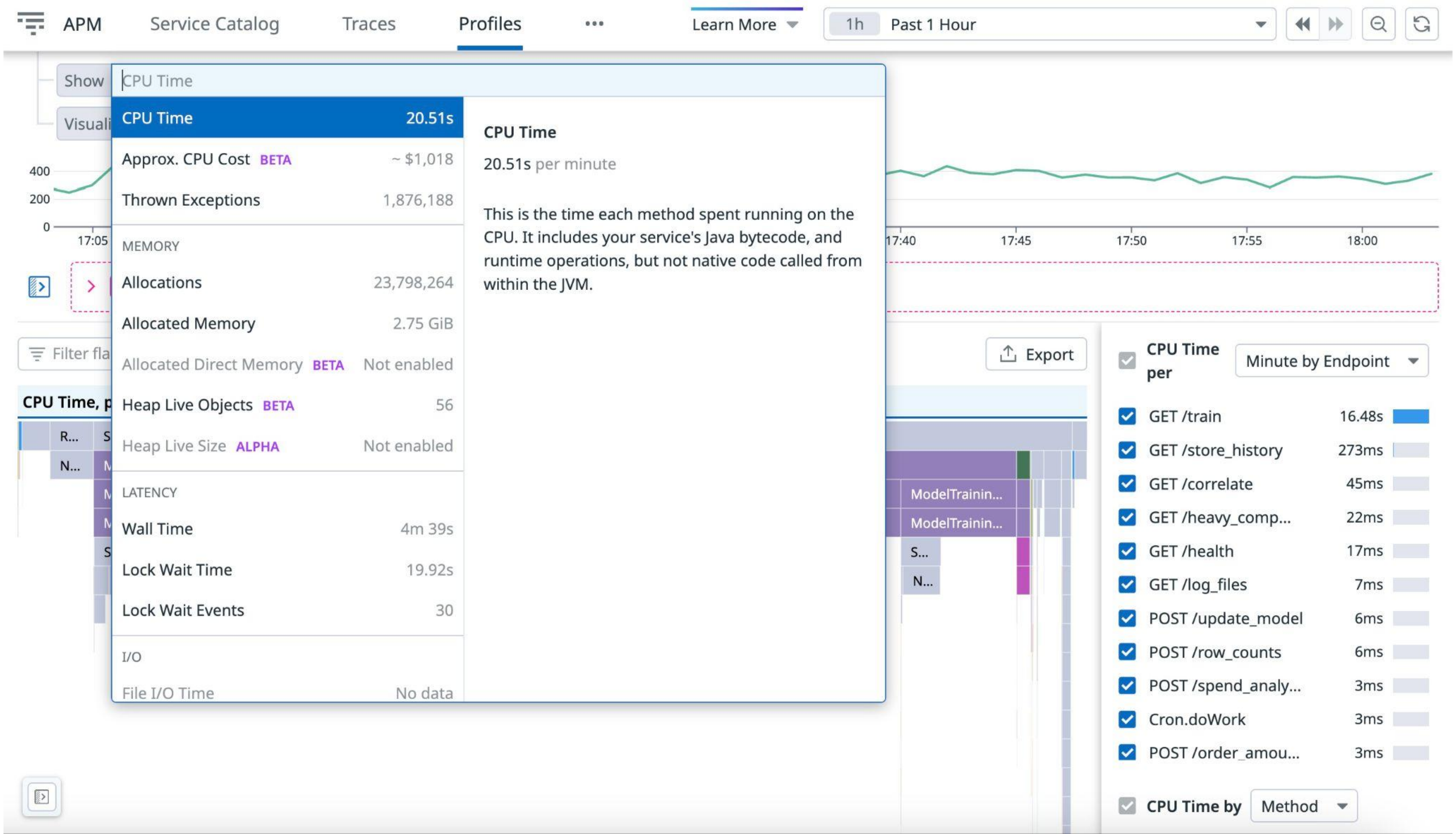
Standard Library | S...

Native Code | N...

CPU Time per: Minute by Endpoint

✓	GET /train	16.48s
✓	GET /store_history	273ms
✓	GET /correlate	45ms
✓	GET /heavy_comp...	22ms
✓	GET /health	17ms
✓	GET /log_files	7ms
✓	POST /update_model	6ms
✓	POST /row_counts	6ms
✓	POST /spend_analy...	3ms
✓	Cron.doWork	3ms
✓	POST /order_amou...	3ms

CPU Time by: Method



CPU Time

CPU Time	20.51s
Approx. CPU Cost BETA	~ $1,018
Thrown Exceptions	1,876,188
MEMORY	
Allocations	23,798,264
Allocated Memory	2.75 GiB
Allocated Direct Memory BETA	Not enabled
Heap Live Objects BETA	56
Heap Live Size ALPHA	Not enabled
LATENCY	
Wall Time	4m 39s
Lock Wait Time	19.92s
Lock Wait Events	30
I/O	
File I/O Time	No data

CPU Time

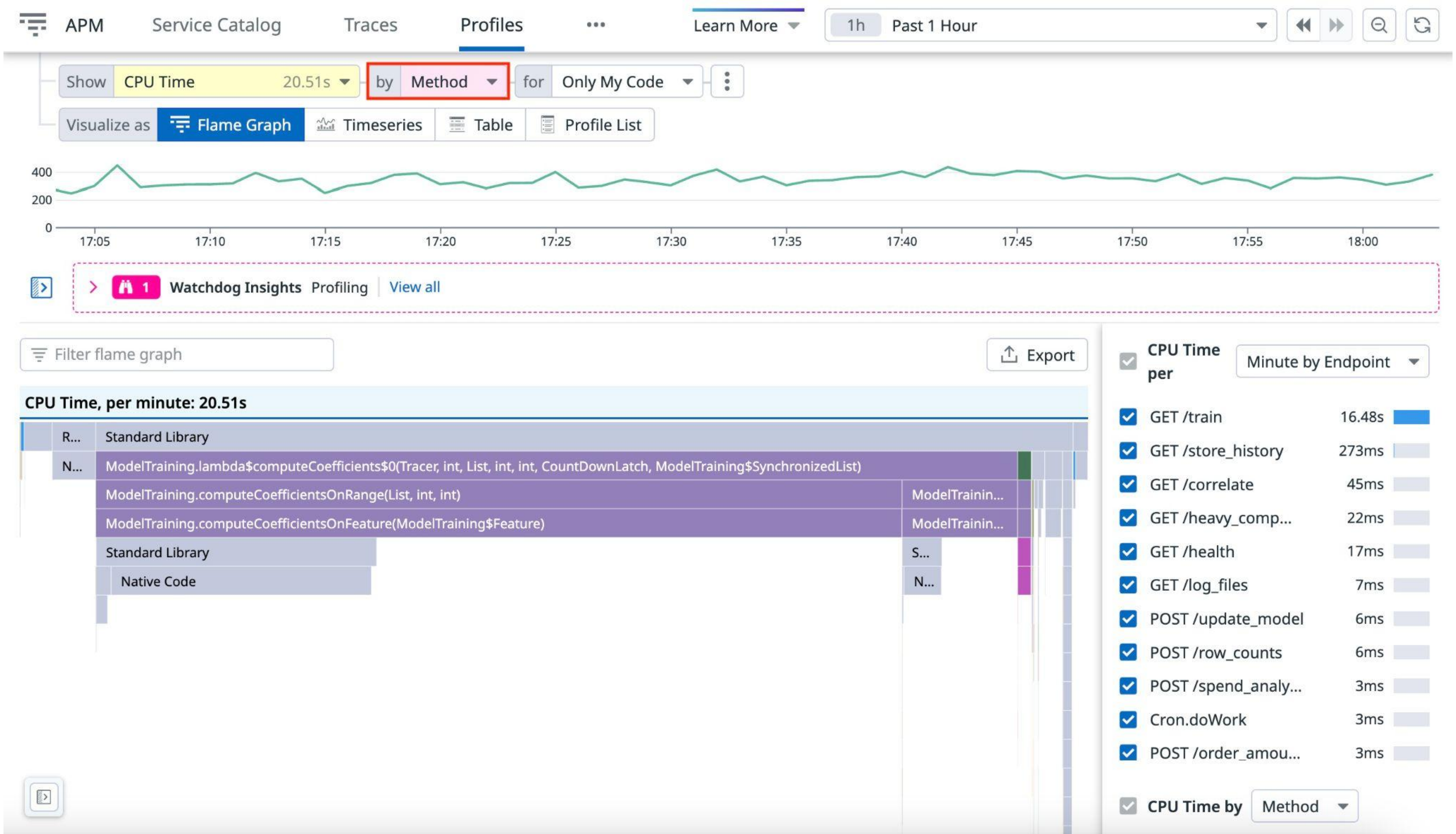
20.51s per minute

This is the time each method spent running on the CPU. It includes your service's Java bytecode, and runtime operations, but not native code called from within the JVM.

CPU Time per: Minute by Endpoint

✓	GET /train	16.48s
✓	GET /store_history	273ms
✓	GET /correlate	45ms
✓	GET /heavy_comp...	22ms
✓	GET /health	17ms
✓	GET /log_files	7ms
✓	POST /update_model	6ms
✓	POST /row_counts	6ms
✓	POST /spend_analy...	3ms
✓	Cron.doWork	3ms
✓	POST /order_amou...	3ms

CPU Time by: Method



APM | Service Catalog | Traces | Profiles | ••• | Learn More | 1h Past 1 Hour

Show CPU Time 20.51s | by Method | for Only My Code

Visualize as: Flame Graph | Timeseries | Table | Profile List

Watchdog Insights 1 Profiling | View all

Filter flame graph | Export

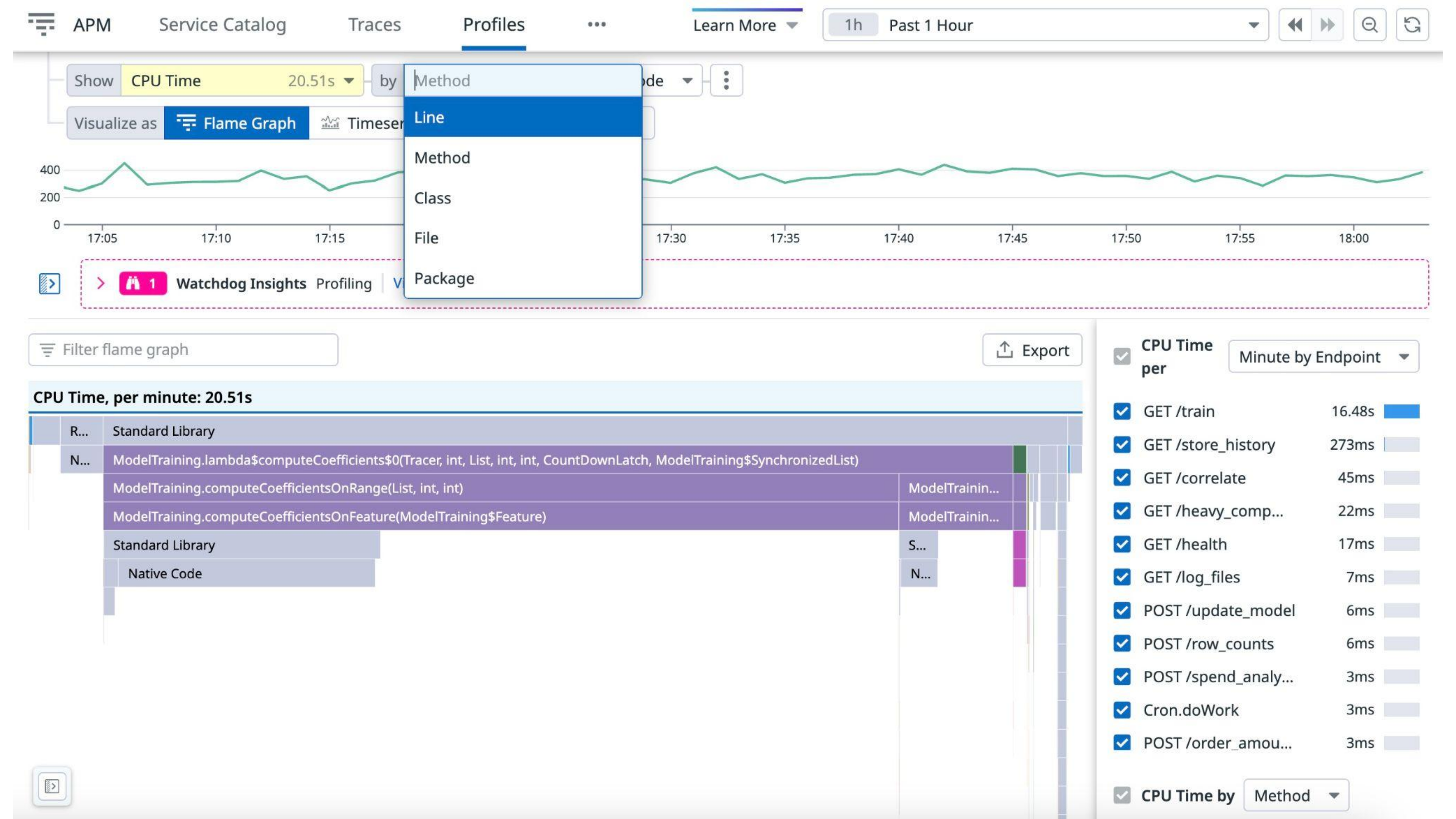
CPU Time, per minute: 20.51s

R... | Standard Library
N... | ModelTraining.lambda$computeCoefficients$0(Tracer, int, List, int, int, CountDownLatch, ModelTraining$SynchronizedList)
ModelTraining.computeCoefficientsOnRange(List, int, int) | ModelTrainin...
ModelTraining.computeCoefficientsOnFeature(ModelTraining$Feature) | ModelTrainin...
Standard Library | S...
Native Code | N...

CPU Time per: Minute by Endpoint

GET /train	16.48s
GET /store_history	273ms
GET /correlate	45ms
GET /heavy_comp...	22ms
GET /health	17ms
GET /log_files	7ms
POST /update_model	6ms
POST /row_counts	6ms
POST /spend_analy...	3ms
Cron.doWork	3ms
POST /order_amou...	3ms

CPU Time by: Method



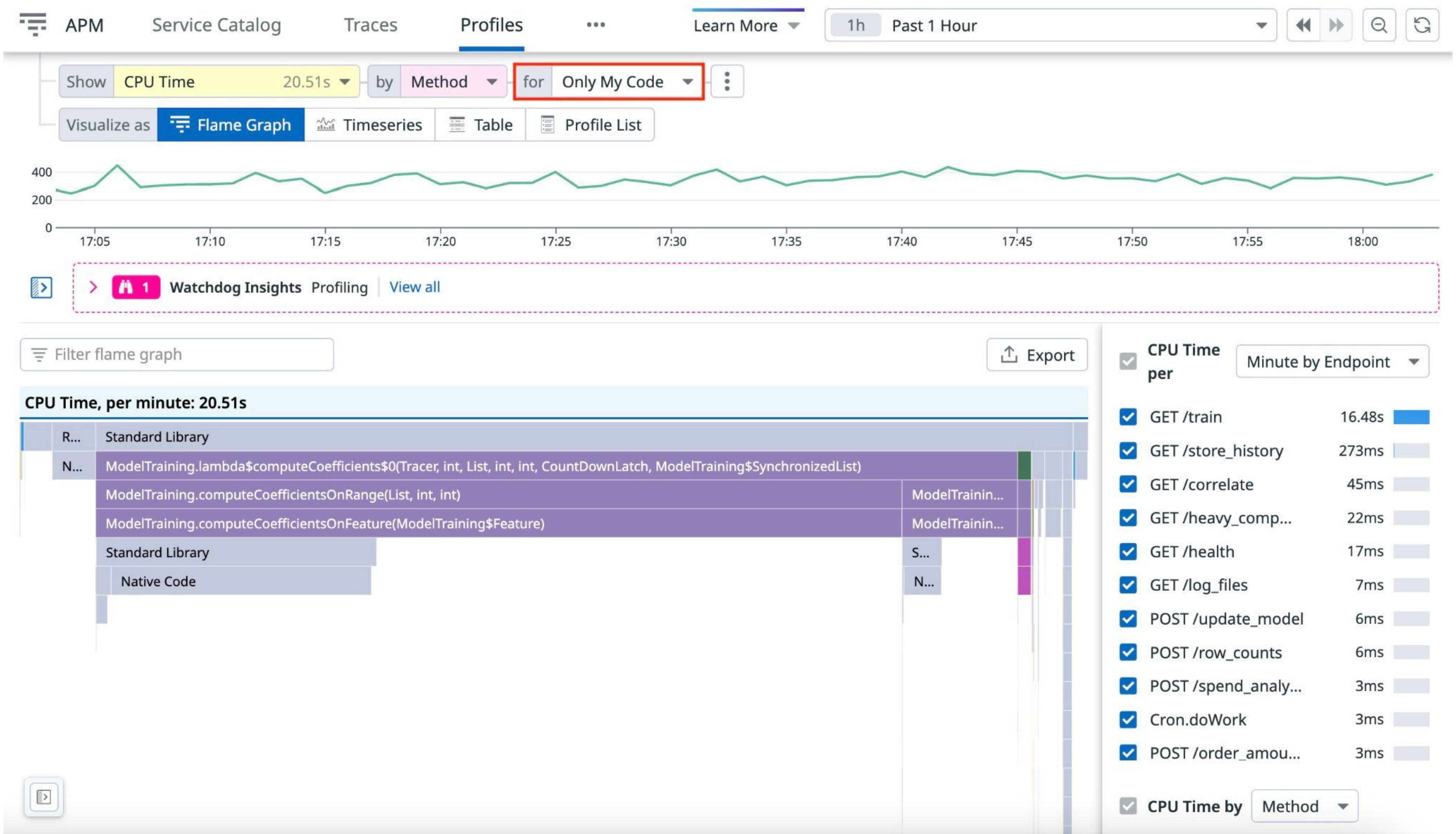
CPU Time, per minute: 20.51s

```
R...  Standard Library
N...  ModelTraining.lambda$computeCoefficients$0(Tracer, int, List, int, int, CountDownLatch, ModelTraining$SynchronizedList)
      ModelTraining.computeCoefficientsOnRange(List, int, int)                ModelTrainin...
      ModelTraining.computeCoefficientsOnFeature(ModelTraining$Feature)       ModelTrainin...
      Standard Library                                                        S...
        Native Code                                                           N...
```

CPU Time per: Minute by Endpoint

✓	GET /train	16.48s
✓	GET /store_history	273ms
✓	GET /correlate	45ms
✓	GET /heavy_comp...	22ms
✓	GET /health	17ms
✓	GET /log_files	7ms
✓	POST /update_model	6ms
✓	POST /row_counts	6ms
✓	POST /spend_analy...	3ms
✓	Cron.doWork	3ms
✓	POST /order_amou...	3ms

CPU Time by: Method



Show CPU Time 20.51s by Method for Only My Code

CPU Time, per minute: 20.51s

R... Standard Library

N... ModelTraining.lambda$computeCoefficients$0(Tracer, int, List, int, int, CountDownLatch, ModelTraining$SynchronizedList)

ModelTraining.computeCoefficientsOnRange(List, int, int) | ModelTrainin...

ModelTraining.computeCoefficientsOnFeature(ModelTraining$Feature) | ModelTrainin...

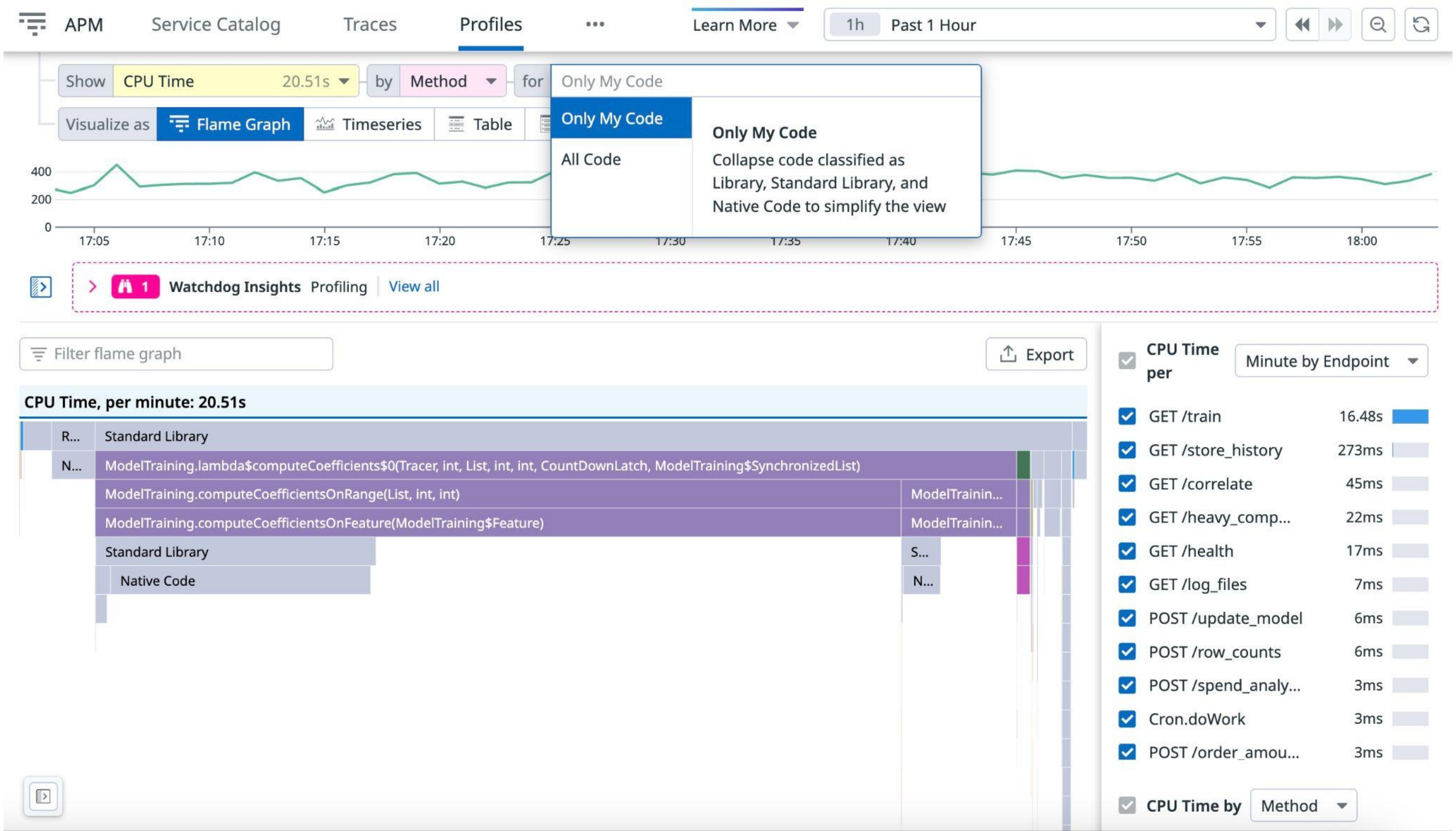
Standard Library | S...

Native Code | N...

CPU Time per Minute by Endpoint

GET /train	16.48s
GET /store_history	273ms
GET /correlate	45ms
GET /heavy_comp...	22ms
GET /health	17ms
GET /log_files	7ms
POST /update_model	6ms
POST /row_counts	6ms
POST /spend_analy...	3ms
Cron.doWork	3ms
POST /order_amou...	3ms

CPU Time by Method



Show CPU Time 20.51s by Method for Only My Code

- Only My Code
- All Code

Only My Code
Collapse code classified as Library, Standard Library, and Native Code to simplify the view

Watchdog Insights 1 Profiling | View all

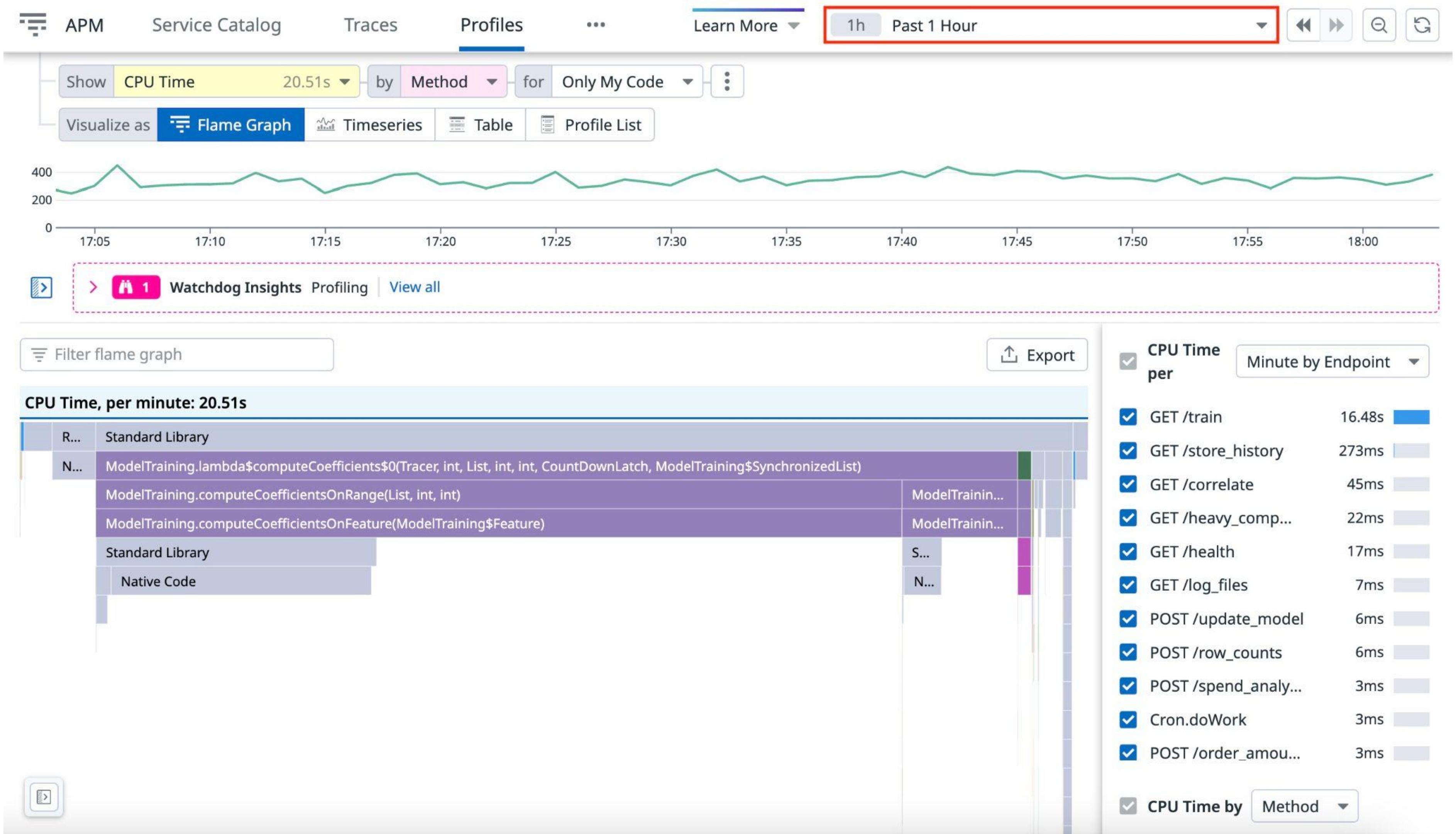
CPU Time, per minute: 20.51s

```
Standard Library
ModelTraining.lambda$computeCoefficients$0(Tracer, int, List, int, int, CountDownLatch, ModelTraining$SynchronizedList)
ModelTraining.computeCoefficientsOnRange(List, int, int)
ModelTraining.computeCoefficientsOnFeature(ModelTraining$Feature)
Standard Library
Native Code
```

CPU Time per Minute by Endpoint

✓	GET /train	16.48s
✓	GET /store_history	273ms
✓	GET /correlate	45ms
✓	GET /heavy_comp...	22ms
✓	GET /health	17ms
✓	GET /log_files	7ms
✓	POST /update_model	6ms
✓	POST /row_counts	6ms
✓	POST /spend_analy...	3ms
✓	Cron.doWork	3ms
✓	POST /order_amou...	3ms

CPU Time by Method


APM | Service Catalog | Traces | Profiles | ••• | Learn More | 1h Past 1 Hour

Show CPU Time 20.51s by Method for Only My Code

Visualize as Flame Graph | Timeseries | Table | Profile List

1 Watchdog Insights Profiling | View all

Filter flame graph | Export

CPU Time, per minute: 20.51s

R... Standard Library

N... ModelTraining.lambda$computeCoefficients$0(Tracer, int, List, int, int, CountDownLatch, ModelTraining$SynchronizedList)

ModelTraining.computeCoefficientsOnRange(List, int, int) | ModelTrainin...

ModelTraining.computeCoefficientsOnFeature(ModelTraining$Feature) | ModelTrainin...

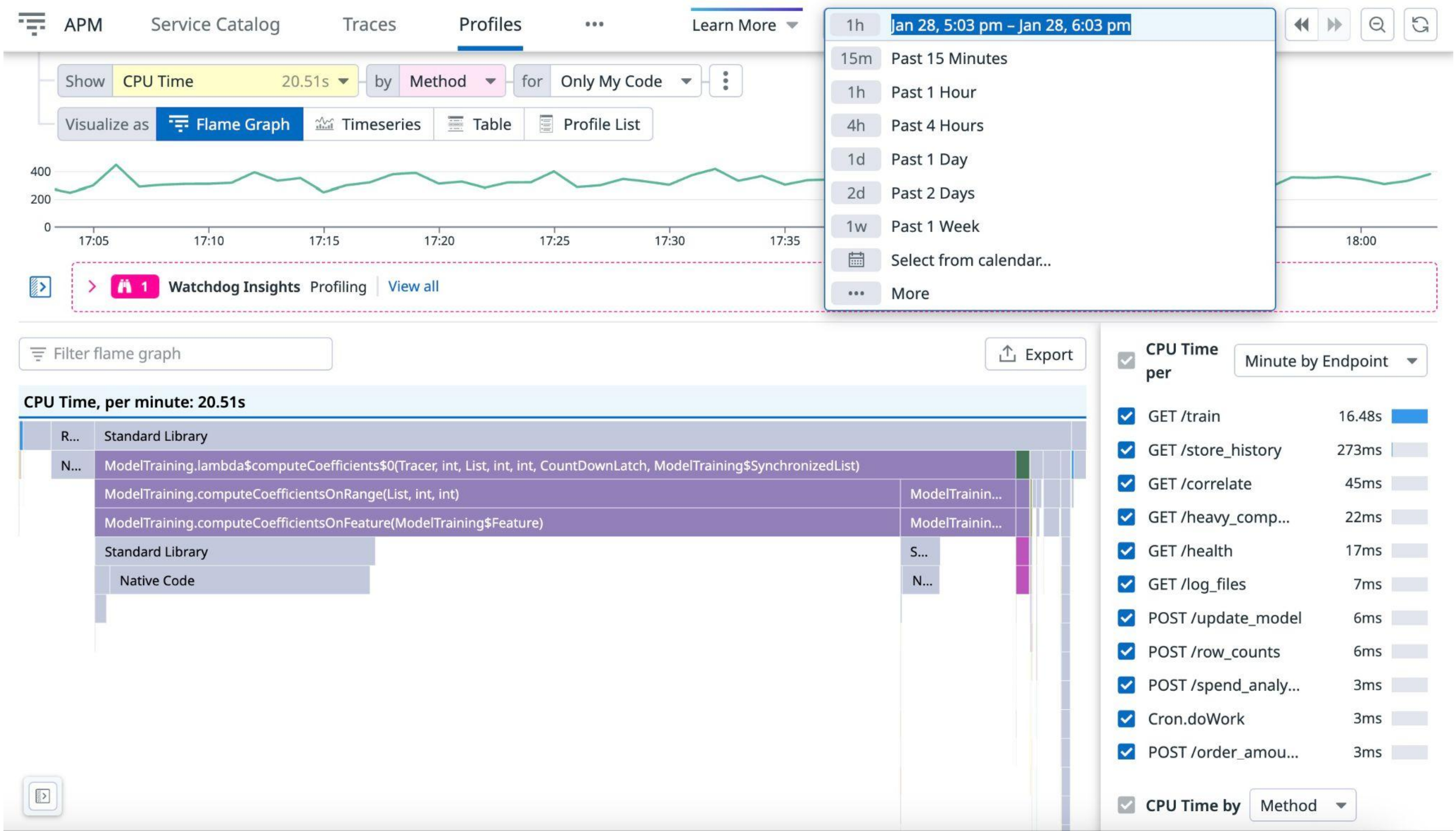
Standard Library | S...

Native Code | N...

CPU Time per Minute by Endpoint

✓	GET /train	16.48s
✓	GET /store_history	273ms
✓	GET /correlate	45ms
✓	GET /heavy_comp...	22ms
✓	GET /health	17ms
✓	GET /log_files	7ms
✓	POST /update_model	6ms
✓	POST /row_counts	6ms
✓	POST /spend_analy...	3ms
✓	Cron.doWork	3ms
✓	POST /order_amou...	3ms

CPU Time by Method



Show CPU Time 20.51s by Method for Only My Code

1h Jan 28, 5:03 pm – Jan 28, 6:03 pm

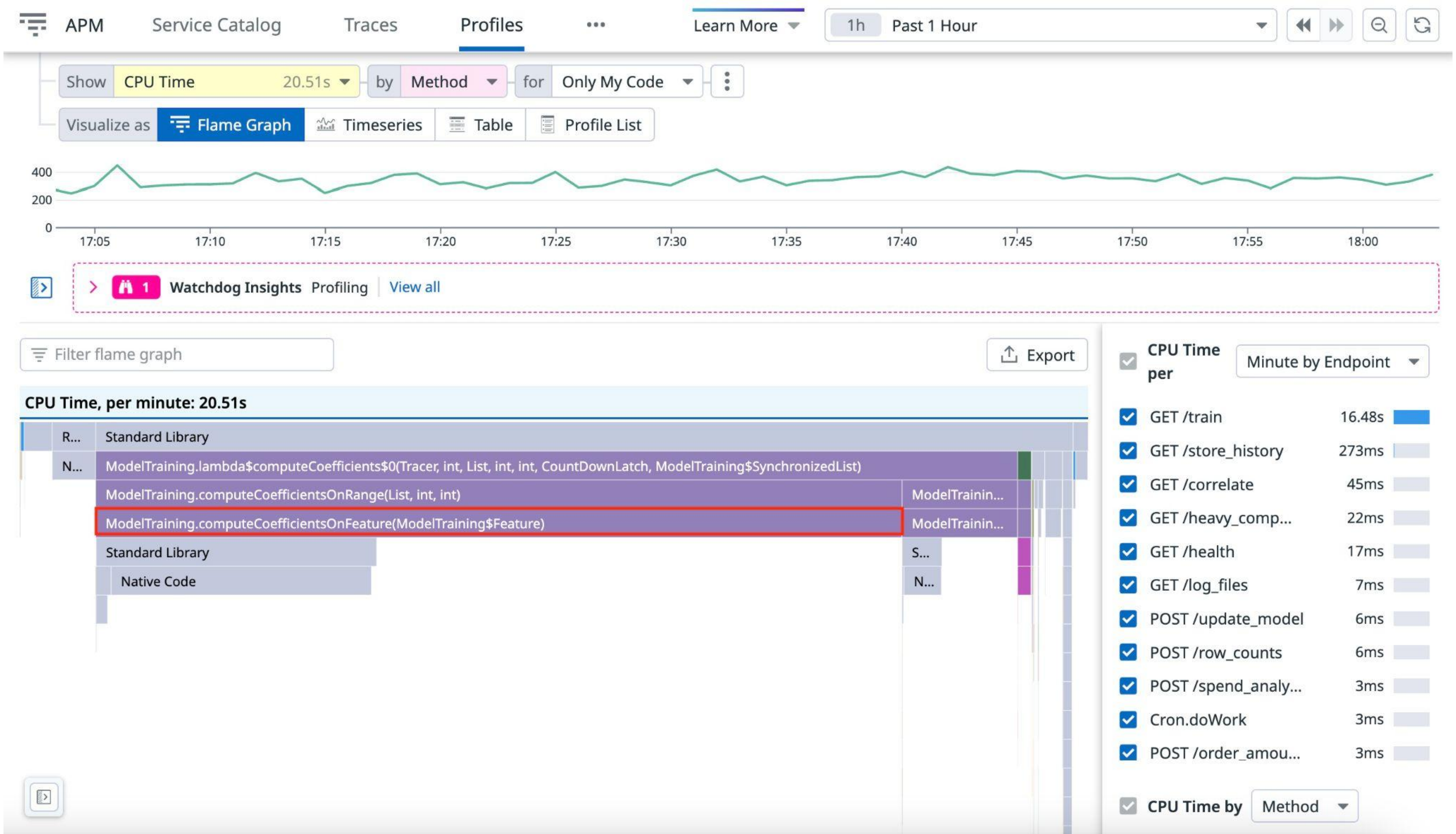
15m	Past 15 Minutes
1h	Past 1 Hour
4h	Past 4 Hours
1d	Past 1 Day
2d	Past 2 Days
1w	Past 1 Week
	Select from calendar...
•••	More

CPU Time, per minute: 20.51s

```
Standard Library
ModelTraining.lambda$computeCoefficients$0(Tracer, int, List, int, int, CountDownLatch, ModelTraining$SynchronizedList)
ModelTraining.computeCoefficientsOnRange(List, int, int)
ModelTraining.computeCoefficientsOnFeature(ModelTraining$Feature)
Standard Library
Native Code
```

CPU Time per: Minute by Endpoint

✓	GET /train	16.48s
✓	GET /store_history	273ms
✓	GET /correlate	45ms
✓	GET /heavy_comp...	22ms
✓	GET /health	17ms
✓	GET /log_files	7ms
✓	POST /update_model	6ms
✓	POST /row_counts	6ms
✓	POST /spend_analy...	3ms
✓	Cron.doWork	3ms
✓	POST /order_amou...	3ms



Show CPU Time 20.51s by Method for Only My Code

CPU Time, per minute: 20.51s

R... Standard Library

N... ModelTraining.lambda$computeCoefficients$0(Tracer, int, List, int, int, CountDownLatch, ModelTraining$SynchronizedList)

ModelTraining.computeCoefficientsOnRange(List, int, int) ModelTrainin...

ModelTraining.computeCoefficientsOnFeature(ModelTraining$Feature) ModelTrainin...

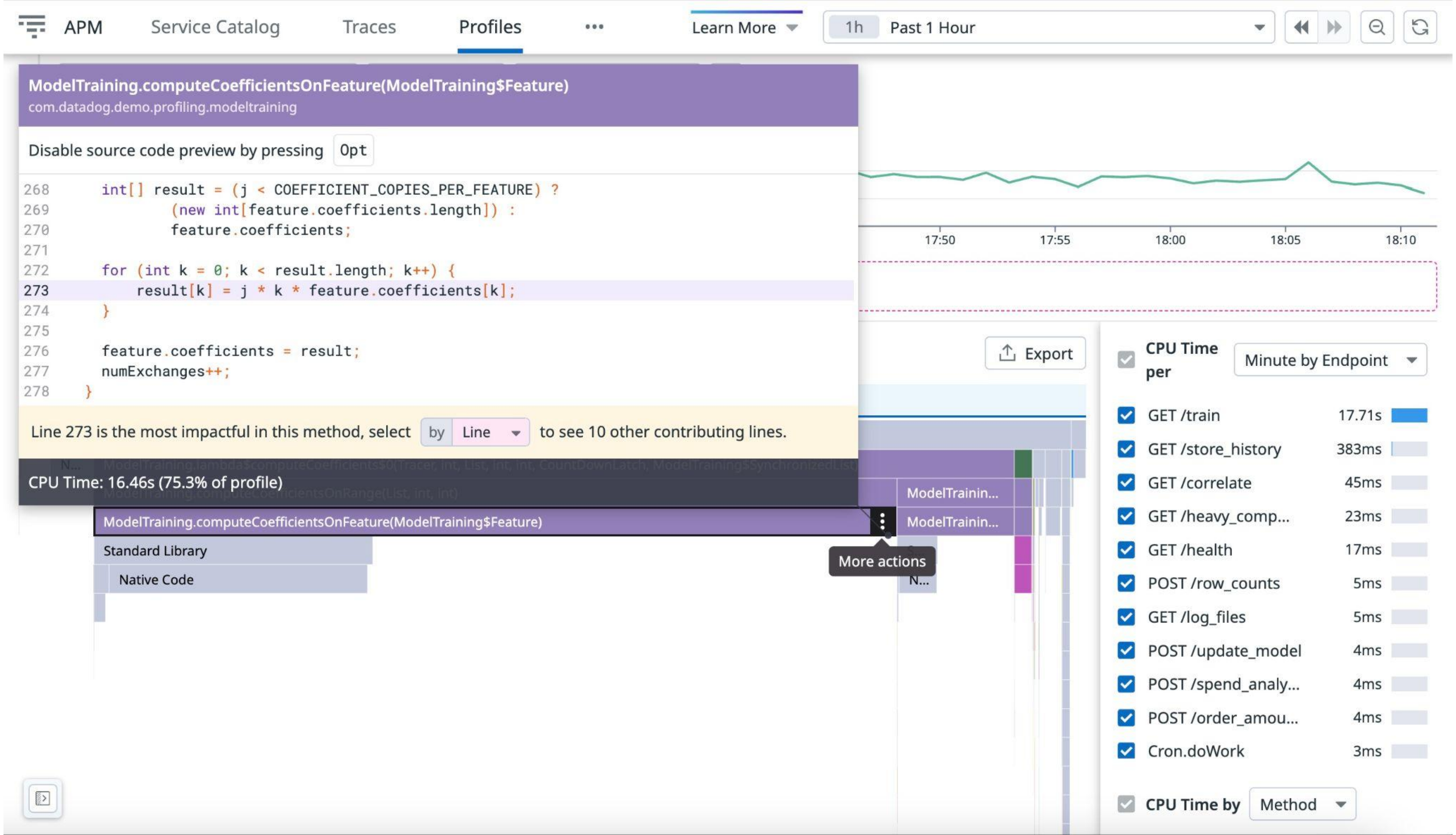
Standard Library S...

Native Code N...

CPU Time per Minute by Endpoint

GET /train	16.48s	
GET /store_history	273ms	
GET /correlate	45ms	
GET /heavy_comp...	22ms	
GET /health	17ms	
GET /log_files	7ms	
POST /update_model	6ms	
POST /row_counts	6ms	
POST /spend_analy...	3ms	
Cron.doWork	3ms	
POST /order_amou...	3ms	

CPU Time by Method



ModelTraining.computeCoefficientsOnFeature(ModelTraining$Feature)
com.datadog.demo.profiling.modeltraining

Disable source code preview by pressing Opt

```
268       int[] result = (j < COEFFICIENT_COPIES_PER_FEATURE) ?
269               (new int[feature.coefficients.length]) :
270               feature.coefficients;
271
272       for (int k = 0; k < result.length; k++) {
273           result[k] = j * k * feature.coefficients[k];
274       }
275
276       feature.coefficients = result;
277       numExchanges++;
278   }
```

Line 273 is the most impactful in this method, select by Line to see 10 other contributing lines.

CPU Time: 16.46s (75.3% of profile)

ModelTraining.lambda$computeCoefficients$0(Tracer, int, List, int, int, CountDownLatch, ModelTraining$SynchronizedList)
ModelTraining.computeCoefficientsOnRange(List, int, int)
ModelTraining.computeCoefficientsOnFeature(ModelTraining$Feature)
ModelTrainin...
ModelTrainin...
Standard Library
Native Code
N...

CPU Time per: Minute by Endpoint

GET /train	17.71s
GET /store_history	383ms
GET /correlate	45ms
GET /heavy_comp...	23ms
GET /health	17ms
POST /row_counts	5ms
GET /log_files	5ms
POST /update_model	4ms
POST /spend_analy...	4ms
POST /order_amou...	4ms
Cron.doWork	3ms

CPU Time by: Method

Why Continuous Profiling

- Reduce MTTR
- Reduce Cloud Costs
- Improve User Experience

Helpful links

On Flame graphs

How we optimized our Akka application using Datadog’s Continuous Profiler

Presentation: Future-proofing JVM profiling

7 February 10:10 - 11:00 Room C3

Thank you!