TWITTER'S QUEST FOR A WHOLLY GRAAL RUNTIME

@CHRISTHALINGER

#TWITTERVMTEAM

@CHRISTHALINGER

Goal?

Save $$$ 🤑

TWITTER

- Huge distributed system
- Comprised of many services
  - tweet service
  - user service
  - timeline service
  - social graph service
  - ...

SERVICES

- Many JVMs per service
- 1,000s of machines running 1,000s of JVMs
- Multiple datacenters

OPEN-SOURCE

https://engineering.twitter.com/opensource

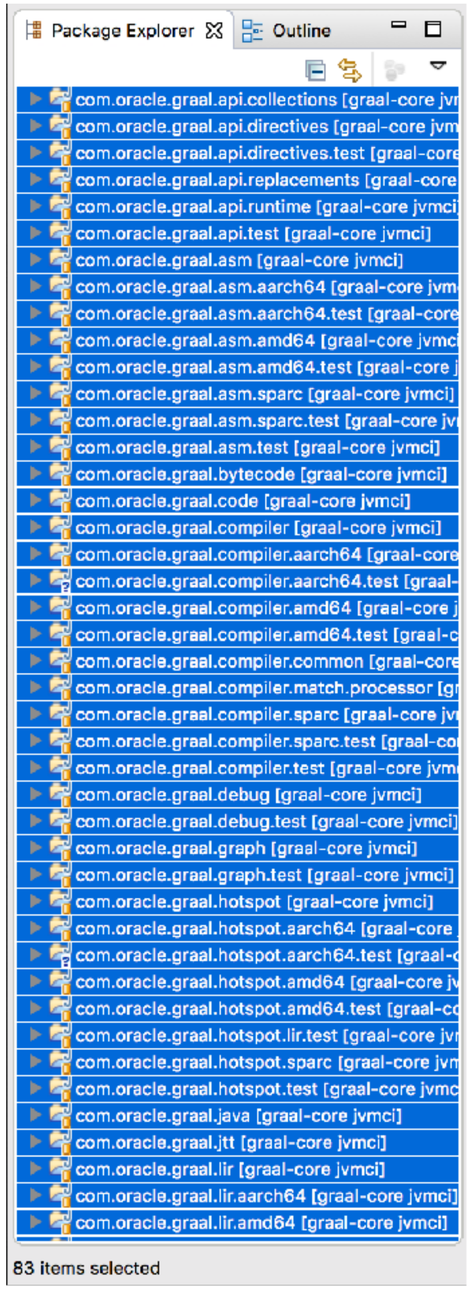
- Twitter is using many open-source projects
- ...and also gives back to the community:
  - https://twitter.github.io
- Graal is a good fit
  - We are contributing, too

TWITTER'S JDK

- Based on OpenJDK 8u
- Includes:
  - JEP 243 backport (JVMCI)
  - Graal
  - Contrail (JFR replacement)
  - CMS improvements

WHY GRAAL?

- C2 is old and very complex
  - Learning curve way too steep
  - The last few years no new major optimizations
  - Reached its end-of-life already years ago
- Graal is easier to understand
  - Modular design
- Better inlining and escape analysis



Bugs

BUG 1: #128

Graal should support same OSR compilations as C1 and C2

- https://github.com/graalvm/graal-core/issues/128
  - (OSR means On-Stack Replacement)
- `PrintCompilation` shows something like this:
  - `COMPILE SKIPPED: OSR with locks not supported`
- The exception looks like this...

BUG 1: #128

Graal should support same OSR compilations as C1 and C2

```
jdk.vm.ci.code.BailoutException: OSR with locks not supported
        at com.oracle.graal.hotspot.phases.OnStackReplacementPhase.getEntryMarker(OnStackReplacementPhase.java:139)
        at com.oracle.graal.hotspot.phases.OnStackReplacementPhase.run(OnStackReplacementPhase.java:69)
        at com.oracle.graal.phases.Phase.run(Phase.java:51)
        at com.oracle.graal.phases.BasePhase.apply(BasePhase.java:166)
        at com.oracle.graal.phases.BasePhase.apply(BasePhase.java:148)
        at com.oracle.graal.phases.PhaseSuite.run(PhaseSuite.java:154)
        at com.oracle.graal.phases.BasePhase.apply(BasePhase.java:166)
        at com.oracle.graal.phases.BasePhase.apply(BasePhase.java:148)
        at com.oracle.graal.compiler.GraalCompiler.emitFrontEnd(GraalCompiler.java:193)
        at com.oracle.graal.compiler.GraalCompiler.compile(GraalCompiler.java:175)
        at com.oracle.graal.compiler.GraalCompiler.compileGraph(GraalCompiler.java:162)
        at com.oracle.graal.hotspot.HotSpotGraalCompiler.compile(HotSpotGraalCompiler.java:179)
        at com.oracle.graal.hotspot.CompilationTask.runCompilation(CompilationTask.java:212)
        at com.oracle.graal.hotspot.HotSpotGraalCompiler.compileMethod(HotSpotGraalCompiler.java:128)
        at jdk.vm.ci.hotspot.HotSpotJVMCIRuntime.compileMethod(HotSpotJVMCIRuntime.java:409)
```

BUG 1: #128

Graal should support same OSR compilations as C1 and C2

- Tom and I discussed this for a bit…
- …and we decided that it's not a big problem
- Tiered compilation takes care of it
  - After a while the failing method will be flagged and compiled with C1
- Still would be good to have this fixed

#128 Closed

P.S. technically it's still open but was fixed by:

[GR-1501] Support OSR with locks on the stack.

master graal-vm-0.22 ··· JDK-8177046

Browse files

davleopo committed on Feb 6 2 parents 82e9913 + f725f5a commit 34dc32ce647bb0732428854049b0c12cdbeab35a

Showing **14 changed files** with **1,038 additions** and **53 deletions**.

Unified Split

BUG 1: #128

christhalinger commented on Mar 24 — Member

The failures I've seen before are fixed but now I'm seeing these:

```
$ cat graal.log | grep -B1 BailoutException
Bailout in net.reduls.gomoku.dic.Morpheme.<clinit>()
org.graalvm.compiler.common.PermanentBailoutException: No OnStackReplacementNode generated
--
Bailout in net.reduls.gomoku.dic.Matrix.<clinit>()
org.graalvm.compiler.common.PermanentBailoutException: No OnStackReplacementNode generated
--
Bailout in java.util.concurrent.atomic.AtomicBoolean.getAndSet(boolean)
org.graalvm.compiler.common.PermanentBailoutException: OSR compilation without OSR entry l
```

tkrodriguez commented on Mar 27 — Member

Any chance you can provide graphs for the last one?

For the first two it looks like the parse didn't reach the OSR point which suggests we're trimming control flow for some reason. I grabbed a copy of gomoku and ran it locally and it appears to be a problem with deoptimizing on static field access in the clinit. There's a little piece of logic in C2's parser which allows access to static fields within the clinit for classes which aren't fully initialized yet.
http://hg.openjdk.java.net/jdk8/jdk8/hotspot/file/87ee5ee27509/src/share/vm/opto/parse3.cpp#l42
I think we need a copy of that logic in the BytecodeParser. I'll create an issue for that.

BUG 1.1: #263

BytecodeParser should allow access to static for uninitialized types in clinit

- https://github.com/graalvm/graal-core/issues/263
- The “little bit of logic in C2’s parser” is this method:

```
bool Parse::static_field_ok_in_clinit(ciField *field, ciMethod *method) {
  // Could be the field_holder's <clinit> method, or <clinit> for a subklass.
  // Better to check now than to Deoptimize as soon as we execute
```

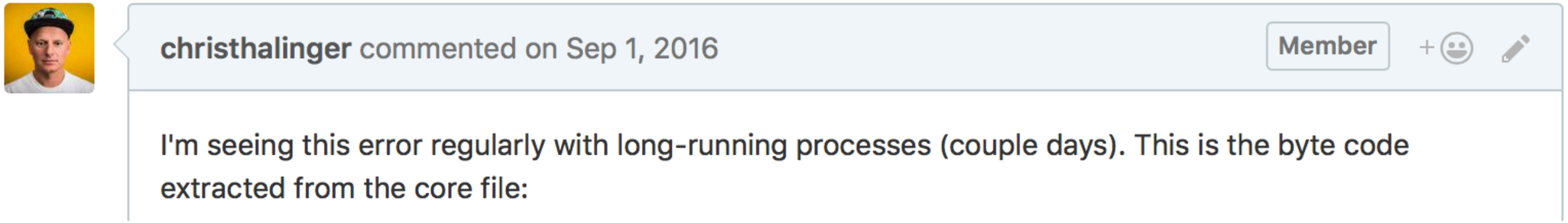
#263 Closed

Showing 1 changed file with **31 additions** and **6 deletions.**

BUG 2: #204

fatal error: no original bytecode found in scala.collection.TraversableLike$class.builder$1(Lscala/collection/TraversableLike;Lscala/collection/generic/CanBuildFrom;)Lscala/collection/mutable/Bu
at bci 24

- https://github.com/graalvm/graal-core/issues/204



BUG 2: #204

fatal error: no original bytecode found in scala.collection.TraversableLike$class.builder$1(Lscala/collection/TraversableLike;Lscala/collection/generic/CanBuildFrom;)Lscala/collection/mutable/Bu
at bci 24

christhalinger commented on Sep 9, 2016 Member

Turns out that the https://github.com/mariusae/heapster agent is causing the problem. I could reproduce it locally with:

BUG 2: #204

fatal error: no original bytecode found in scala.collection.TraversableLike$class.builder$1(Lscala/collection/TraversableLike;Lscala/collection/generic/CanBuildFrom;)Lscala/collection/mutable/Bu
at bci 24

- https://github.com/mariusae/heapster

Heapster

Heapster provides an agent library to do heap profiling for JVM processes with output compatible with Google perftools.
The goal of Heapster is to be able to do meaningful (sampled) heap profiling in a production setting.

dougxc commented on Sep 12, 2016 • edited

Member

As I suspected, the problem is that we don't isolate snippet bytecode from instrumentation agents. In this case, it's the bytecode of `Double.valueOf` inlined into `com.oracle.graal.replacements.BoxingSnippets.doubleValueOf(double)`. Time to think of a solution.

tkrodriguez commented on Sep 12, 2016

Member

This isn't even code that we own so we don't have a good way around this. I think the assert in ReplacementsImpl.notifyNotInlined should be raised to a hard error since it was masking this issue. We might also need more checks in snippet handling in general so that this kind of misassociation of bci and method doesn't happen in other places.

dougxc commented on Sep 12, 2016

Member

I think it's time to bite the bullet and parse the bytecodes for snippets (and their callees) directly. I'm looking into the best way to do this.

#204 Closed

Showing **65 changed files** with **2,767 additions** and **446 deletions.**

BUG 3: #206

Netty 4.1.5 tests fail

- https://github.com/graalvm/graal-core/issues/206
- For the longest time we were seeing connection exceptions in the logs
- We didn't know where they were coming from
- Until I, independently, decided to run the Netty 4 test suite…

BUG 3: #206

Netty 4.1.5 tests fail

christhalinger commented on Sep 9, 2016 Member

Today we narrowed down where these odd connection exceptions we were seeing are coming from:

```
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary:
[INFO]
[INFO] Netty ..................................... SUCCESS [  5.584 s]
[INFO] Netty/Common .............................. SUCCESS [01:15 min]
[INFO] Netty/Buffer .............................. FAILURE [ 48.142 s]
```

tkrodriguez commented on Sep 12, 2016 Member

[9/12/16, 11:07:31 AM] Tom Rodriguez: we seem to be writing the values with the wrong endianness.
[9/12/16, 11:07:39 AM] Tom Rodriguez: java.lang.AssertionError: expected:<54694> but was:<42709>
[9/12/16, 11:07:46 AM] Tom Rodriguez: $ tohex 42709
a6d5
$ tohex 54694
d5a6

tkrodriguez commented on Sep 12, 2016 Member

The problem is that the reverseBytes plugins for short and char do incorrect canonicalization that may eliminate that reverse completely.

#206 Closed

Showing 5 changed files with **68 additions** and **18 deletions.**

Contribution

PR #251

[AMD64] implement String.compareTo substitutions

- https://github.com/graalvm/graal/pull/251
- JDK 9 has Compact Strings (JEP 254)
  - A lot of the important conversion methods have hand-optimized intrinsics (using SSE, AVX and AVX 512)
  - We need all these in Graal as well

PR #251

Graal:

Benchmark	(size)	Mode	Cnt	Score		Error	Units
CompareToBench.cmp1_cmp1	1	avgt	25	5.480	±	0.147	ns/op
CompareToBench.cmp1_cmp1	64	avgt	25	61.178	±	1.768	ns/op
CompareToBench.cmp1_cmp1	4096	avgt	25	2621.249	±	63.138	ns/op
CompareToBench.cmp1_cmp2	1	avgt	25	5.914	±	0.186	ns/op
CompareToBench.cmp1_cmp2	64	avgt	25	93.736	±	1.029	ns/op
CompareToBench.cmp1_cmp2	4096	avgt	25	4975.549	±	45.001	ns/op
CompareToBench.cmp2_cmp1	1	avgt	25	5.841	±	0.049	ns/op
CompareToBench.cmp2_cmp1	64	avgt	25	94.098	±	0.921	ns/op
CompareToBench.cmp2_cmp1	4096	avgt	25	5052.871	±	80.579	ns/op
CompareToBench.cmp2_cmp2	1	avgt	25	5.924	±	0.062	ns/op
CompareToBench.cmp2_cmp2	64	avgt	25	119.063	±	1.316	ns/op
CompareToBench.cmp2_cmp2	4096	avgt	25	6674.597	±	63.872	ns/op

Graal with intrinsic:

Benchmark	(size)	Mode	Cnt	Score		Error	Units
CompareToBench.cmp1_cmp1	1	avgt	25	5.898	±	0.059	ns/op
CompareToBench.cmp1_cmp1	64	avgt	25	14.540	±	0.160	ns/op
CompareToBench.cmp1_cmp1	4096	avgt	25	126.307	±	6.868	ns/op
CompareToBench.cmp1_cmp2	1	avgt	25	5.736	±	0.082	ns/op
CompareToBench.cmp1_cmp2	64	avgt	25	15.717	±	0.219	ns/op
CompareToBench.cmp1_cmp2	4096	avgt	25	210.045	±	5.888	ns/op
CompareToBench.cmp2_cmp1	1	avgt	25	6.591	±	0.863	ns/op
CompareToBench.cmp2_cmp1	64	avgt	25	16.949	±	1.856	ns/op
CompareToBench.cmp2_cmp1	4096	avgt	25	211.254	±	8.404	ns/op
CompareToBench.cmp2_cmp2	1	avgt	25	5.928	±	0.091	ns/op
CompareToBench.cmp2_cmp2	64	avgt	25	15.976	±	0.188	ns/op
CompareToBench.cmp2_cmp2	4096	avgt	25	206.201	±	7.247	ns/op

Numbers

TWEET SERVICE

- Finagle-Thrift service
  - https://twitter.github.io/finagle/

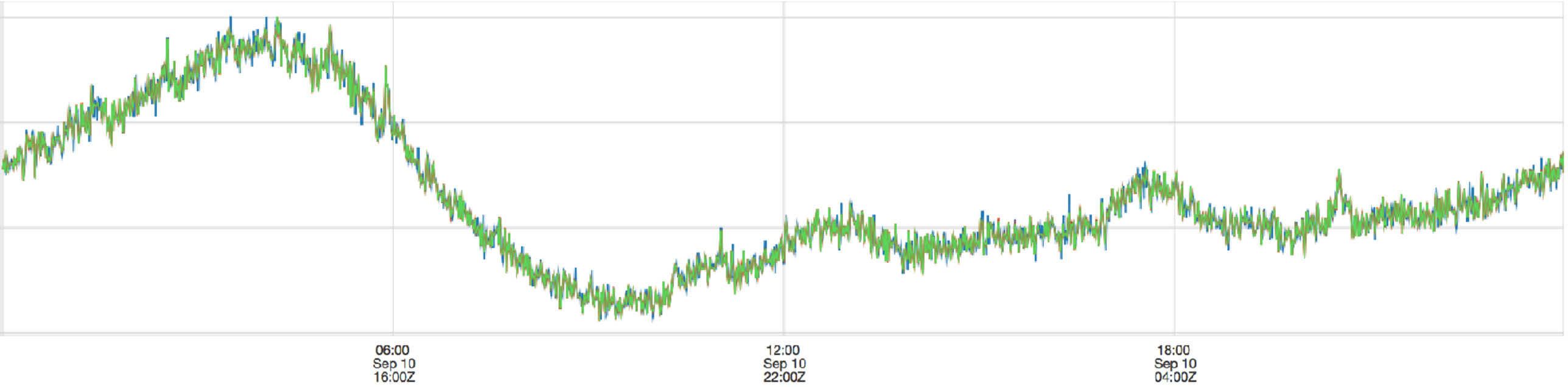
Finagle is an extensible RPC system for the JVM, used to construct high-concurrency servers. Finagle implements uniform client and server APIs for several protocols, and is designed for high performance and concurrency. Most of Finagle's code is protocol agnostic, simplifying the implementation of new protocols.

TEST SETUP

- 1 dedicated machine per instance/configuration
  - All instances receive the exact same requests
- Run with `jvmci-0.30` and `graal-vm-0.22`
  - Default tiered C1/Graal setup
  - Graal bootstrapping is not an issue for most of our services
  - If bootstrapping is an issue, JDK 9 AOT will solve this

REQUESTS/SEC

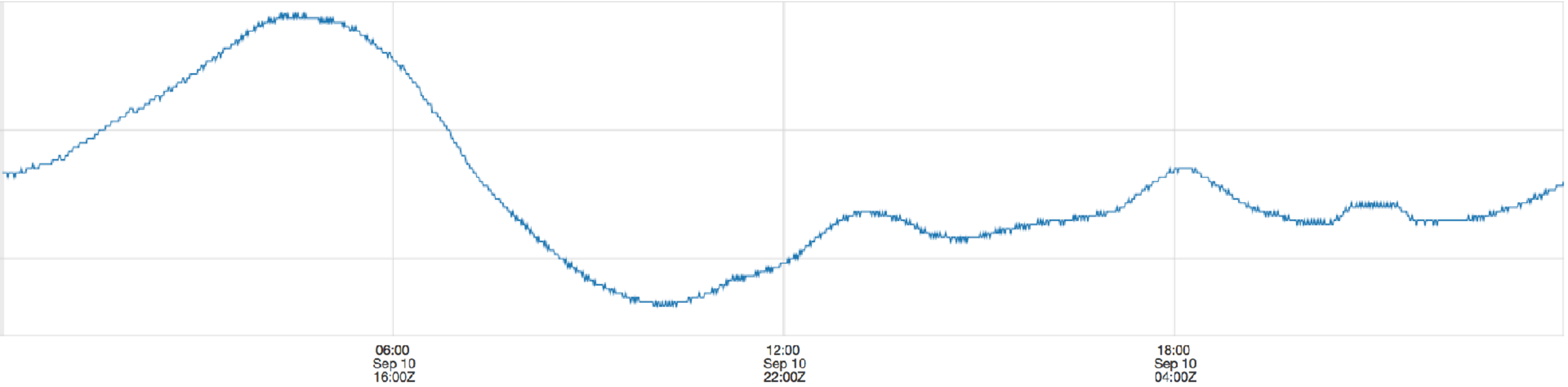
24 hours



C2 Graal C2 JDK9 Graal JDK9

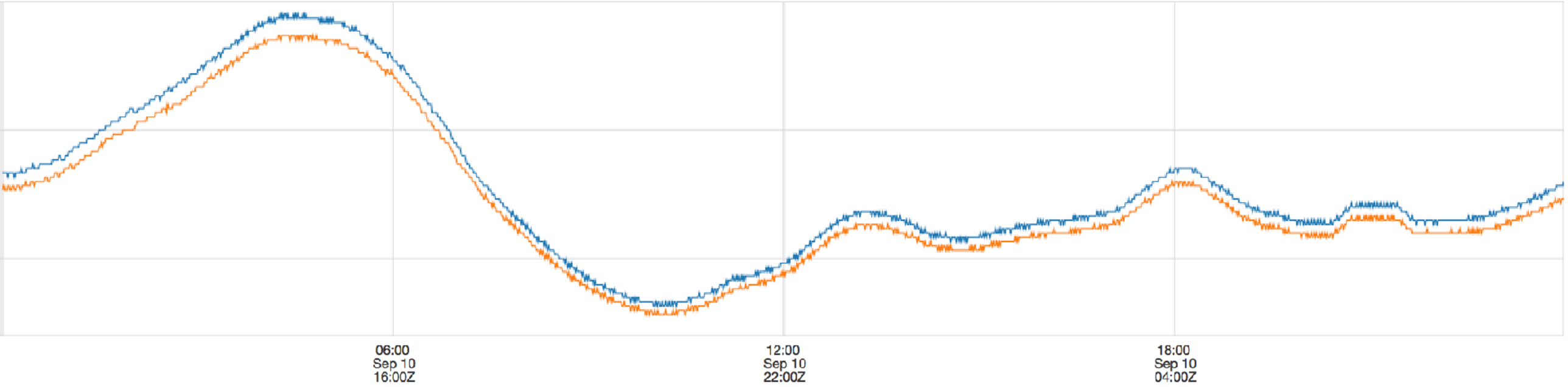
PS SCAVENGE CYCLES

movingavg(60)



PS SCAVENGE CYCLES

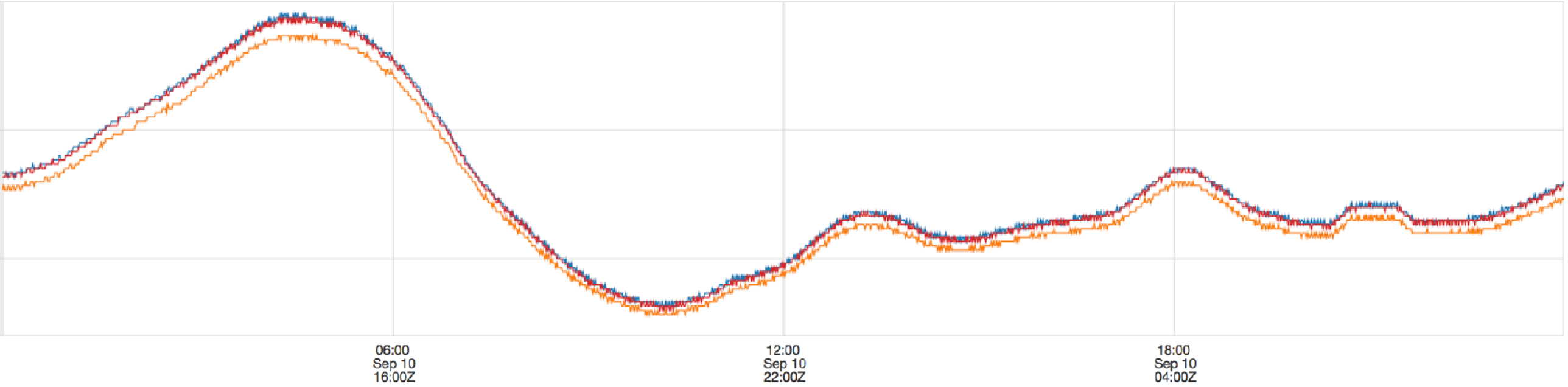
movingavg(60)



C2 Graal

PS SCAVENGE CYCLES

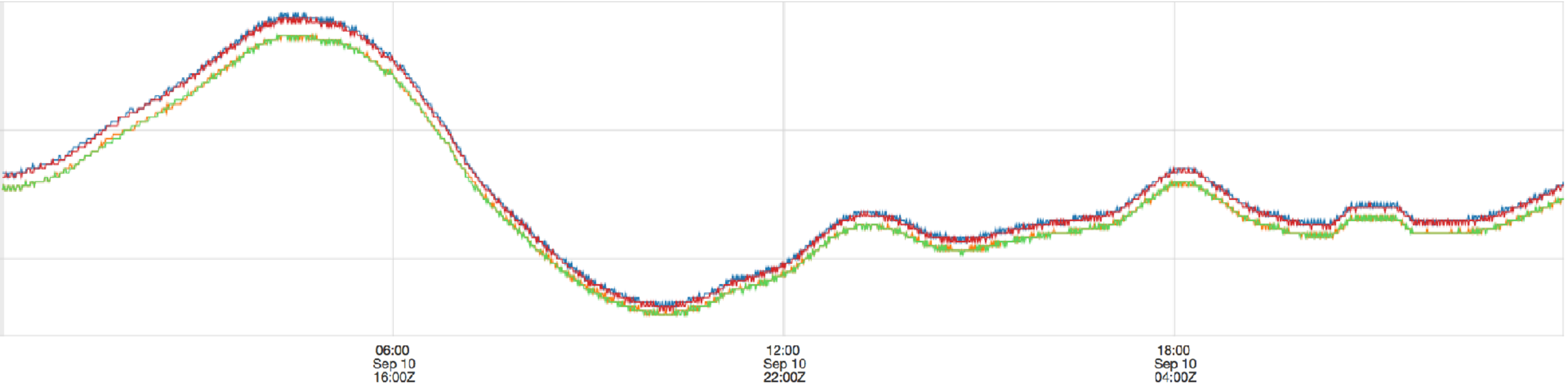
movingavg(60)



C2 Graal C2 JDK9

PS SCAVENGE CYCLES

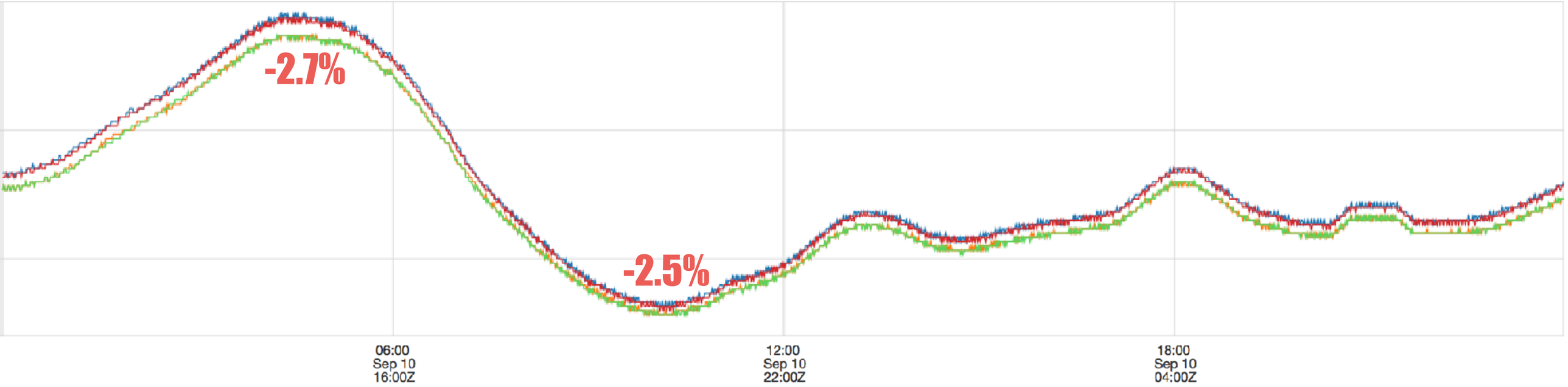
movingavg(60)



C2 Graal C2 JDK9 Graal JDK9

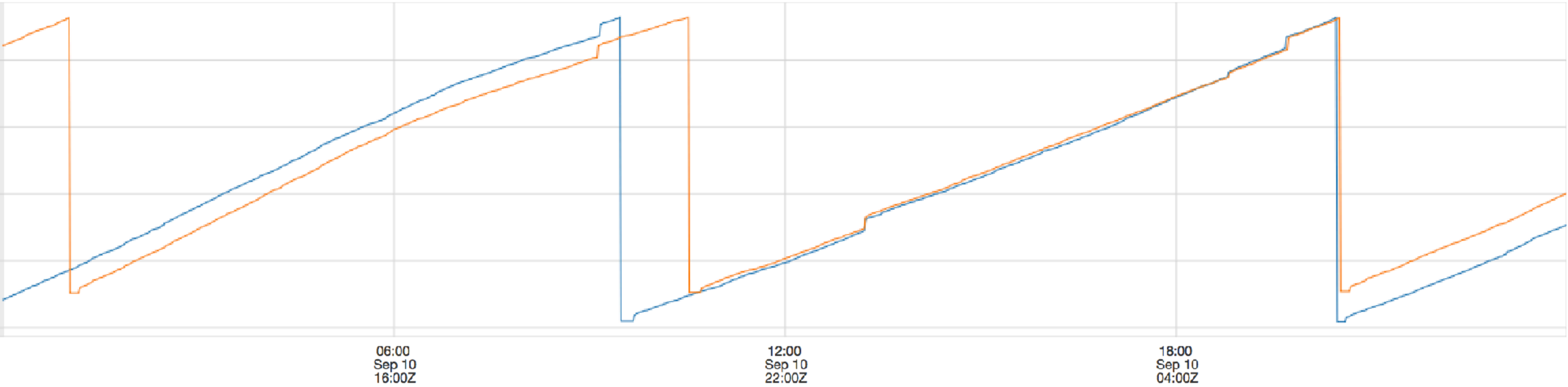
PS SCAVENGE CYCLES

movingavg(60)



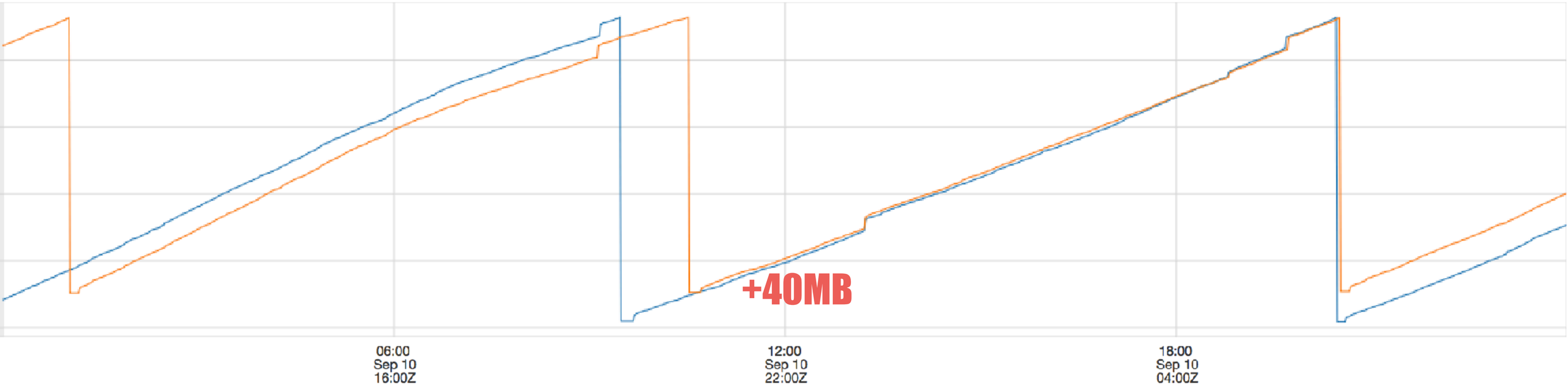
C2 Graal C2 JDK9 Graal JDK9

OLD GEN USED



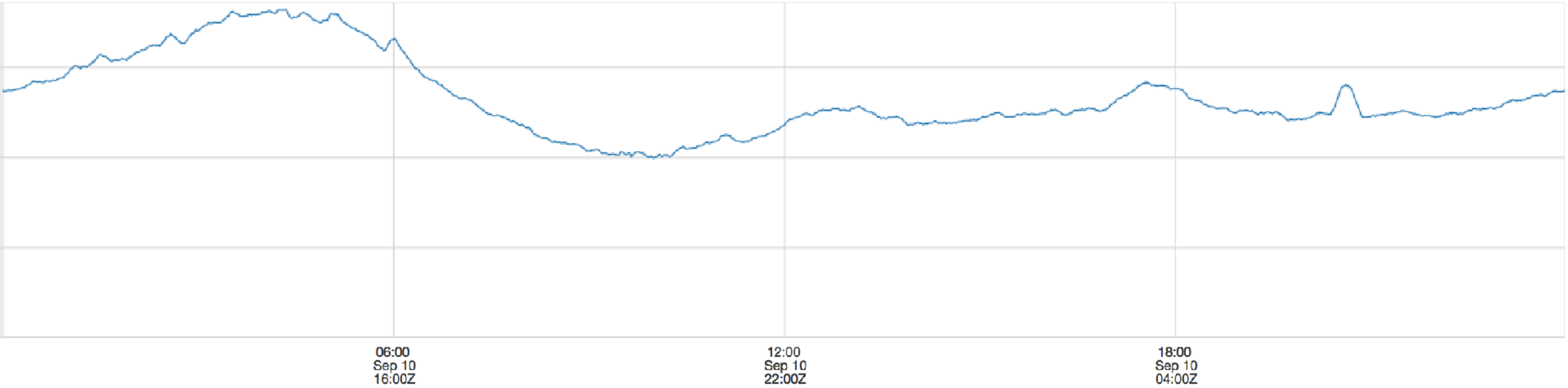
C2 Graal

OLD GEN USED

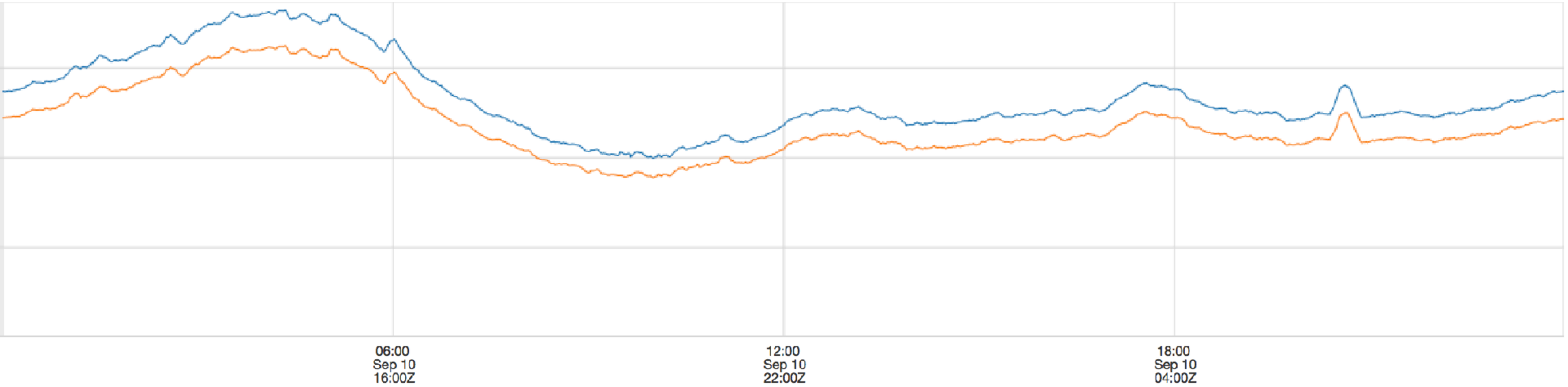


C2 Graal

USER CPU TIME

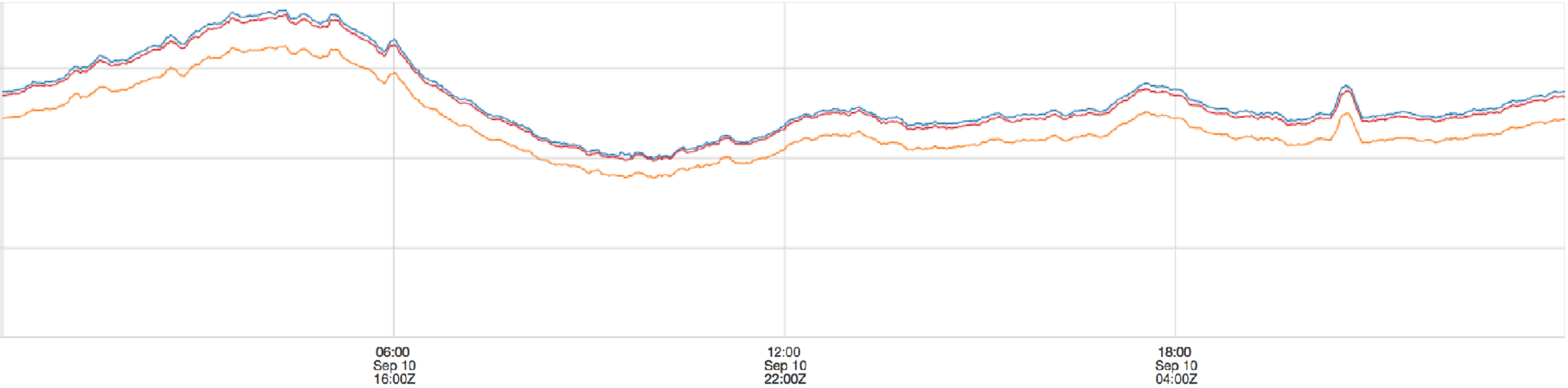


USER CPU TIME



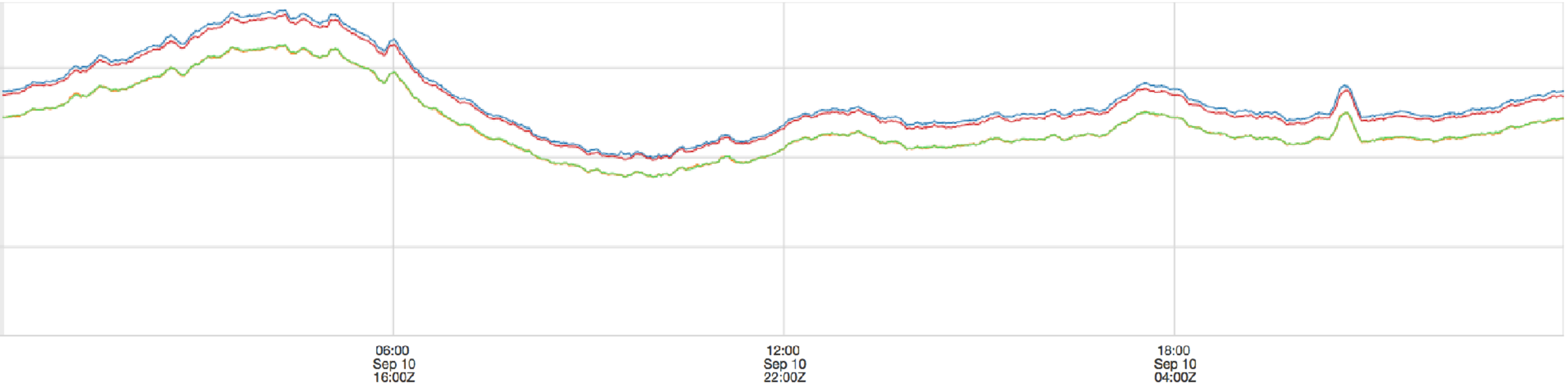
C2 Graal

USER CPU TIME



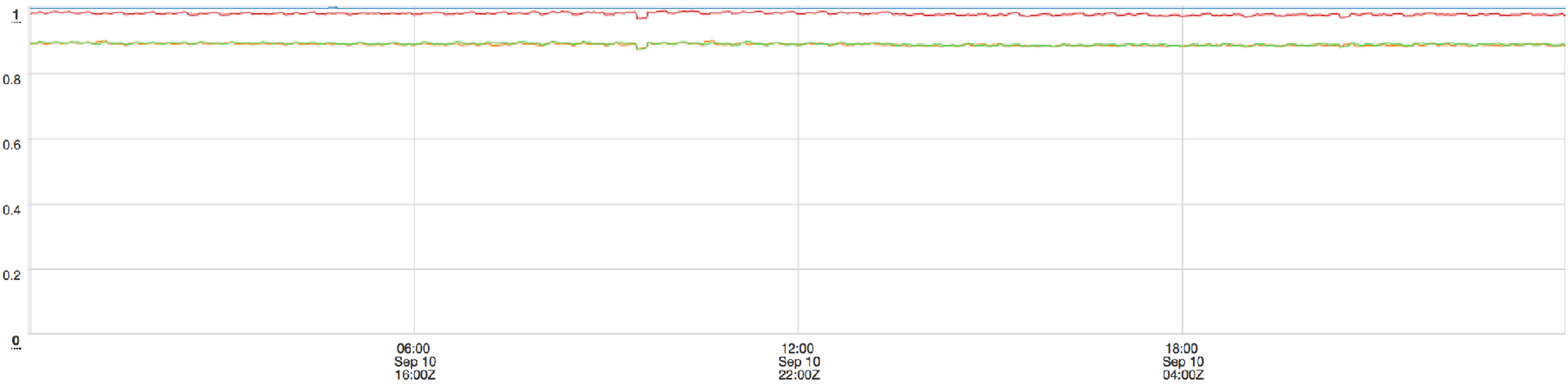
C2 Graal C2 JDK9

USER CPU TIME



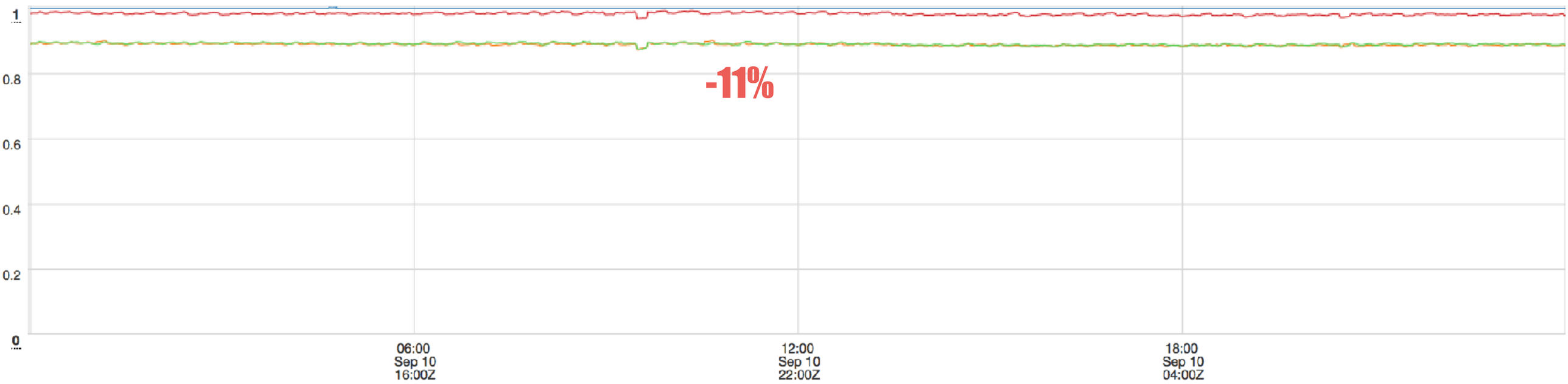
C2 Graal C2 JDK9 Graal JDK9

USER CPU TIME - RATIO



C2 Graal C2 JDK9 Graal JDK9

USER CPU TIME - RATIO



C2 Graal C2 JDK9 Graal JDK9

TWEET SERVICE

- Runs 100% on Graal in production
  - (you should tweet right now to test it #twittervmteam)
  - As are a bunch of other services
  - Including important ones like user service and social graph service

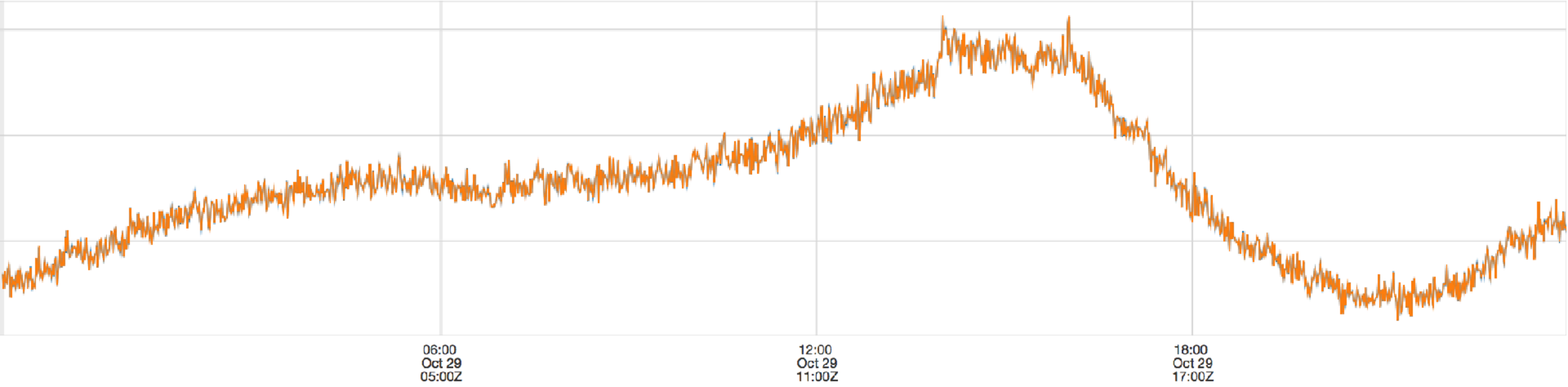
Is it escape analysis?

TWEET SERVICE

- Let’s turn off escape analysis…

REQUESTS/SEC (NO EA)

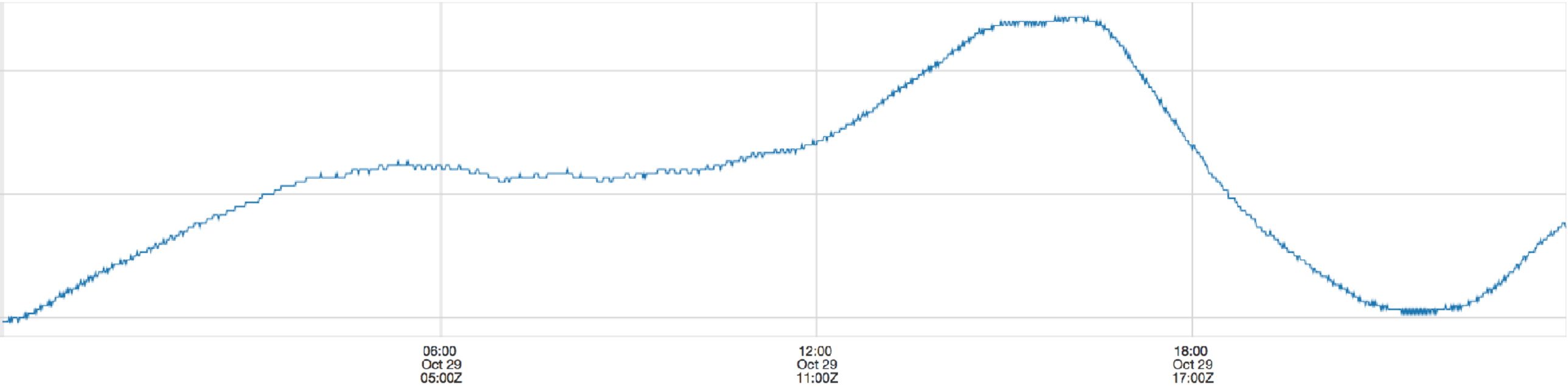
24 hours



C2 Graal

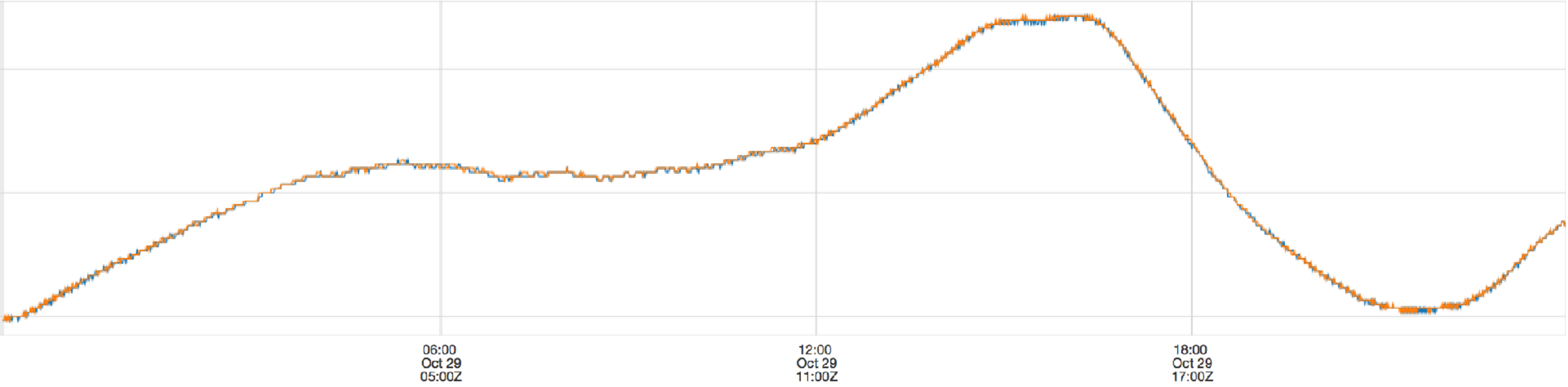
PS SCAVENGE CYCLES (NO EA)

movingavg(60)



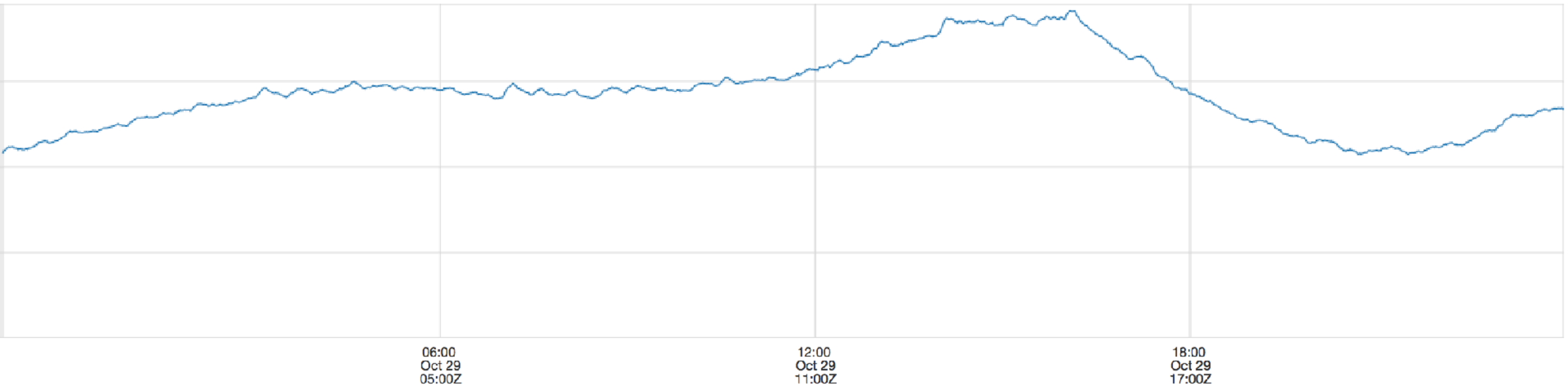
PS SCAVENGE CYCLES (NO EA)

movingavg(60)

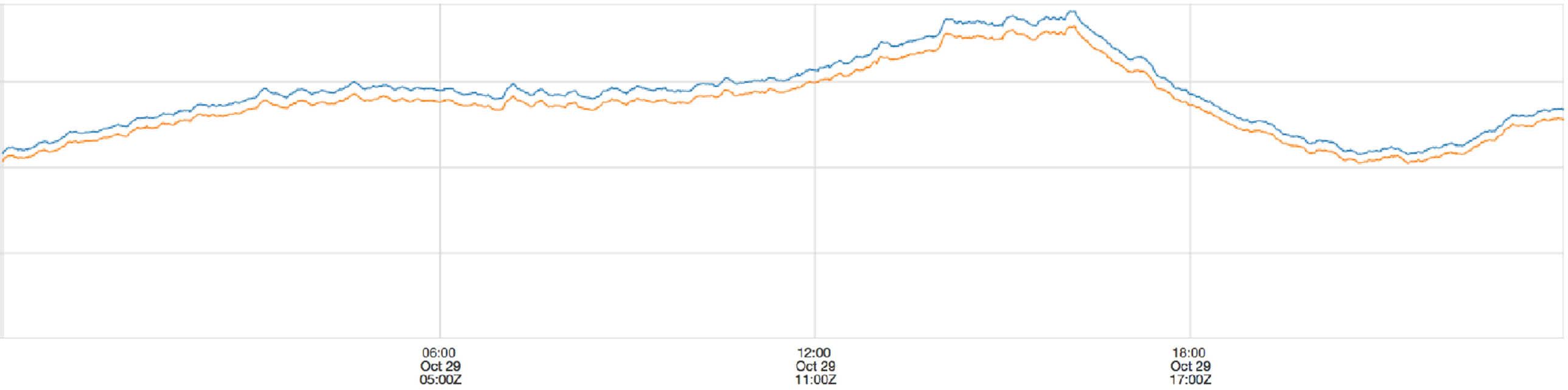


C2 Graal

USER CPU TIME (NO EA)

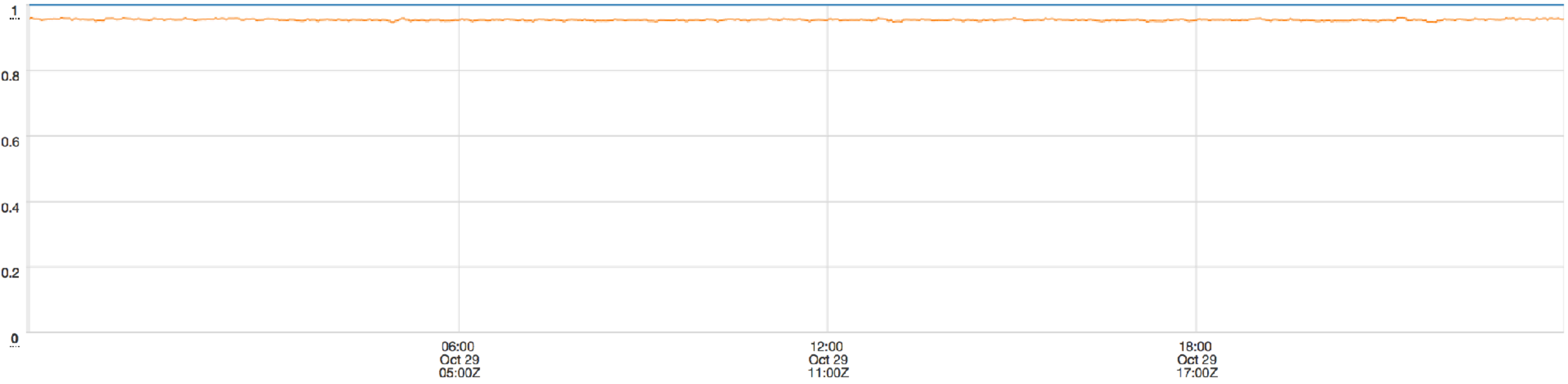


USER CPU TIME (NO EA)



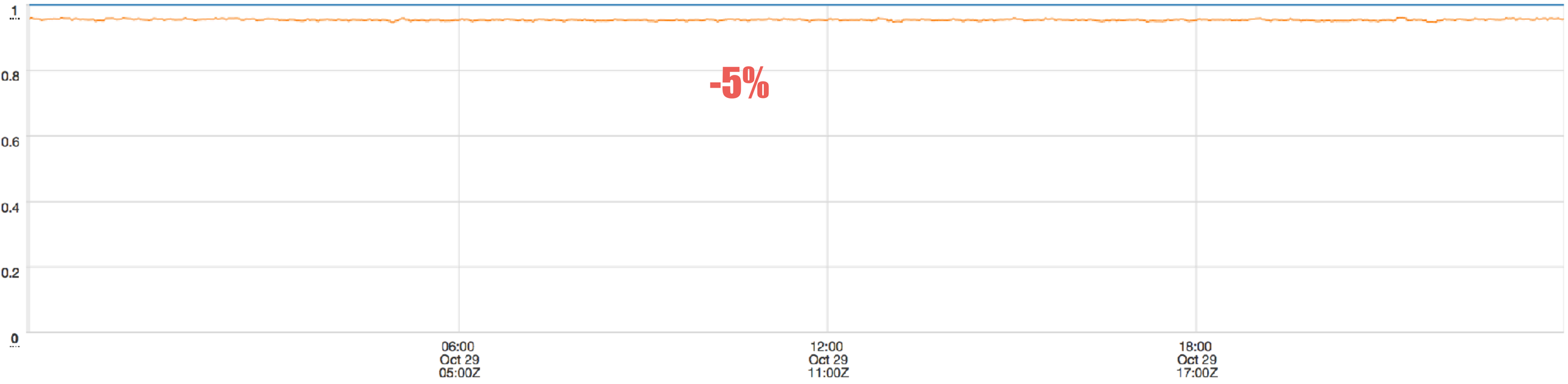
C2 Graal

USER CPU TIME - RATIO (NO EA)



C2 Graal

USER CPU TIME - RATIO (NO EA)



C2 Graal

How much $$$?

YOURCOMPANY.COM

- Let’s do a little back-of-the-envelope calculation…

CLOUD HOSTING PRICES

1 CPU Core

$0.0195/hr

+$0.0115/hr for Windows

1 Core

AMD Opteron™	Intel® Xeon®
$0.024	$0.0506
/ hour	/ hour
+0.0133 $/hour	+0.0266 $/Hour
(Windows Server)	(Windows Server)

Charged Item	Units	Price per Unit (1)
CPU Units	1 Core	$6.00 per month

Resource	hour
CPU core	$0.0112

Specification	CPU	RAM	Cost/h	Cost/mo
1 Core 2GB RAM	1	2	$0.02614	$18.8208
1 Core 4GB RAM	1	4	$0.04248	$30.5856
2 Core 4GB RAM	2	4	$0.05228	$37.6416

CLOUD HOSTING PRICES

$170/year

1 CPU Core

$0.0195/hr

+$0.0115/hr for Windows

$210/year

1 Core

AMD Opteron™	Intel® Xeon®
$0.024	$0.0506
/ hour	/ hour
+0.0133 $/hour (Windows Server)	+0.0266 $/Hour (Windows Server)

$72/year

Charged Item	Units	Price per Unit (1)
CPU Units	1 Core	$6.00 per month

Specification	CPU	RAM	Cost/h	Cost/mo
1 Core 2GB RAM	1	2	$0.02614	$18.8208
1 Core 4GB RAM	1	4	$0.04248	$30.5856
2 Core 4GB RAM	2	4	$0.05228	$37.6416

$85/year

Resource	hour
CPU core	$0.0112

$98/year

YOURCOMPANY.COM

- That averages to: **$127/year per CPU core**
- So, how many cores are we talking?

AMAZON, GOOGLE, MICROSOFT, …

- Bloomberg in 2014

2 Million Servers

Amazon has 11 cloud regions across the world, said James Hamilton, an Amazon distinguished engineer, during a presentation at re:Invent. Each region has multiple sets of data centers, and there are 28 total sets across the world. Each of those has one or more data centers, with a typical facility containing 50,000 to 80,000 servers. A conservative estimate puts Amazon over 1.5 million servers globally. Lydia Leong, an analyst at research firm Gartner, puts it at 2 million or more.

By comparison, Rackspace Hosting has a little over 100,000 servers spread across six data centers. Google has three regions with eight total sets, and Microsoft has 17 regions. Got all that? Last year, Steve Ballmer, then Microsoft's CEO, said the company had over a million servers within its data center infrastructure and that Google had even more.

HTTPS://WWW.BLOOMBERG.COM/NEWS/2014-11-14/5-NUMBERS-THAT-ILLUSTRATE-THE-MIND-BENDING-SIZE-OF-AMAZON-S-CLOUD.HTML

YOURCOMPANY.COM < GOOGLE

- Random number: 10,000 servers
- Each CPU (usually) has 24 cores

$127/year per CPU core * 24 * 10,000 = $3,048,000

$3,048,000 * 11% = $335,280

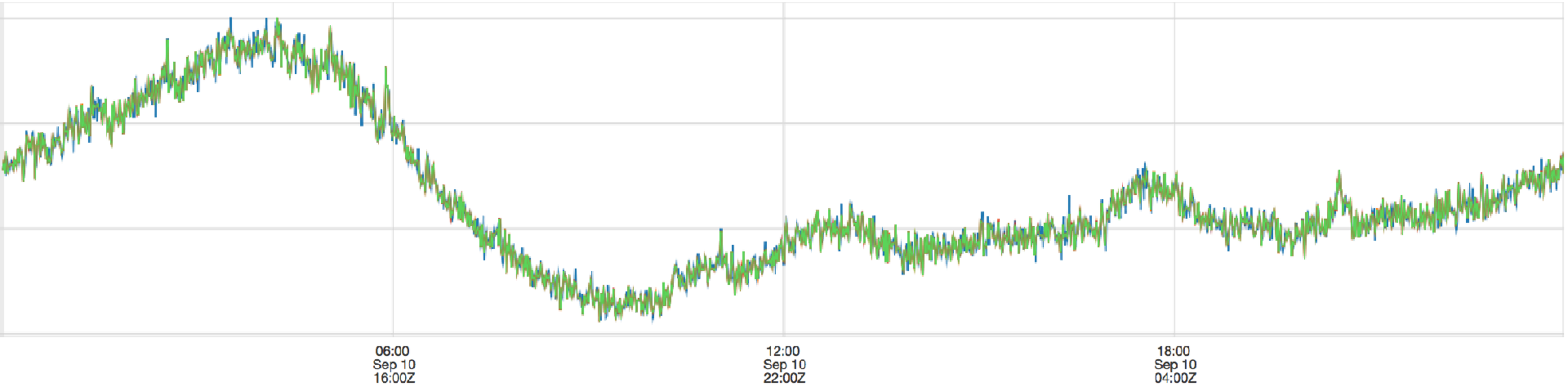
Can we get more?

YES, WE CAN

- About an afternoon later of manually fiddling around with inlining parameters…

REQUESTS/SEC

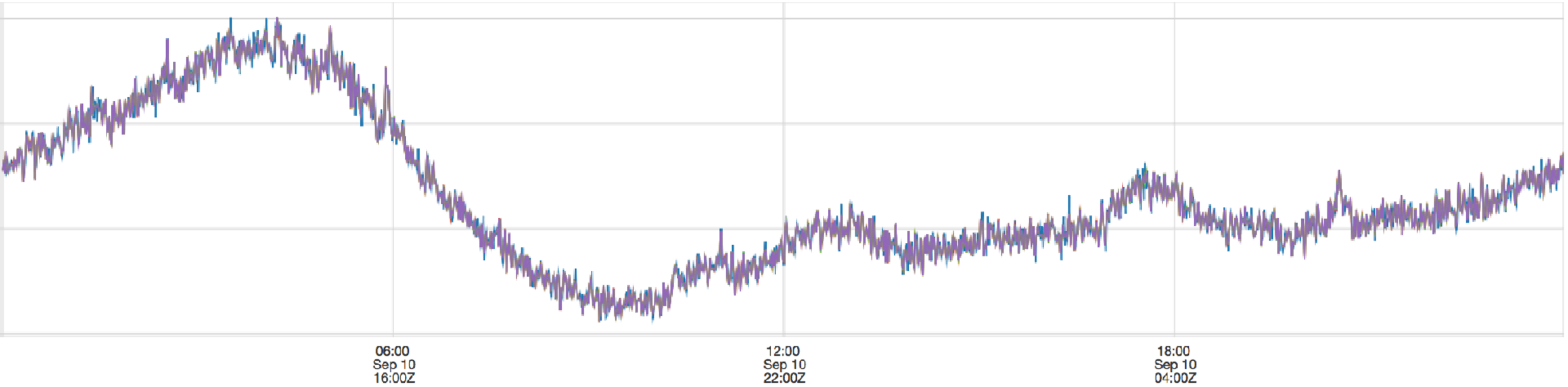
24 hours



C2 Graal C2 JDK9 Graal JDK9

REQUESTS/SEC

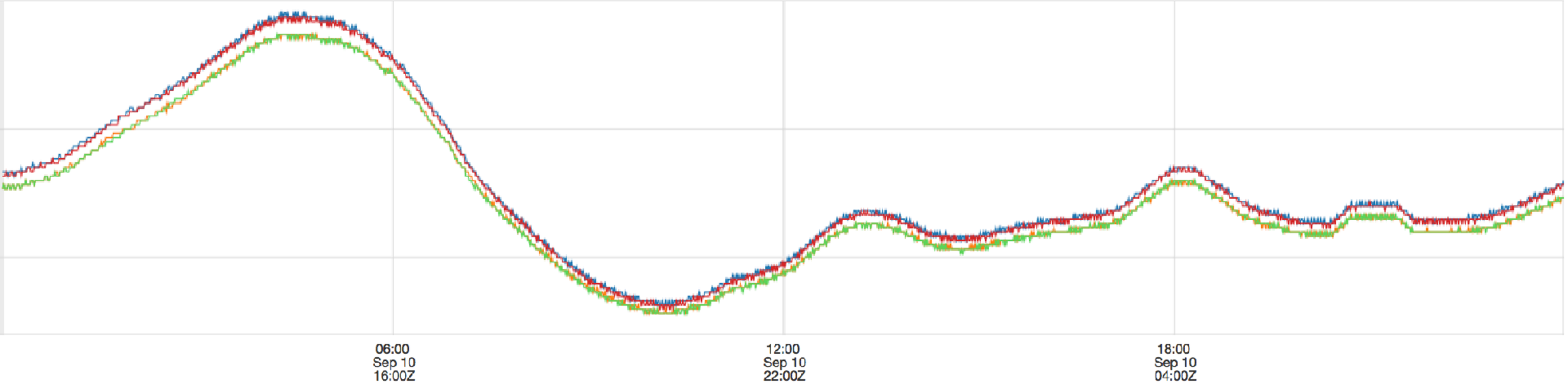
24 hours



C2 Graal C2 JDK9 Graal JDK9 Ex

PS SCAVENGE CYCLES

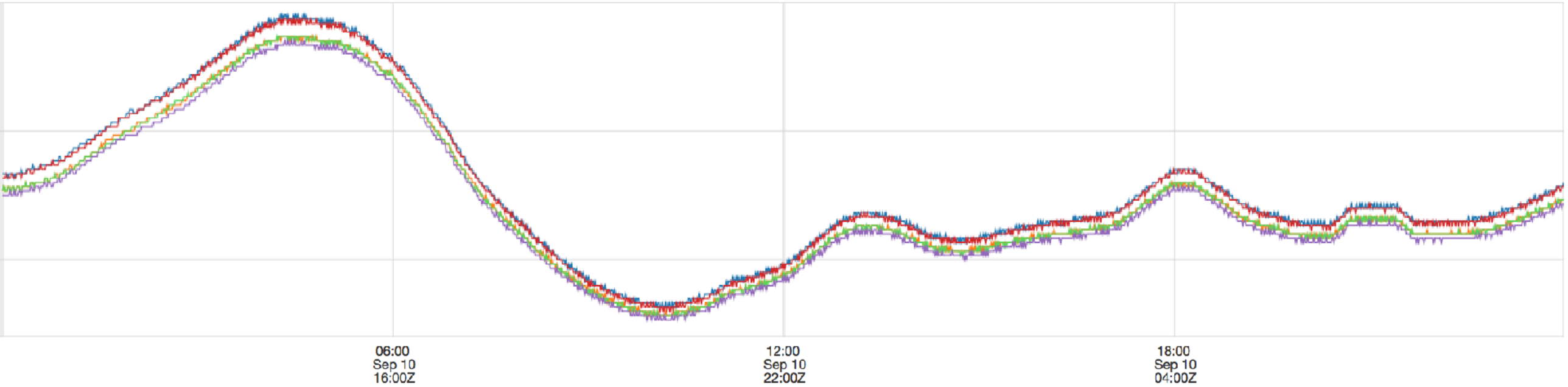
movingavg(60)



C2 Graal C2 JDK9 Graal JDK9

PS SCAVENGE CYCLES

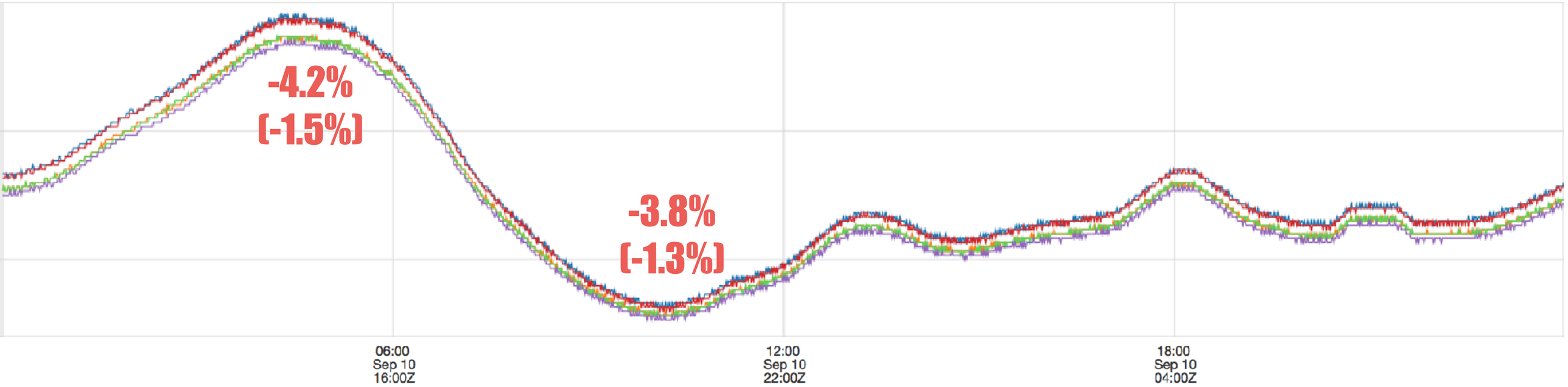
movingavg(60)



C2 Graal C2 JDK9 Graal JDK9 Ex

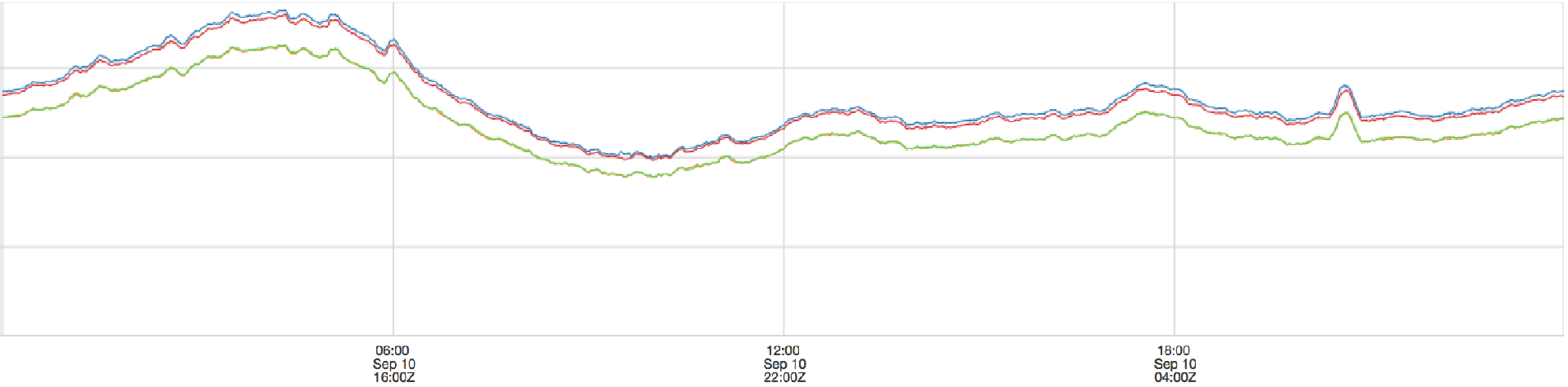
PS SCAVENGE CYCLES

movingavg(60)



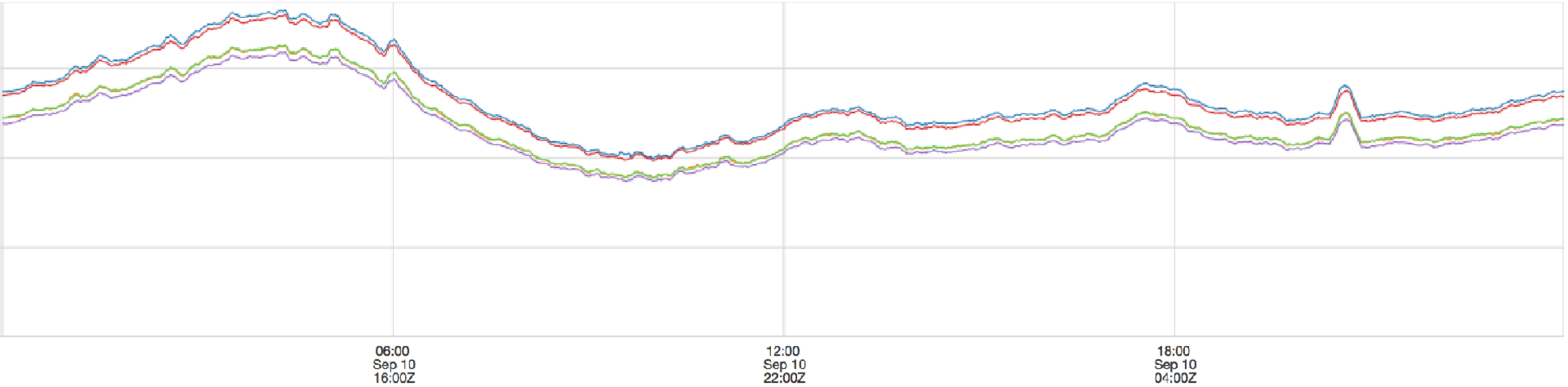
C2 Graal C2 JDK9 Graal JDK9 Ex

USER CPU TIME



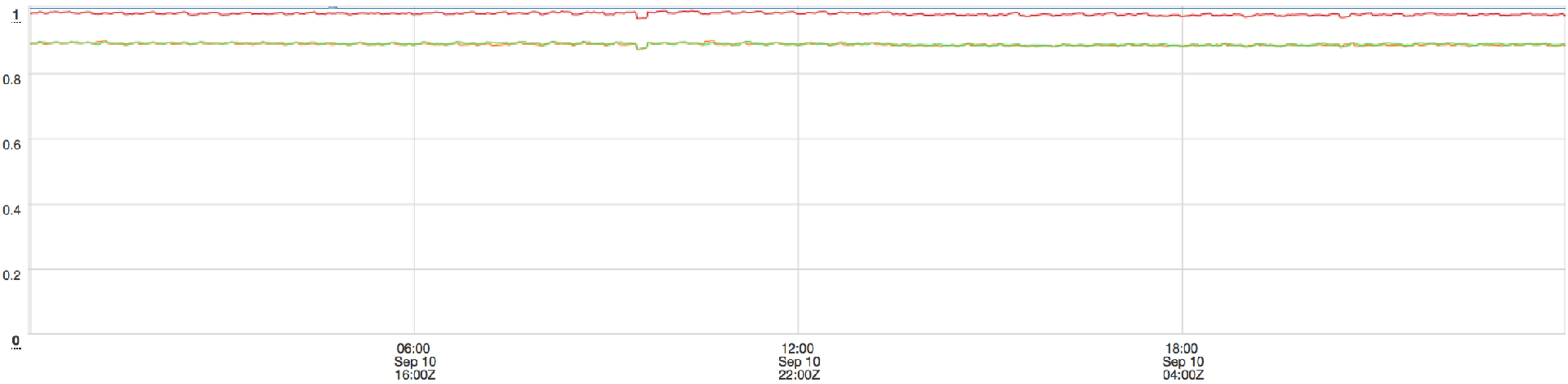
C2 Graal C2 JDK9 Graal JDK9

USER CPU TIME



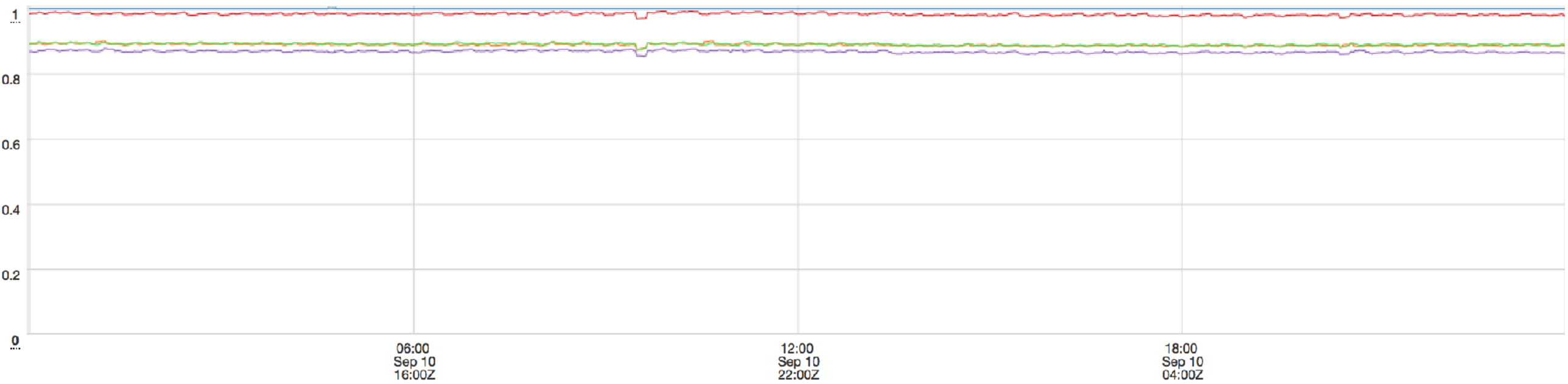
C2 Graal C2 JDK9 Graal JDK9 Ex

USER CPU TIME - RATIO



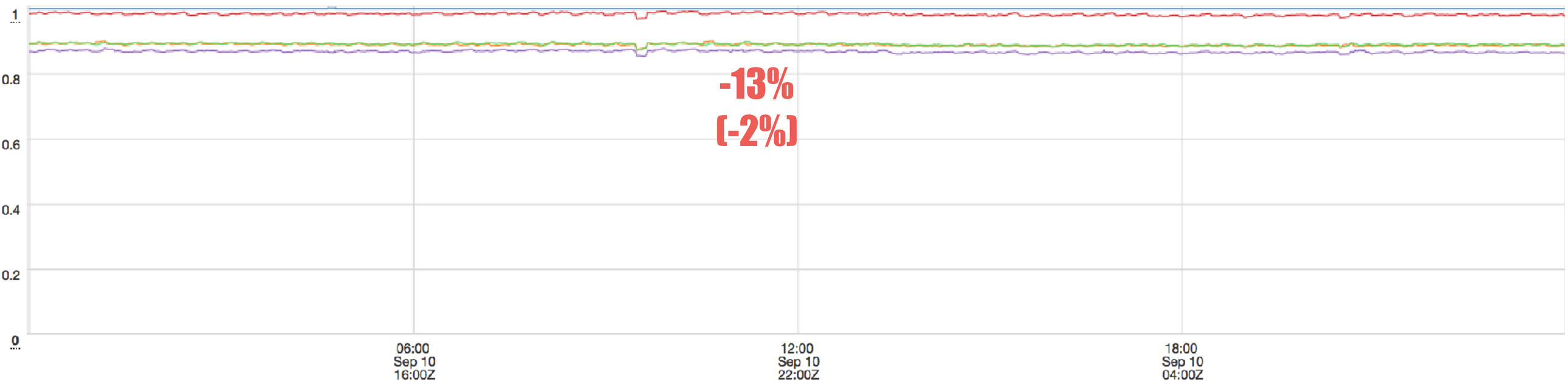
C2 Graal C2 JDK9 Graal JDK9

USER CPU TIME - RATIO



C2 Graal C2 JDK9 Graal JDK9 Ex

USER CPU TIME - RATIO



C2 Graal C2 JDK9 Graal JDK9 Ex

Any official support?

WELL...

- JDK 9 contains Graal
  - Because AOT is using it
- And it can also be used as a JIT compiler:

```
java -XX:+UnlockExperimentalVMOptions -XX:+EnableJVMCI -XX:+UseJVMCICompiler
```

ALSO...

[10] RFR(S) 8186462: [Graal] build Graal regardless AOT build

Vladimir Kozlov vladimir.kozlov at oracle.com
Mon Aug 28 20:18:13 UTC 2017

- Previous message: JDK10/RFR(L): 8167199: Add C2 SPARC intrinsic for BigInteger::multiplyToLen() method.
- Next message: [10] RFR(S) 8186462: [Graal] build Graal regardless AOT build
- **Messages sorted by:** [date] [thread] [subject] [author]

---

http://cr.openjdk.java.net/~kvn/8186462/webrev/
https://bugs.openjdk.java.net/browse/JDK-8186462

We are planning to have Graal as experimental JIT compiler on linux-x64
in next release. For that we should enable its build independent from
AOT build.

Thanks,
Vladimir

ALSO...

https://bugs.openjdk.java.net/browse/JDK-8166462

We are planning to have Graal as experimental JIT compiler on linux-x64 in next release. For that we should enable its build independent from AOT build.

Thanks

PROJECT METROPOLIS

CFV: New Project: Metropolis

John Rose john.r.rose at oracle.com
Sat Sep 30 05:43:11 UTC 2017

- Previous message: CFV: New Project: JDK Updates
- **Messages sorted by:** [date] [thread] [subject] [author]

I hereby propose the creation of the Metropolis Project with Vladimir Kozlov as the Lead and the HotSpot Group as the sponsoring Group.

In accordance with the OpenJDK guidelines [1], this project will provide a venue to explore and incubate advanced "Java-on-Java" implementation techniques for HotSpot, the OpenJDK implementation of the Java virtual machine. Our starting point is earlier proposals [2] for using the Graal compiler and AOT static compilation technology to replace the HotSpot server compiler, and possibly other components of HotSpot.

LASTLY...

- Most Twitter services are written in Scala
  - And that's the main reason for this big improvement
- Java is a whole different story
  - It is possible to see improvements there as well
  - Likely not as big
  - But that topic needs its own talk, maybe next time

#TWITTERVMTEAM