Microsoft has acquired jClarity!

Today we are **delighted to** announce that Microsoft has acquired jClarity (see Microsoft's official statement).

It's always been jClarity's core mission to support the Java ecosystem. We started with our world-class performance tooling and then later became a leader in the AdoptOpenJDK project. Microsoft leads the world in backing developers and their communities, and after speaking to their engineering and programme leadership, it was a no brainer to enter formal discussions. With the passion and deep expertise of Microsoft's people, we'll be able to support the Java ecosystem better than ever before!

jClarity will be bringing its expertise in Java, OpenJDK and performance tuning and applying that to Java workloads on Azure for customers like Adobe, Daimler, and Société Générale. We'll also be supporting other services such as Azure HDInsights and Microsoft affiliated organizations such as Minecraft[1].

The jClarity team will continue to work out in the open in various Java communities. With Microsoft's support, we anticipate being able to contribute back in new and exciting ways.

We look forward to working with Microsoft Azure engineers to make Azure a better platform for Microsoft's Java customers, developers and end-users.

—

To our customers – we will be contacting each of you in the coming weeks to guide you on product and support matters. We cannot express our gratitude enough for the trust you have

placed in us over the past seven years, and we look forward to working with you at Microsoft.

To folks who were looking to explore Censum, Censum as a Service, Illuminate or our commercial AdoptOpenJDK support, please contact me at martijn.verburg@microsoft.com

To our supporters and *Friends of jClarity* community – you have been epic! We look forward to you continuing the journey with us in taming those Java workloads at Microsoft. Stay tuned for more news soon on how we'll interact with you going forward. Don't worry – we are not going away whatsoever!

To my co-founders, the jClarity team and the people behind them, start-ups are **challenging**. Through your arduous work, jClarity has been a successful company and also had a massive positive impact on the Java ecosystem. I could not be prouder of our team who never lost focus on doing the right thing by the community, supporting customers, and building out real technical innovation.

I am looking forward to carrying on with you all and achieving even more at Microsoft. For the curious, I will share some personal thoughts on my (very bare) Medium later on.

Cheers,

Martijn (formerly CEO @ jClarity)

Now proudly the Principal Engineering Group Manager (Java) @ Microsoft.

[1] We know how important this one is to kids and kids at heart! We're humbled that we'll get a chance to help on this icon of the gaming world.

Java CPU release on the 16th of April 2019

Hi all,

At jClarity, we had some questions about the upcoming CPU release for Oracle's Java and whether OpenJDK providers (like AdoptOpenJDK) will be releasing the same patched binaries in the same time frame. We've distilled down the short decision-making process you should go through in order to get the update.

CPU Release by OpenJDK and Oracle JDK

The OpenJDK and 8 and 11 upstream projects led by Red Hat is tagging the same patch set as Oracle on the same day (16th of April). If you have already decided to go to an alternative OpenJDK provider or are considering doing so (we recommend AdoptOpenJDK) then this may be a good time to make the move as you will get the free public update from that provider shortly after the 16th of April!

Steps you should take to update your OpenJDK / Oracle JDK

If you are on Oracle's Java

1. Read the likely Java 11.0.3 release notes or the Java 8u212 release notes and assess if you really need any of the fixes.

2. If you do need the fixes then you can either move to an OpenJDK provider (we recommend AdoptOpenJDK) and get the patches for free *or* you can contact Oracle and take up a subscription to get the patches in

Oracle’s JDK.

Else if you do not need the fixes then you can stay on your existing version and simply not update.

If you are on

OpenJDK from an alternative provider

Simple, you will get the free public update from your provider shortly after the 16th of April.

We hope this short guide was helpful, if there are any questions don't hesitate to contact our engineering team!

Cheers,
Martijn (CEO) and the jClarity team.

Commercial Support for AdoptOpenJDK

Talk to our engineers!

Public Service Announcem

ent: Java Is Still Free!

Hi all,

With the recent changes to Oracle JDK distribution and support, there has

been considerable uncertainty in the Java ecosystem. In particular, there is confusion over the rights to use Oracle JDK vs Oracle's OpenJDK builds vs OpenJDK builds from other providers such as AdoptOpenJDK!

Working with the various providers, the Java Champions (an independent body of Java experts) have put together a comprehensive Java Is Still Free document on the changes. It also covers the choices

you have going forward, and yes Java is Still Free!

The Java Is Still Free document has comments and suggested edit access switched on and will periodically be updated to reflect

the latest accurate information. It is being Disseminated widely and we'd appreciate you sharing this with your colleagues and within your organisations. Please do update the Disseminated doc when you do so!

jClarity will be offering commercial support forAdoptOpenJDK in early 2019, we are happy to give professional advice on which JDK to move forwards with for your organisation! Just drop us a line at

support AT jclarity DOT com.

Cheers,
Martijn

Coping with long haul flights

One of the more interesting non-technical sessions

we had at JCrete (www.jcrete.org) this summer covered the topic of health and travel. The session attracted more than it fair share of attendees many of whom travel extensively as part of their jobs. In fact, one of the

attendees was in Turkey on business during the coup attempt! He managed to get out without anything real happening but he was close enough to the action that he still has anxiety issues from the incident. While this is an

extreme case, being sick on the road can be extra stressful if only because you're in an unfamiliar place and you often have no one to offer comfort or help you navigate foreign systems, language issues, treatment choices,

needing to change travel plans and schedules.

It is because of this that those of us speaking regularly at conference are sensitive too and often recognize when a fellow speaker or

attendee needs medical attention even when it’s not blatantly obvious. Sometimes that even means intervening when the situation gets beyond the persons ability to manage this situation on their own. For example, at

JAX a couple of speakers noted that an attendee who was diabetic was in need of emergency care at an after party. At first everyone thought he was quite drunk but then… Fortunately it all worked out.

Jetlag

One of the bigger topics was; how to deal with the dreaded Jetlag. Jetlag for a frequent traveler is no joke, not getting enough sleep or not sleeping at all for

extended periods can have both immediate and long term effects on one's heath. The burning question at the session quickly became; *"How does one cope with Jetlag?"* Here are some of the points that came out of

that session.

- Restful sleep is the most powerful weapon against jet lag. Prefer it over all over activities including the meal!
- Get a good set of very comfortable

noise canceling headphones. Yes they can be expensive but the good ones are well worth it. I often don’t have them plugged in or when I do, only “white noise” is playing.

- Always fly business class for

long haul flights for frequent flyers. Those who fly infrequently will view this as a luxury, it’s not. It’s a matter of the looking after longer term health effect.

- Schedule the longest leg to be

as long as possible.

- If you're going east, take off as close to the time you'd normally sleep as is possible and land later in the day.
- If you're going west land as early as possible.

- When possible don’t bother shifting into the time zone you’re in.

This works when flying west up to 6 time zones. After that it gets more difficult but you can still limit the

shift. Flying east is a different story. The idea is wake up at your normal time, go about your business, start your business day and then go back to your room and sleep when the day wraps up. This requires a bit of

planning for meals and is not so easy if you have evening events or obligations!

Light and Exercise

Sunlight and exercise is clearly important.

- Be aware of the effects of light both good and bad.
- Try to make sure you make the room dark even in the middle of the day. I traveled from San Francisco to Stockholm on a week in mid-June. At that time of

year the sun sort of sets after 1am and rises about 3:30ish. The rest of the time it’s mostly bright day light. I wasn’t able to make the room dark which was miserable. Next year I did the same trip at

the same time and rented room in the basement. It worked brilliantly!

- When feeling sluggish, a little shot of sun works wonders.
- Walking helps a lot.

Drinks and Pharmaceuticals

Many inexperienced travellers get this wrong!

- Drinking lots of water helps by reducing stress

caused by dehydration.

- alcohol, caffeine and sugar laden/diet drinks dehydrate.
- Melatonin can help you to sleep but won’t keep you there.

The biggest point to

come out of the session is being sick on the road is no fun. There is no shame in not feeling well, it’s happened to all of us at one point in time during our travels. Our hope is that by talking about it, we’ll all feel less

shy about asking for help when we need it or offering help when we see that others suffering from some ailment. If this happens then I'd score this session to be a huge success.

Happy travels!

Kirk (CTO)

No more memory leaks and application pauses!

Start Your Free

Trial!

Find out why my app is slow and tell me how to fix it!

Start
Your
Free

Trial!

What’s required

to make a Container aware Java Runtime

Hi all,

Java today has some fairly serious challenges when it comes to running efficiently on Container based technologies (Docker being the popular choice). This basically boils down to the container reporting incorrect

values for important system resources and/or the Java runtime looking at default operating system locations as opposed to a container provided value. This mismatch in communication leads to the JVM using numbers that

aren’t representative of the resources that a container is actually providing and so Java tends to run inefficiently in a container environment.

For example, at JVM startup the JVM

interrogates the runtime so that is can decide how to best adapt to that run time. For example, max heap is set to 1/4 of real RAM. If the container doesn’t answer how much RAM is available then the JVM will

configure it’s self based on all the real RAM on the machine. There are similar heuristics used when deciding on how many helper threads should be used for the common thread pool, the parallel phases of the garbage

collector and so on.

This is one of the main reasons that very few jClarity customers are running containerised Java applications in production today.

Luckily, Oracle are

stepping up their efforts to provide first class support for Containers with regards to the Java Runtime.

The official ‘bug’ tracking their JDK Enhancement Proposal (JEP) work on this is at:

https://bugs.openjdk.java.net/browse/JDK-8182070.

We'll add some of our own commentary and insights here, deliberately mapping against the titles of that JEP. I'll quote various sections of the

Oracle JEP (cutting out some less interesting stuff) and add our commentary beneath.

Goals

Enhance the JVM and Core libraries to detect running in a container and adapt

to the system resources available to it. This JEP will only support Docker on Linux-x64 although the design should be flexible enough to allow support for other platforms and container technologies. The

initial focus will be on Linux low level container technology such as cgroups so that we will be able to easily support other container technologies running on Linux in addition to Docker.

There’s a danger here that this will make running Java a 2nd class citizen on containers that run on a Mac OS X or Windows machine (two of the main platforms that we see out in the wild alongside Linux). We’re hoping that a

very rough implementation for another container or O/S target will be built (even if it's a throwaway prototype) in order to help validate the flexibility of the design for future container technologies (and

non Linux operating systems).

A mailing list thread suggests that this is being taken into consideration.

Non-Goals

It is not a goal of this JEP to support

any platform other than Docker container technology running on Linux x64.

See above! That said, Java on Linux is a very high percentage of mission critical apps in production

out there today and it makes sense to target this first.

Success Metrics

Success will be measured by the improved efficiency of running multiple

Java containers on a host system with out of the box options.

We’d like to see some before and after numbers on this. CPU, memory, socket, disk – latencies and throughputs under

certain loads. In fact if anyone has already run such tests on their applications and are willing to share the results we'd love to hear from you!

Motivation

Container technology is becoming more and more prevalent in Cloud based applications. This technology provides process isolation and allows the

platform vendor to specify limits and alter the behavior of a process running inside a container that the Java runtime is not aware of. This causes the Java runtime to potentially attempt to use more system

resources than are available to it causing performance degradation or even termination.

Totally agree here – we ourselves are increasingly using Docker for development and QA purposes but would

not consider it in PRD yet due to the concerns listed above.

Description

This enhancement will be made up of the following work items:

B. Exposing container resource limits and configuration.

There are several configuration options and limits that can be imposed upon a running container. Not all of these are

important to a running Java process. We clearly want to be able to detect how many CPUs have been allocated to our process along with the maximum amount of memory that we be allocated but there are other

options that we might want to base runtime decisions on.

In addition, since Container typically impose limits on system resources, they also provide the ability to easily access the

amount of consumption of these resources. I intent on providing this information in addition to the configuration data.

I propose adding a new jdk.internal.Platform class that will

allow access to this information. Since some of this information is needed during the startup of the VM, I propose that much of the implementation of the methods in the Platform class be done in the VM and

exposed as JVM_xxxxxx functions. In hotspot, the JVM_xxxxxx function will be implemented via the os.hpp interface.

Here are the categories of configuration and

consumption statistics that will be made available (The exact API is TBD):

isContainerized
Memory Limit
Total Memory Limit
Soft Memory Limit
Max Memory Usage
Current Memory Usage

Maximum Kernel Memory
CPU Shares
CPU Period
CPU Quote
Number of CPUs
CPU Sets
CPU Set Memory Nodes
CPU Usage
CPU Usage Per CPU
Block I/O Weight
Block I/O Device

Weight
Device I/O Read Rate
Device I/O Write Rate
OOM Kill Enabled
OOM Score Adjustment
Memory Swappiness
Shared Memory Size

This looks like a pretty good list to

us, it’s interesting to note that when it comes down to the performance of a managed runtime that you are actually looking at quite a limited set of resources that all balance against each other.

C. Adjusting Java runtime configuration based on limits.

Java startup normally queries the operating system in order to setup runtime defaults for things such as the number

of GC threads and default memory limits. When running in a container, the operating system functions used provide information about the host and does not include the containers configuration and

limits. The VM and core libraries will be modified as part of this JEP to first determine if the current running process is running in a container. It will then cause the runtime to use the container values rather than the

general operating system functions for configuring and managing the Java process. There have been a few attempts to correct some of these issue in the VM but they are not complete. The CPU detection in the VM currently only

handles a container that limits cpu usage via CPU sets. If the Docker –cpu or –cpu-period along with –cpu-quota options are specified, it currently has no effect on the VMs configuration.

The experimental memory detection that has been implemented only impacts the Heap selection and does not apply to the os::physical_memory or os::available_memory low level functions. This

leaves other parts of the VM and core libraries to believe there is more memory available than there actually is.

The Numa support available in the VM is also not correct when running in a

container. The number of available memory nodes and enabled nodes as reported by the libnuma library does not take into account the impact of the Docker –cpuset-mems option which restricts which memory nodes

the container can use. Inside the container, the file /proc/{pid}/self does report the correct Cpus_allowed and Mems_Allowed but libnuma doesn't respect this. This has been verified via the numactl

utility.

To correct these shortcomings and make this support more robust, here's a list of the current cgroup subsystems that we be examined in order to update the internal VM and

core library configuration.

Number of CPUs

Use a combination of number_of_cpus() and cpu_sets() in order to determine how many processors are available to the process and

adjust the JVMs os::active_processor_count appropriately. The number_of_cpus() will be calculated based on the cpu_quota() and cpu_period() using this formula: number_of_cpus() = cpu_quota() /

cpu_period(). Since it's not currently possible to understand the relative weight of the running container against all other containers, altering the cpu_shares of a running container

will have no affect on altering Java's configuration.

Also add a new VM flag that allows the number of CPUs to be overridden. This flag will be honored even if UseContainerSupport is not enabled.

The last option will actually be very useful for those who want to pin their JVM's ergonomics to a subset of the available CPU on a server. For example, your JVM process might not be as important as that Python routine on

the host. It makes sense that you can more easily restrict what the JVM is using.

Total available memory

Use the memory_limit() value from the

cgroup file system to initialize the os::physical_memory() value in the VM. This value will propagate to all other parts of the Java runtime.

We might also consider examining the

soft_memory_limit and total_memory_limit in addition to the memory_limit during the ergonomics startup processing in order to fine tuning some of the other VM settings.

We'd like to see

these extra options enabled. In really latency sensitive applications, being able to tune that last dial is sometimes required!

D. Adding container configuration to error crash logs and Unified JVM

logging.

As as troubleshooting aid, we will dump any available container statistics to the hotspot error log and add container specific information to the

JVM logging system.

This would be invaluable. Virtualisation and containerisation complexities make for some of the trickier analyses that we perform for customers (both with our tooling and

sometimes a human eye). Having to correlate logging information from so many disparate layers can be frustrating. As a side note, the fact that a GC log can provide some useful CPU usage and safe pointing information

actually allows you to diagnose a whole host of performance issues that are not caused by the GC sub system. This in built correlation has proven to be most helpful!

E. Adding a startup flag to

enable/disable this support.

Add a -XX:+UseContainerSupport VM option that will be used to enable this support. The default will be off until this feature is proven.

The Oracle engineers are (once more) showing great wisdom in feature toggling. If you don’t apply this technique to your mission critical apps, it’s time to start looking at things like LaunchDarkly!

F. Configuration change notifications

An additional API will be provided to allow an application to receive a notification when configuration changes occur.

Configuration change events will not necessarily cause the VM and Java core libraries to reconfigure their usage of resources. This support will be optional.

We'd hope to see

this accessible via logs or JMX so there’s a chance to warn the end user that something has changed and that the JVM may or may not have been able to adjust.

Conclusion

Java’s evolution can seem frustratingly slow at times (if you were to read Hacker News and Reddit you’d think that Docker was a production std for several years now)

but we think that the timing of this JEP is about right for Java and will meet the industry need for moving Java applications onto containers in an efficient manner. We look forward to helping test out the early

implementations!

Cheers,
Martijn (CEO) and the jClarity Team.

No more memory leaks and

application pauses!

Start
Your
Free
Trial!

Find out why my app is

slow and tell me how to fix it!

Start
Your
Free
Trial!

Managing Developer Stress

Some years ago I was interviewed by

Oliver White at ZeroTurnaround about the topic of developer stress. In their brief survey they ascertained the top 5 developers stresses of the time:

1. Making deadlines
2. Application

performance issues

3. Is my code good enough?
4. What new stuff do I need to learn?
5. Did I do X in the best way?

I wrote back privately to Oliver at the time but thought it would be

interesting to rehash the topic here as it covers some of the early Agile thinking at the time and serves as a good reminder.

Making

Deadlines

Managing deadlines is something that a lot of developers feel that they cannot learn or is out of their control (e.g. Their manager tells them what the deadline is).

However, managing deadlines is a skill that can definitely be learned! For example, developers can learn to:

- Scope work into manageable (e.g. 1 day) chunks
- Define what _done_ means (95% there

is not done)

- Factor in contingency
- Communicate risks and issues to stakeholders
- Learn to prototype ideas to keep the overall project flowing

There are a number

of tools to assist you in managing the scope and communication around deadlines, but always remember, *“Whatever they tell you, it’s a people problem”*, so developers should look at their communication and

expectation setting first.

Some example tooling that can help:

- Simple Lean/Kanban style planning with sticky notes and a whiteboard
- Electronic versions of the

above (e.g. Atlassian's Greenhopper, Trello, GitHub Issues)

- BDD style tests with an electronic wallboard (therefore the whole business can see exactly where progress is at)

- A publicly shareable issue tracker that tracks timings (e.g. Atlassian's JIRA)

Application

performance issues

Performance and performance tuning is terrifying for most developers because they have no idea where to start. Modern software applications are so

complex that it can feel like finding a needle in a haystack. However, performance and performance tuning is actually a _science_, not an art and definitely not guesswork. Kirk Pepperdine (Our CTO and one of Java's

foremost expert in this field) always hammers home the point *“Measure, don’t guess”*. By following the sort of scientific methodology that Kirk (and others like him) teach, you can systematically track down and fix

performance issues as well as learning to bake performance in right from the beginning. There is a host of tooling that can assist in this area as well, but it’s the methodology that’s truly important. I can’t

recommend Kirk's course enough!

Is my code good enough?

Is my code good enough? A question that all developers

secretly fear. The fact is you can’t definitively state what good code is! The Softwarecraftsmanship folks will beg to differ here ;-). Define what Clean Code is! At the end of the day there are many differing

opinions on what defines ‘good code’. However, there are some clear traits of what is commonly perceived as good code.

- It’s broken up into small classes/methods/blocks each that

perform one thing and one thing well

- Some level of interface is used so alternative implementations can be used
- The code is easy to test against
- You can ascertain by the naming used in the code, what

the code is used for, e.g. It's problem domain

- Common language pitfalls are avoided – type1, the sorts of things that static code analysers warn you against
- Common language pitfalls are

avoided – type2, the sorts of things you pick up in Josh Bloch's "Effective Java" title.

There's of course much more, the list is fairly long!

So my advice is to

try to follow the better practices out there (note I avoid using ‘best practices’, horribly overloaded term), make sure that you’re using the static code analysis tools, try to practice TDD and above all else get a

2nd pair of eyes on the code, preferably via Pair programming to begin with.

What new stuff do I need to

learn?

There’s always the fear of being left behind, can you as a developer guess what’s going to big next? If your technology CV is out of date you might struggle to find

that next job!

Take a deep breath and relax. Most new technologies are simply short lived fads and it’s impossible to keep up with everything! There’s no way a developer can stay up to date with the

latest Java libraries, learn Scala, pick up Clojure, hack some iOS, deploy that all to the cloud on a NoSQL distributed grid environment. See what I mean?

The trick is to identify trends.

Cloud is a trend, so you should learn about the principles behind it, but don’t sweat if you haven’t learned to deploy to the 5+ different Java cloud providers out there today. Functional programming is a trend, learn _why_

(hint – Multi-core processors and concurrency) it is and see if it's something that you need to learn about now or whether you can wait a few years. The same goes for any new craze you read about or hear at a

conference.

At the end of the day, our industry is still re-inventing the wheel, we have memories like Goldfish :-). For example, some of the older hands are certainly chuckling away at this new

functional fad, they were doing it over 20 years ago…

Did I do X in the best way?

This is also a fear at the macro design

level. Developers can fear that if they chose the wrong web framework or picked the wrong app server or designed the n-tier architecture wrong that they’re doomed.

A common problem here is that

developers aren’t standing on the shoulders of Giants, most problems are not actually that unique and there is a wealth of free advice on sites like programmers.stackexchange on mailing lists, in books, at conferences and of

course at your local user groups.

Another common problem is in the prototyping space, not enough developers prototype the risky parts of their projects soon enough. Worried that Jboss might not

support AMQP? Spend a day upfront proving that it does or doesn’t!

Cheers,
Martijn (CEO) and the jClarity Team!

No more

memory leaks and application pauses!

Start Your Free Trial!

Find out why my app is slow and tell me how to fix it!

Start Your Free Trial!

Censum 3.0.25 has been

released!

Hello fellow performance tuners!

We are pleased to announce the next minor release of Censum! This release fixes several small issues with parsing

of GC logs and adds some new reports around G1.

Release Notes

- Added a G1 Failure Reason Report
- Fixed some rare divide by 0 bugs
- Fix Bytes Promoted Report reporting

incorrect numbers under certain conditions

Download your free trial today! OR if you are already a customer then you can download the latest version from http://download.jclarity.com using your

original purchase email address and UUID.

Contact support@jclarity.com if you need any help.

Cheers,
Martijn (CEO) and the jClarity team.

No more memory leaks and application pauses!

Start Your Free Trial!

Find out why my app is slow and tell me how to fix it!

Start Your Free Trial!

Censum 3.0.22 has been

released!

Hello fellow performance tuners!

We are pleased to announce the next minor release of Censum! This release fixes several small issues with parsing

of GC logs and adds some new capabilities around free lists for CMS and Mixed Collections for G1.

Release Notes

- Added support to parse CMS FLS and promotion failure

records

- Added new Free List views for CMS
- Added new Mixed Collection ratio analytic for G1
- Misc parsing fixes

The sharp eyed amongst you will see the version number is set to 3.0.22 or 3.0.23 –

don’t panic this is a side effect of our build system!

you are already a customer then you can download the latest version from http://download.jclarity.com using your original purchase email address and UUID.

Contact support@jclarity.com

if you need any help.

Cheers,
Martijn (CEO) and the jClarity team.

No more memory leaks and

application pauses!

Start
Your
Free
Trial!

Find out why my app is

slow and tell me how to fix it!

Start
Your
Free
Trial!

Java 9 (Jigsaw) Hackday

and Exercises

Hi all,

jClarity has long been involved in the Java and OpenJDK communities. Martijn, Kirk, Ben

and John are all regular contributors to the Adopt OpenJDK programme, represent the London Java Community on the Java standards body, as well as contributing to several other Java related communities and open source

projects.

On the 22nd of April we participated in a Java 9 Hackday, exploring the new modularity system (*Jigsaw*), the new JRE binary creation tool (*jlink*) and the new REPL for Java (*jshell*). This event

was co-ordinated by the Virtual Java User Group and included the London Java Community and other Java User Groups from around the world including but not limited to Iceland, South Africa, Australia and Germany!

The recording of this 5 hour session can be found here: https://youtu.be/y868lMk6NtY

There is an accompanying Adopt OpenJDK jdk9-jigsaw Github repo which holds all of the exercises that you

can follow yourself.

Java 9 modularity in particular is a pretty big paradigm shift and you'll be scratching your head a bit working around Jigsaw's definition of a module and it's interaction with packages, classes,

reflection and much more. We highly recommend going through the exercises to get a heads up of what's coming at in mid 2017!

Cheers,
Martijn (CEO) and the jClarity Team

No more memory leaks and application pauses!

Start Your Free Trial!

Find out why my app is slow and tell me how to fix it!

Start Your Free Trial!

Censum 3.0.20 has been

released!

Hello fellow performance tuners!

We are pleased to announce the next minor release of Censum! This release fixes several small issues with parsing

of GC logs and adds some new analytics. There is now an analytic giving you guidance on what to set the mixed collection ratio too, in order to meet your pause time goals. There is also a new analytic to determine if there

is interference from outside the JVM that's adding to the overall application stop time.

Release Notes

- Several minor parsing fixes
- Added Parsing support for simple

PS logs without PrintGCDetails

- New G1 Mixed Collection Count Analytic
- Improved High Kernel Times Report – Outside interference detection

The sharp eyed amongst you will see the version number is set to 3.0.20 or 3.0.21 –

don’t panic this is a side effect of our build system!

Download your free trial today! OR if you are already a customer then you can download the latest version from http://download.jclarity.com using your original purchase email address and UUID.

Contact support@jclarity.com if you need any help.

Cheers,
Martijn (CEO) and the jClarity team.

No more

memory leaks and application pauses!

Start Your Free Trial!

Find out why my app is slow and tell me how to fix it!

Start Your Free Trial!