

# 16566 - Raising Healthy Java apps in CICS using the Java Health Center

Phil\_Wakelin@uk.ibm.com

CICS Strategy & Design, IBM Hursley UK





SHARE is an independent volunteer-run information technology association that provides education, professional networking and industry influence.







### **Disclaimer**

IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion.

Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.

The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.



### SHARE, Educate - Network - Influence

# Agenda

- What is the Health Center
- Installing into CICS Explorer
- Monitoring JVM server





IBM Developer Kits for Java ibm.biz/javasdk



Very low overhead live monitoring capability for Java and Node.js

- Pre-installed in <u>IBM SDKs for Java</u> since Java 5
- Built on live sampling within the JVM

Provides insight into runtime and application execution, including

- Memory and CPU usage
- Garbage Collection
- Application execution

Visualization provided via Eclipse Client UI

- Available from Eclipse Marketplace
- Available from <u>IBM Support Assistant</u>
- Eclipse p2 install

Data access API provided

- Allows creation of custom monitoring tools
- Full <u>API Javadoc available</u>







roperty	Value											
Java parameters												
	-Dcom.ibm.oti.vm.bootstrap.library.path=c:\javabuilds\java6sr8\sdk\jre\bin											
	-Dconsole.encoding=Cp850											
	•Djava.class.path=.;c:\javabuilds\java6sr8\sdk\lib;											
	-Djava.ext.dirs=c:\javabuilds\java6sr8\sdk\jre\lib\ext											
	-Djava.home=c:\javabuilds\java6sr8\sdk\jre											
	-Djava.library.path=c:\javabuilds\java6	sr8\sdk\jre\bin;.;c:\javabuilds\java6s	r8\sdk\jre\bin;c:\javal	ouilds								
	-Dsun.boot.library.path=c:\javabuilds\	ava6sr8\sdk\jre\bin										
	<ul> <li>Dsun.java.command=TestApplication</li> </ul>											
	-Dsun.java.launcher=SUN_STANDARD			1								
	-Duser.dir=C:\java\testApplication											
	-Xdump -Xhealthcenter											
	-Xjcl:jclscar_24											
	_j2se_j9=71168											
	_org.apache.harmony.vmi.portlib											
-	_port_library			*								
* [												
🖕 Java Runtime Envir	onment 🛙 🗖 🗖	🛱 System 🖾										
Property	Value	Property	Value									
Agent version	1.1.0.20100219	Architecture	x86									
	JRE 1.6.0 IBM Windows 32 build o	Host name	CORBIN-PC									
Full version	the second second second p		2									
Full version Java home	c:\iavabuilds\iava6sr8\sdk\ire	Number of available processors	4									
Full version Java home Java vendor	c:\javabuilds\java6sr8\sdk\jre IBM Corporation	Number of available processors Operating system	Windows 7									
Full version Java home Java vendor Java virtual machine i	c:\javabuilds\java6sr8\sdk\jre IBM Corporation IBM J9 VM	Number of available processors Operating system Operating system version	Windows 7 6.1 build 7600									
Full version Java home Java vendor Java virtual machine r Process id	c/javabuilds/java6sr8/sdk/jre IBM Corporation IBM J9 VM 12160	Number of available processors Operating system Operating system version	Windows 7 6.1 build 7600									
Full version Java home Java vendor Java virtual machine r Process id Version	c\javabuilds\java6sr8\sdk\jre IBM Corporation IBM J9 VM 12160 1.6	Number of available processors Operating system Operating system version	Windows 7 6.1 build 7600									
Full version Java home Java vendor Java virtual machine i Process id Version	c:\javabuilds\java6sr8\sdk\jre IBM Corporation IBM 8 VM 12160 1.6	Number of available processors Operating system Operating system version	Windows 7 6.1 build 7600									
Full version Java home Java vendor Java virtual machine i Process id Version	c:\javabuilds\java6st8\sdk\jre IBM Corporation IBM.39 VM 12160 1.6	Number of available processors Operating system Operating system version	2 Windows 7 6.1 build 7600									

#### Environment

Hardware and Operating System Configuration Process environment and configuration Highlights incorrect or non-standard configurations

### Memory Utilization



Detect native memory leaks in application Determine if external forces are using more memory View components using the most native memory





Visualizes process CPU usage over time Visualizes system CPU usage over time







#### Garbage Collection

Visualizes heap usage and GC pause times Identifies memory leaks Suggests command-line and tuning parameters

### Object Allocations 🖄

Understand types of data being allocated Determine which code is allocating data



hread name filter:	Apply Clea		N	umber o	f threads	
hread name	Thread state	20.0				
nain	RUNNABLE					
T Compilation Thread	RUNNABLE	150				
ignal Dispatcher	RUNNABLE	- 15 <i>0</i>				
ic Slave Thread	RUNNABLE	d C				
inalizer thread	RUNNABLE	E 10.0				_
MI TCP Accept-1972	RUNNABLE	2				
lealth Center trace subscriber	RUNNABLE					
T=0:P=800369:O=0:port=55465	RUNNABLE	5.0				
MI TCP Connection(1)-9.20.187.149	RUNNABLE					
ttach API wait loop	RUNNABLE	0.0				
MI TCP Connection(3)-9.20.187.149	RUNNABLE		0:11	0:11	0:11	0:11
T=0:P=800369:O=0:TCPTransport	RUNNABLE		elap	sed tim	e (minutes)	
VT=1	RUNNABLE					
VT=2	RUNNABLE	2 Thread	details 🖂			
VT=3	RUNNABLE					
MI Scheduler(0)	TIMED_WAITING	Owned m	ionitor nam	e		
hread-3	WAITING	java.net.S	ocksSocket	Impl@1	19c119c	
top JMX Server on shutdown	WAITING					
MX server connection timeout 23	WAITING					
VT=4	WAITING					
		٠				E.
		Contended	monitor			



List of current threads and states Number of threads over time See contended monitors





Always-on profiling shows application activity Identifies the hottest methods in an application

Full call stacks to identify where methods are being called from and what methods they call

No byte code instrumentation, no recompiling

#### Lock Profiling



Always-on lock monitoring

Allows the usage of all locks to be profiled Identifies points of contention that affect scaling



Select the required dumps  Heap Dump - Picture of in-memory objects on the Java heap, used for memory analysis.  System Dump - Also known as core dump. Involves dumping the entire address space and as such can be very large. Java Dump - Also known as thread dump or Java core. Used for viewing the thread activity inside the JVM at a given tin	Dump Options	
Heap Dump - Picture of in-memory objects on the Java heap, used for memory analysis. System Dump - Also known as core dump. Involves dumping the entire address space and as such can be very large. Java Dump - Also known as thread dump or Java core. Used for viewing the thread activity inside the JVM at a given tin	Select the required dumps	
System Dump - Also known as core dump. Involves dumping the entire address space and as such can be very large. Java Dump - Also known as thread dump or Java core. Used for viewing the thread activity inside the JVM at a given tin	Heap Dump - Picture of in-memory objects on the Jav	va heap, used for memory analysis.
I Java Dump - Also known as thread dump or Java core. Used for viewing the thread activity inside the JVM at a given tin	System Dump - Also known as core dump. Involves d	dumping the entire address space and as such can be very large.
	Java Dump - Also known as thread dump or Java core.	. Used for viewing the thread activity inside the JVM at a given time
	Java Jump - Also known as thread dump or Java core.	b. Used for viewing the thread activity inside the JVM at a given time
	m Java Dump - Also known as thread dump or Java core.	. Used for viewing the thread activity inside the JVM at a given time
	[[] Java Uump - Also known as thread dump or Java core.	. Used for viewing the thread activity inside the JVM at a given time
	🛄 Java Dump - Also known as thread dump or Java core.	. Used for viewing the thread activity inside the JVM at a given time
	Java Dump - Also known as thread dump or Java core.	. Used for viewing the thread activity inside the JVM at a given time

#### Live runtime control 🐔

Trigger dumps Enable additional data collection



# Installing into CICS Explorer



#### Help -> Install New Software -> Add

http://public.dhe.ibm.com/software/websphere/runtimes/tools/healthcenter

💮 Install		
Available Software Check the items that you wish to install.		
Work with: Health Center - http://public.dhe.ibm.com Find	/software/websphere/runtimes/tools/healthcenter/ 👻 🚺	Add
type filter text		
Name	Version	
IBM Runtime Monitoring and Diagnostic Too     Image: Applied to the select All Select All Deselect All 1 item selected	ls 2.2.0.201401101227	
Details		12
Show only the latest versions of available software	Hide items that are already installed	
Group items by category	What is <u>already installed</u> ?	
Show only software applicable to target environment		
Contact all update sites during install to find required s	oftware	Cancel
	A DOCK INEXL > FINISN	Cancel



# **Deployment Modes – Point to Point**

• Using JMX connection directly to application



### Setup



Define additional iiop port if firewall rules prevent dynamic allocation

#### JVM profile settings

-Xhealthcenter:port=8115 -Dcom.ibm.java.diagnostics.healthcenter.agent.iiop.port=8116 -

#### • Output - stderr

Dec 31, 2013 1:16:48 PM com.ibm.java.diagnostics.healthcenter.agent.mbean.HCLaunchMBean <init> INFO: Agent version "2.2.0.20131003" Dec 31, 2013 1:16:48 PM com.ibm.java.diagnostics.healthcenter.agent.mbean.HCLaunchMBean startMBeanServer INFO: IIOP will be listening on port 8116 Dec 31, 2013 1:16:49 PM com.ibm.java.diagnostics.healthcenter.agent.mbean.HCLaunchMBean startAgent INFO: Health Center agent started on port 8115.

#### Check the Health Center agent is listening using TSO netstat command:-

NETSTAT (PORT 8115 MVS TCP/IP NETSTAT CS V2R1 TCPIP Name: TCPIP 13:55:46 User Id Conn State CICS2A20 000DE8A6 Listen Local Socket: ::..8115 Foreign Socket: ::..0

# **Deployment Modes - Headless**



- Utilizes zFS file system to store hcd files until client connects
- Hcd files read directly by Eclipse client



### **Setup - headless**



#### JVM profile settings



-Dcom.ibm.java.diagnostics.healthc.nter.data.collection.level=headless -Dcom.ibm.java.diagnostics.healtncenter.headless.output.directory =/cicsjava/logs/&APPLID;/&JVMSERVER;

- -Dcom.ibm.java.diagnostics.healthcenter.headless.files.max.size=10000000
- -Dcom.ibm.java.diagnostics.healthcenter.headless.run.number.of.runs=2
- -Dcom.ibm.java.diagnostics.healthcenter.headless.files.to.keep=10
- -Dcom.ibm.java.diagnostics.healthcenter.headless.run.duration=10,

Time in

### **Deployment Modes – Late attach**



5

• No data collection until client connects





### **Setup – Late attach**



#### • JVM profile settings

- -Xhealthcenter:port=8115
- -Dcom.ibm.java.diagnostics.healthcenter.agent.iiop.port=8116
- -Dcom.ibm.java.diagnostics.healthcenter.data.collection.level=off



# System environment



ile Edit Navigate Sea	rch Droject	Data Pun Monitor	d IVM Window	Hale		3	
ne cuit Navigate sea						to and the set	
			1 mm ~0 10 6				
			Quick Access	🔡 🔛 CI	CS SM 🛛 🖧 z/OS 🧕 Healt	h Center Environ	nen
🕈 Status 🕱 🗔 Conn	- 0	Sconfiguration	🔲 System prope	rties 🔲 Environment v	variables	-	
		Property	Value				*
CPU	(i)	Java parameters					
<b>A</b> a	0	773 \	-Xoptionsfile=/j	java/java71_bit64_GA/J7.	1_64/lib/s390x/compressedr	efs/options.de	
Casses	0		-Xlockword:mo	de=default,noLockword	=java/lang/String,noLockwo	rd=java/util/	
Environment			-Xjcl:jclse7b_27				
Garbage Collection	0		-Dcom.ibm.oti.	vm.bootstrap.library.patl	h=/java/java71_bit64_GA/J7.:	L_64/lib/s390x	_
140	0		-Dsun.boot.libra	ary.path=/java/java/1_br	t64_GA/J/.1_64/lib/s390x/coi	mpressedrets:/	-
	U		-Djava.library.pa	atn=/java/java/1_bito4_0 iava/iava71_bit64_GA/I7	1 64	ssearers:/java/	-
🔀 Locking			-Diava.ext.dirs=	/iava/iava71_bit64_GA/I7	1_04 / 1_64/lib/ext		Ξ
hethod Trace	0		-Duser.dir=/u/w	vakelin/cicsjava/logs/IYK	ZZ32E/OSGUVM1		
Mative Memory	0		-Djava.runtime.	version=pmz6470_27-20	131115_04		-
			-Xjit:noResumal	bleTrapHandler			
10 Profiling	۵		-XXnosuballoc3	2bitmem			
₩ <u>Threads</u>	0		-Xhealthcenter:	port=8115			
WebCohere Dost Timo	0		-Dcom.ibm.java	a.diagnostics.healthcente	er.agent.iiop.port=8116		-
09% webshire web inne	U		-Xshareclasses:r	name=cics.IYK2Z32E			-
			-Xms30M				-
🖪 Analysis and Rec 🔀	- 8		-Xmso128K				
			-Dcom.ibm.cics	.ivmserver.configroot=/	u/wakelin/cicsiava/z32e		۰.
No configuration pro	hleme			,,,			
vere detected.	Junetitis	🦠 Java runtime enviro	nment 🖾	- 8	🖨 System 🖾	-	
				Value		Value	
		Process id		67174691	Number of available pro	cessors 4	
		Health Center Agent	library build date	Oct 3 2013 12:03:18			
		Health Center Agent	version	2.2.0.20131003			
				1 2			
	12					1	

### **GC** Analysis



											Summary	€ð Pa	ause time	s	Analysis	and Recor	m 🛛		
Health Cer File Edit M T + H M & Heap and	nter Garbage C Vavigate Sea C C C d pause times 	iollection - IBM C rch Project D ]   🗠 🚉   C S (after collection	CICS Explore Data Run D CICS Explore Quick A Quick A	er - C:\Users\IB Monitored JV D D D D C ccess	M_ADMIN\D M Window B & & & E   E   Summary	ocuments\W Help the second second the second	orkspaces\HC_c + ?? + ? 2/OS ect all	demo	nvironm	op th ent ( ac ent , pr es frr ali	<ol> <li>The me otimal.</li> <li>The nu</li> <li>The nu</li> <li>The nu comp cap usage w ctivity or fra roblem. If the equency of ready using the nurservision</li> </ol>	mber o mber o vared to vas 6%, gmenta ne work collecti a gene ize.	upancy in f collectio the midd which sug ation rathe load is no ions may l rrational p	the in ons in lle thi ggest er tha ot con be no oolicy,	nursery is icreased b ird. Howe is that an i an a memo istant ther othing to v , you may	68% which y 76% in th ver, the ch ncrease in ory leak m the chan worry abou wish to co	t is close the last the ange in the applicat ay be the ge in the st. As yo onsider t	e to hird of the tion e e u are uning	
30.0- 원 20.0- 빗 10.0- 0.0-	Heap size Pause tim 14:26:40 Clo	e		0.0 0.0 0.0 time 0.0 0.0 0.0 0	System (for Minor colle Minor colle Concurrent Largest mer GC Mode Number of Global colle Proportion Proportion Global colle Global colle	ced) garbage ctions - Mear ctions - Num collection co mory request collections tr ctions - Num of time spent of time spent ctions - Mea ctions - Mea	collection cour n garbage collec- liber of collectio punt iggered by alloc ober of collectio cunpaused (%) t in Garbage Colle n garbage colle n interval betwe	nt tion pause ns cation failure ns lection pauses ( ction pause een collections	0 5.31 1 0 253 Defa 731 4 100. %) 0.00 36.4 1282	ms KB ault (gencc 0% 29% ms 2391 ms	5m)								*
JVM Serv CNX02111 C Region IVK2Z32E	ers 🙁 ontext: IYK2Z32 Name OSGIJVM1	2E. Resource: JVN Max Threads 200	ASERV. 1 rec Threads 21	ords collected Use Count 124804	at 31-Dec-20 Init Heap 31457280	13 15:42:12 Heap 7362824	GC Min Heap 5794120	me: OSGUVM1 GC Events 730	Minor)	K ⊽ ⊏ GC Event 4	ts (Ma								
•				m					* 10.50		•								



### **Garbage collection**





# **Garbage Collection – JVM Server**

- JVM Server- Garbage collection
  - Performed in-line using standard JVM facilities
  - Defaults to -Xgcpolicy:gencon
    - GC triggered by object allocation failure
    - All work in JVM stopped whilst collection occurs
    - GC CPU split between T8 TCB and GC helper threads
- Generational Concurrent
  - Heap is split into new and old segments
  - Long lived objects are promoted to the old space (tenured)
  - Short-lived objects are garbage collected quickly in the new space (nursery)



# **JVM Heap and Garbage collection**

- JVMServer
  - Minor collections for short lived/small objects
  - Major(global) collections for long lived objects
- Tuning strategy:
  - Start JVM
    - Default is gencon with 256MB max heap
    - Run Java workload
    - Analyse Heap usage
  - Set Max Heap to Peak Heap + ~10%
    - Check occupancy does not reach > 50% MaxHeap
    - Check GC time < 2% of JVM time
    - Check time between GCs > 1s
    - Check GC times < 10ms</li>

# **JVM Tuning Options**



- Heap
  - Reduce/increase max heap if peak does/does not reach max (Xmx)
  - To fix size of nursery and tenured areas
    - » Pre-allocate heap to required size: -Xmx=-Xms
    - » Fix size of nursery area: -Xmnx=-Xmns
    - » Fix size of tenured area: -Xmox=-Xmos
- Compressed references
  - Xcompressedrefs
  - Reduces heap usage and improves GC efficiency
  - Works for (smaller) heaps up to 25GB
  - Set as default in Java V7.1
- Shared class cache
  - -Xshareclasses:name=cics.&APPLID;
  - Enables Java6 shared class cache
  - Improves startup time, class loading, and JITing (AOT)
  - Ensure its not full, default is 16MB
    - » -Xscmx128M

### Heap management views



Monitored System->Garbage Collection and allocation data collection

- Object allocation data
  - Use this view to identify code that is allocating large objects
  - Set low and high thresholds using Expensive to collect.. Not for production
- Samples by object
  - Identify code that is allocating large numbers of objects outside of the thread local heap.
  - Enable collection of call stacks to show call hierarchy

ciuss no	inco.				[rippi] [cit
Count	%	%	Total size (KB)	Allocated Object	
4	23.5		0.13	java/lang/StringBuffer	
3	17.6		0.12	com/ibm/rmi/util/buffer/ByteBuffer\$Position	
3	17.6	•	0.12	java/util/HashMap\$Entry	
1	5.88	f	0.1	com/ibm/ws/genericbnf/internal/HeaderElement	
1	5.88	i	0.023	com/ibm/rmi/iiop/CDRReader\$PendingReadMarker	
1	5.88	í	0.07	java/util/HashMap	
1	5.88	i	0.031	java/lang/StringBuilder	
1	5.88	ſ	0.023	java/util/HashMap\$KeySet	
1	5.88	i	0.055	java/lang/StackTraceElement	
1	5.88	1	0.19	com/ibm/rmi/iiop/RequestMessage	
mary  •	Call I	nierarchy	없고. Timeline		-
Calls to Calls to Com. Com. Com. Com. Com. Com. Com. Com	allocat ibm.ws m.ibm. com.ib	ion of clas http.chan ws.http.di om.cics.wlj	ss java/lang/StringBi nel.internal.HttpReq spatcher.internal.cha p.impl.CICSHttpRunr server internal CICST	uffer uestMessageImpl.getRequestURL (HttpRequestMessageImpl.java:686) (100%) nnel.HttpRequestImpl.getURL (HttpRequestImpl.java:143) (100%) nable.getURL (CICSHttpRunnable.java:419) (100%) breadExecutorSDefaultThreadEactors3 run (CICSThreadExecutor iava:571) (100%)	

# Profiling



- Method level profiling of the applications running within the JVM using JIT sampler data filtered by class or package name.
- The Method profile view shows sample counts for specific methods.
- Self is when the method is at the top of a call stack and tree is when a method appears in a call stack.
- Invocation and Called method views allow you to analyze the call path of each profiled method.
   Health Center Profiling IBM CICS Explorer C/Users/UBM\_ADMIN/Documents/Workspaces/HC\_demo

Health Center Profiling -	IBM CICS Explorer	- C:\Users\IBM_		Docume	nts\Wor	kspaces	\HC_demo	9
ile Edit Navigate Sea	rch Project Dat	a Run Moni	ored JVN	/ Wind	dow H	elp		
9 • 🗑 🕅 🕹 🖉 🕷	0 🛛 🔛 🖉	🖸 🖷 🗎 🛓			5 200	2 Q	• 🛷 • 🖢 • 🕸 • 🌾 • • • • 🖉	
Health Center Garbage	Collection 🙃 Hea	Ith Center Locki	na 📖	Health C	enter N	ative Me	mory 🚯 Health Center Profiling 🕅 Health Center	r Thre
		Contraction of the	J				. [	
🐮 Status 🔀 🗐 🛄 Connec	tion 🗖 🗖	Method g	profile S	3				8
		Filter meth	ods: co	m.ibm.c	ics.test		Apply	Clea
CPU	(i)		003. [00		-concar			0.000
G Classes	(?)	Samples	Self	Self	Tre	Tree	Method	
Environment	0	44	8.98		23.1		com.ibm.cics.test.JavaLinker.main(java.lang.String[]	)
Garbage Collection	0	0	0.0		8.57		com.ibm.cics.test.JavaLinker.printMsg(com.ibm.cic	s.serv
2 1/0	(?)	3	0.61		8.37		com.ibm.cics.test.JavaLinker.formatTime()	
B Locking	0	0	0.0		0.62		com.ibm.cics.test.JavaLinker.readQ(java.lang.string,	1
S Method Trace	1							
INative Memory	(i)							
10 Profiling	0							
₩ <u>Threads</u>	0							
🔊 WebSphere Real Time	(?)		1	m	-	-		
👖 Analysis and Recomme	23 - 8	Sinvocatio	n 🖾	🗞 Ca	illed me	:h 🤇	👌 Timeline 🛛 📰 Method trace 🖓 Samples ove	-
				-				
S Execution time was r	elatively	Methods that	t call Jav	aLinker.r	main()			
venly balanced between r	a 🚺 Java	Linker.m	iain					
bvious candidates for opt	⊿ 🛈	Generate	dMetho	dAccess	or6.invo	ke (100%)		
vere round.		4	Deleg	lethod in	thodAc	cessorin 10%)	npilinvoke (100%)	
			⊿ (	Wrapp	er.call_r	nain (10	0%)	
				🔞 Wr	apper.c	llosgic	lass (100%)	
	*			4 🔞	Wrapp	er.invok	eJvmServerOSGiClass (100%)	0
							1	2.

# **Thread analysis**



ile Edit Navigate Sea	rch Project Data	a Run Monitored JVM Window	Help	0		
3 • R R 🕹 🛛 🖲	) 🖂 🚼 🖌 😡	🖸 🖷 🗊 🖸 💼 👒 🐻 🂐	3 🗠 💊 🔸 🔗	• 🛛 • 🕅 •	****	* 8
] Health Center Garbage (	Collection 🔒 Hea	lth Center Locking 🛛 🛲 Health Cente	r Native Memory	🖔 Health Center	Profiling 🔛 H	lealth Center Thread
🖹 Status 🕱 🗔 Connec	tion 🗖 🗖	🔟 Current threads 🖾	- 0	₩ Number of	threads 🖾	
CPU .	Ð	Thread name filter: *.TASK*	Apply Clear		Number	of threads
Classes	(?)	Thread name	Thread	50.0	10 -0 4	A
Environment	0	LINKER.TASK53520.CSMI	TIMED	Y	WW VV	www.
Garbage Collection	0	LINKER.TASK53521.CSMI	TIMED_	40.0		
N 1/0	0	LINKER.TASK53522.CSMI	TIMED_	£		
A Lasting	() ()	LINKER.TASK53523.CSMI	TIMED_	) 30.0		
	0	LINKER.TASK53524.CSMI	TIMED_	le la		
Wethod Trace	0	LINKER, TASK53525, CSMI	TIMED_	= 20.0		
Mative Memory	(L)			40.0		
10 Profiling	8			10.0		
₩ <u>Threads</u>	$\odot$			0.0		
WebSphere Real Time	(?)			0.0	3:36:40 13:45:	00 13:53:20
		*	E	1.00	Clock tir	ne (time)
Analysis and Recomme	x - D					
Vour application has	47 threads					
• rosi oppression nos	Tr circous					
		₩ Thread stack 🔀				- E
		▲ 🖉 LINKER.TASK53525.CSMI				
		a 🔟 java.lang.Thread.sleep	(Native Method)	5		
		⊿ Ø java.lang.Thread.s ▷ Ø com.ibm.cics.t	leep(Thread.java:977 test.JavaLinker.main(	) JavaLinker.java:1	68)	
	, 1991				2054531	

24

# Lock analysis



- The Locking perspective profiles Java lock (aka monitors in Java) usage and helps identify points of contention in the application or Java<sup>™</sup> runtime environment that prevent the application from scaling
- Useful metrics are:

96795

35899

----

0

0

0

0

60442

0

0

0

- % miss:- percentage of non-recursive requests that had to wait for the lock
- Slow:- number of times a requests had to wait



12292

19090

· · · · · ·

[0000004884A9F8D0] com/ibm/rmi/iiop/WorkQueue@000000488894/

[00000048FDE4BD68] [I@0000004889CF3C68 (Object)

% util:- percentage of time this lock was held during the measurement interval

## **Customising data collection**



If an application generates more data than Health Center can process, it is possible that Health Center might lose some data. If data loss occurs, you see a message about dropped data points in the agent connection view.

You can reduce the likelihood of losing data by turning off the collection of data from areas that you are not interested in.

To access these options, use **Monitored JVM > Data Collection Settings.** 



### **Gathering trace**



- Various JVM diagnostics actions can be driven from the Health Center client by using Monitored JVM
- -> Request a dump to produce either Heap, System or Javacore dumps to a file
- -> Garbage Collection to select verbosegc data be written to a file
- -> Trace settings to enable and disable Java method tracing

🕄 Dump Wizard	
Dump Options Select the required dumps	
<ul> <li>Heap Dump - Picture of in-memory objects on the Java heap, use</li> <li>System Dump - Also known as core dump. Involves dumping the</li> <li>Java Dump - Also known as thread dump or Java core. Used for v</li> </ul>	GC Data Collection Settings     GC Data Collection of GC and Allocation Data     Use these settings to control collection of verbose GC data and data for the analysis of object allocations requests
	Write verbose GC data to file  Enable collection of call stacks for sampled object allocation events  Enable collection of object allocation events within thresholds Low threshold (bytes) High threshold (bytes) Undo
	Maximum number of stack entries to collect per event
	Finish Cancel

### Performance





Measured using WebSphere App Server and the DayTrader benchmark with 50 clients Running WAS 8.5.5, IBM Java 7 SR5, AIX 7.1, POWER7

Throughput determined by number of completed transactions on 4 saturated CPUs

- Validation in CICS has shown no measureable overhead in late attach mode
- 1% CPU overhead when client connected



# Demos



### References

- IBM Monitoring and Diagnostics Health Center: <u>https://www.ibm.com/developerworks/java/jdk/tools/healthcenter/</u>
- IBM Support Assistant User's Guide
   http://www.ibm.com/support/knowledgecenter/SSLLVC\_4.1.0/com.ibm.java.diagnostics.healthcenter.
   doc/homepage/plugin-homepage-hc.html
- Customizing perspectives in CICS Explorer CICSdev article Extending CICS
   Explorer: Creating custom perspectives
- Setting up Health Center in CICS Explorer CICSdev article Integrating IBM Health Center and CICS Explorer
- Analyzing JVM server performance IBM Redpaper IBM CICS Performance Series: CICS TS V4.2 and Java Performance – REDP4850



Complete your session evaluations online at www.SHARE.org/Seattle-Eval