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

Phil_Wakelin@uk.ibm.com

CICS Strategy & Design, IBM Hursley UK

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.

Agenda

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

WebShere Liberty Profile
wasdev.net

IBM Developer Kits for Java
ibm.biz/javasdk

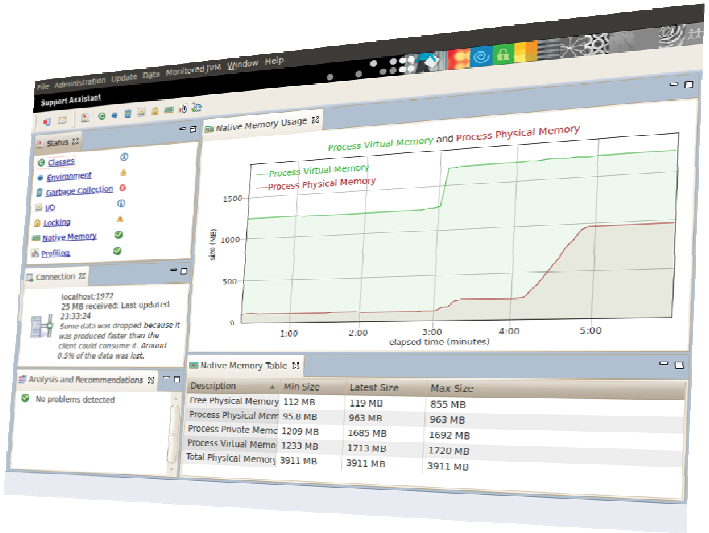
IBM Monitoring and Diagnostics: Health Center

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

- Pre-installed in IBM SDKs for Java 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



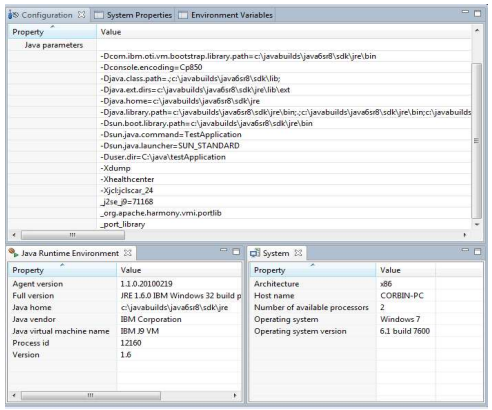
Visualization provided via Eclipse Client UI

- Available from Eclipse Marketplace
- Available from IBM Support Assistant
- Eclipse p2 install

Data access API provided

- Allows creation of custom monitoring tools
- Full API Javadoc available

IBM Monitoring and Diagnostics: Health Center



Environment

Hardware and Operating System Configuration

Process environment and configuration

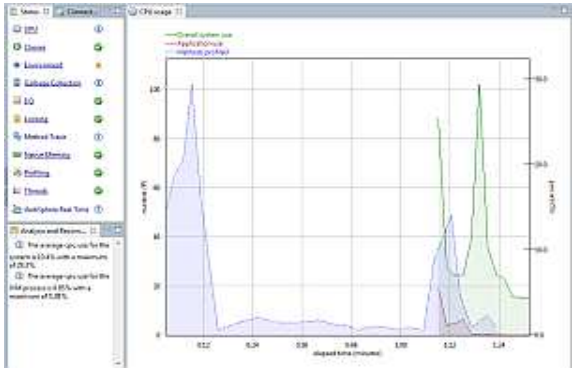
Highlights incorrect or non-standard configurations

Memory Utilization

Detect native memory leaks in application

Determine if external forces are using more memory

View components using the most native memory



CPU Utilization

Visualizes process CPU usage over time

Visualizes system CPU usage over time

IBM Monitoring and Diagnostics: Health Center

Garbage Collection

Visualizes heap usage and GC pause times

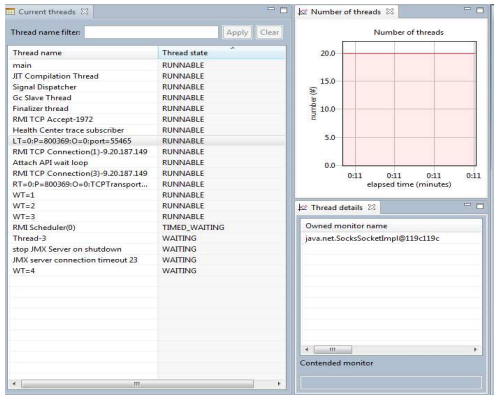
Identifies memory leaks

Suggests command-line and tuning parameters

Object Allocations

Understand types of data being allocated

Determine which code is allocating data



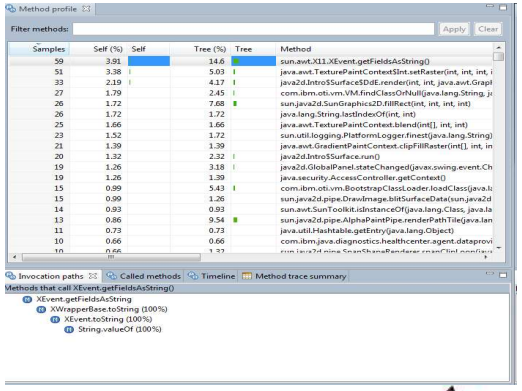
Threads

List of current threads and states

Number of threads over time

See contended monitors

IBM Monitoring and Diagnostics: Health Center



Method Profiling

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

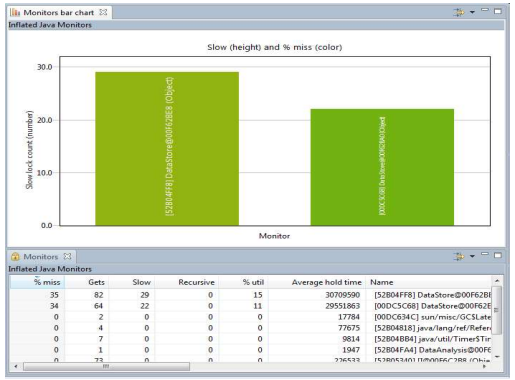
No byte code instrumentation, no recompiling

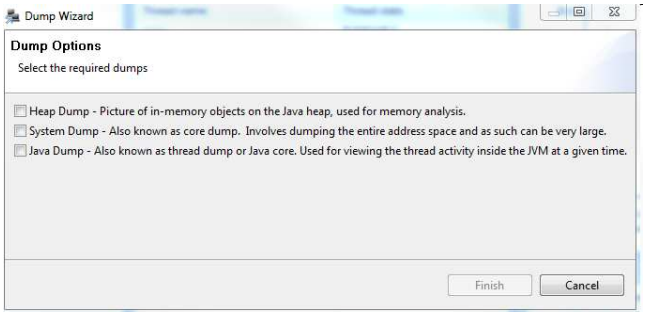
Lock Profiling

Always-on lock monitoring

Allows the usage of all locks to be profiled

Identifies points of contention that affect scaling





Live runtime control

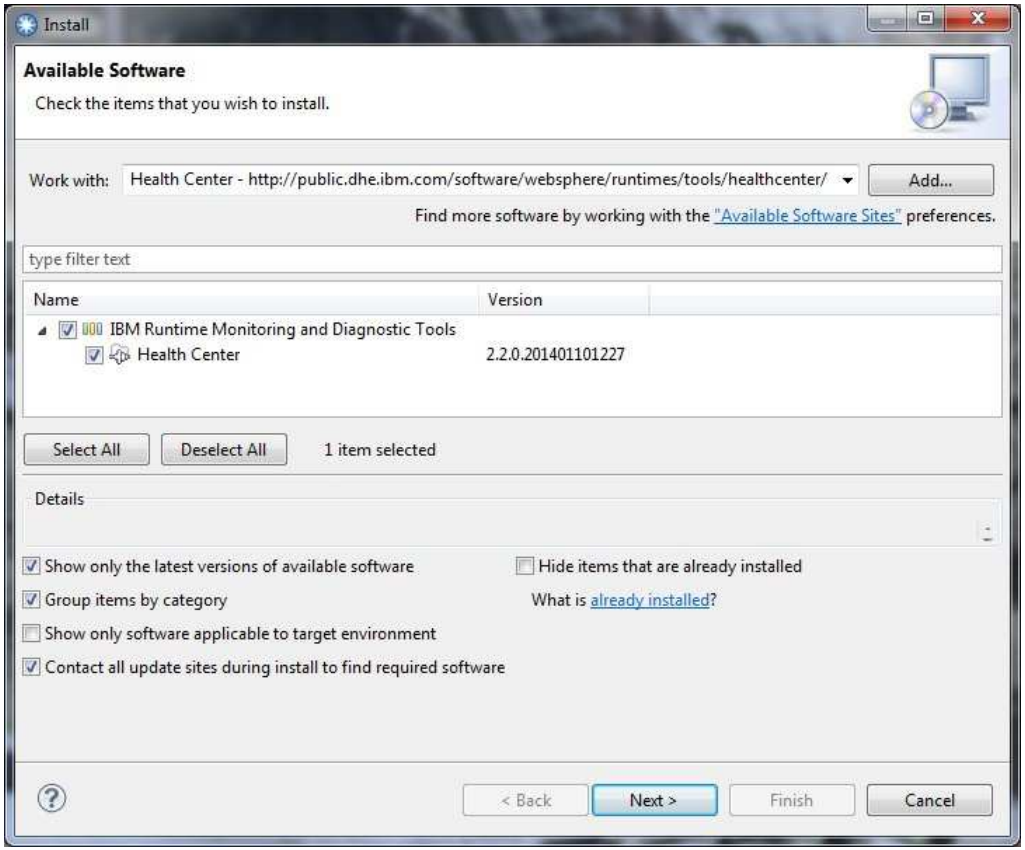
Trigger dumps

Enable additional data collection

Installing into CICS Explorer

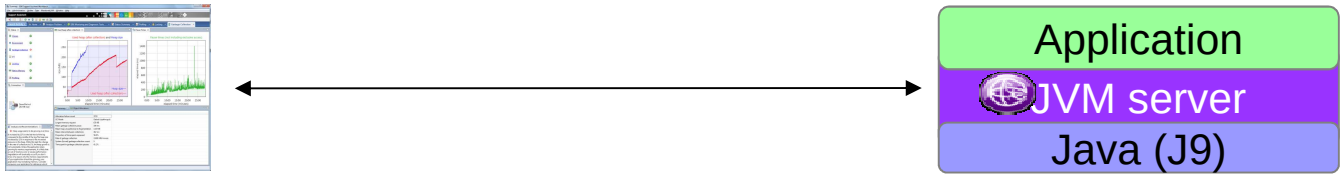
Help -> Install New Software -> Add

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



Deployment Modes – Point to Point

- Using JMX connection directly to application



Eclipse client

JVM server

Setup

- JVM profile settings

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

Define additional iiop port if firewall rules prevent dynamic allocation

- Output - stderr

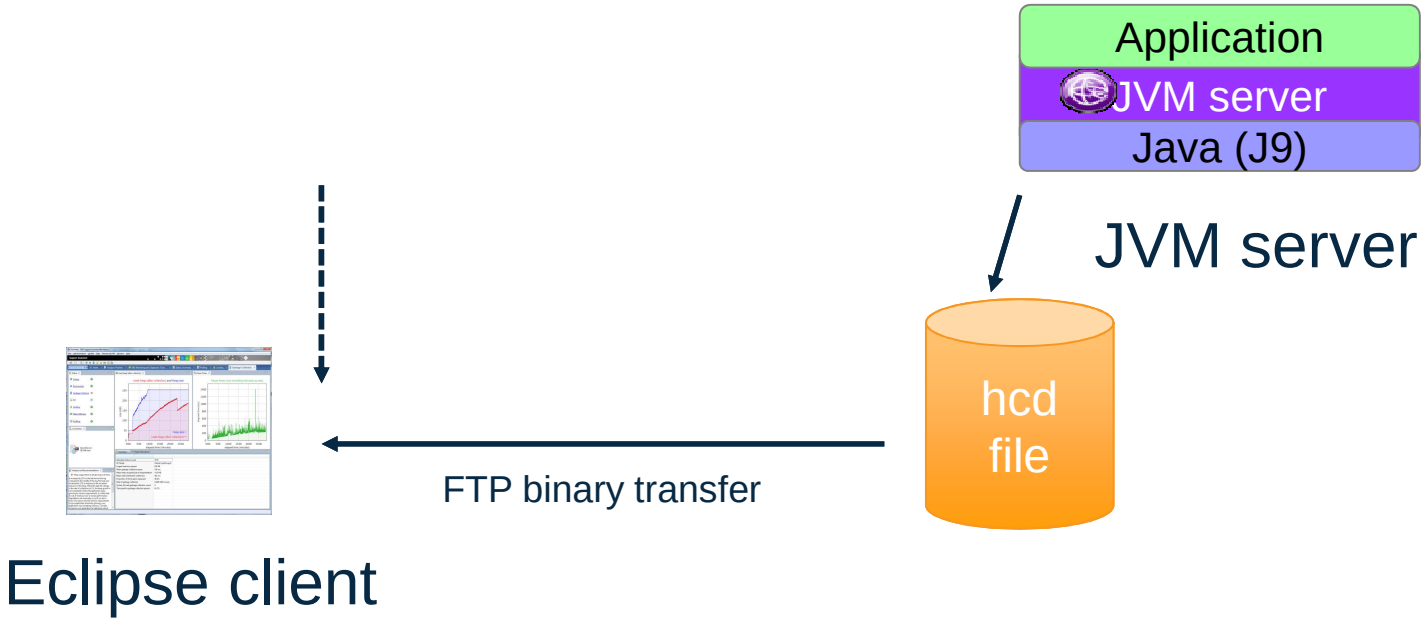
```
Dec 31, 2013 1:16:48 PM
com.ibm.java.diagnostics.healthcenter.agent.mbean.HCLaunchMBean <init>
INFO: Agent version "2.2.0.20131003"
Dec 31, 2013 1:16:48 PM
com.ibm.java.diagnostics.healthcenter.agent.mbean.HCLaunchMBean startMBeanServer
INFO: IIOP will be listening on port 8116
Dec 31, 2013 1:16:49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

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

-Dcom.ibm.java.diagnostics.healthcenter.data.collection.level=headless
-Dcom.ibm.java.diagnostics.healthcenter.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
```

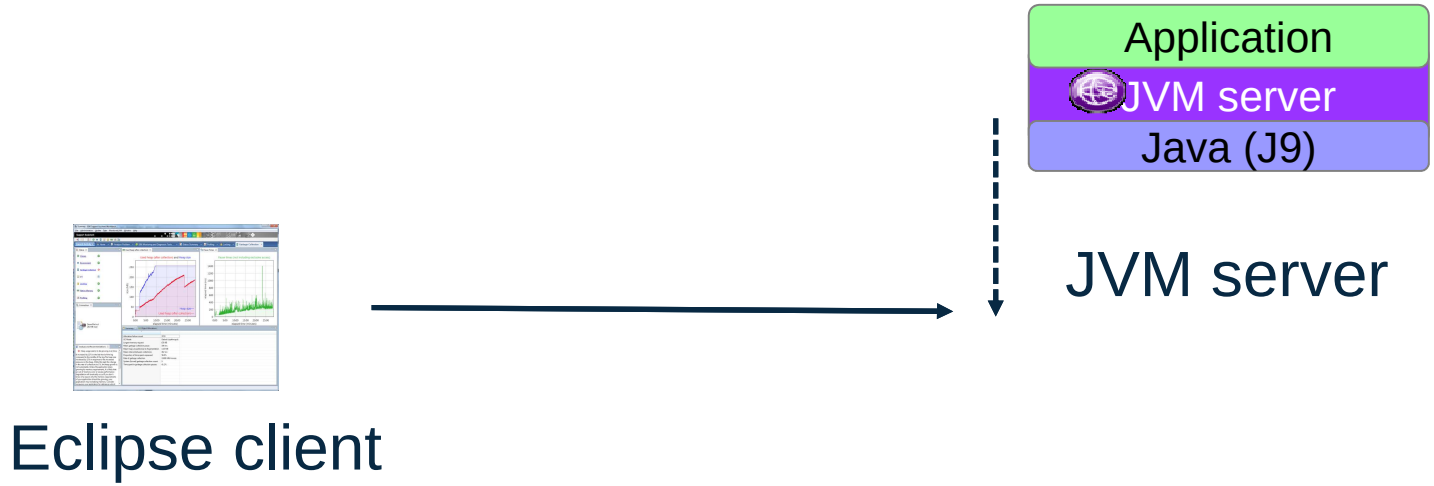
Can use CICS JVM server symbols

Max file size, 2GB default

Time in minutes

Deployment Modes – Late attach

- 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
```

No internal data collection

System environment

Health Center Environment - IBM CICS Explorer - C:\Users\IBM_ADMIN\Documents\Workspaces\HC_demo

File Edit Navigate Search Project Data Run Monitored JVM Window Help

Quick Access | CICS SM | z/OS | Health Center Environment

Status | Conn...

- CPU
- Classes
- Environment
- Garbage Collection
- I/O
- Locking
- Method Trace
- Native Memory
- Profiling
- Threads
- WebSphere Real Time

Analysis and Rec...

No configuration problems were detected.

Configuration | System properties | Environment variables

Property	Value
Java parameters	
	-Xoptionsfile=/java/java71_bit64_GA/J7.1_64/lib/s390x/compressedrefs/options.de...
	-Xlockword:mode=default,noLockword=java/lang/String,noLockword=java/util/...
	-Xjcl:jclse7b_27
	-Dcom.ibm.oti.vm.bootstrap.library.path=/java/java71_bit64_GA/J7.1_64/lib/s390x...
	-Dsun.boot.library.path=/java/java71_bit64_GA/J7.1_64/lib/s390x/compressedrefs:/...
	-Djava.library.path=/java/java71_bit64_GA/J7.1_64/lib/s390x/compressedrefs:/java/...
	-Djava.home=/java/java71_bit64_GA/J7.1_64
	-Djava.ext.dirs=/java/java71_bit64_GA/J7.1_64/lib/ext
	-Duser.dir=/u/wakelin/cicsjava/logs/IYK2Z32E/OSGIJVM1
	-Djava.runtime.version=pmz6470_27-20131115_04
	-Xjit:noResumableTrapHandler
	-XXnosuballoc32bitmem
	-Xhealthcenter:port=8115
	-Dcom.ibm.java.diagnostics.healthcenter.agent.iiop.port=8116
	-Xshareclasses:name=cics.IYK2Z32E
	-Xms30M
	-Xmx256M
	-Xmso128K
	-Dcom.ibm.cics.jvmserver.configroot=/u/wakelin/cicsjava/z32e

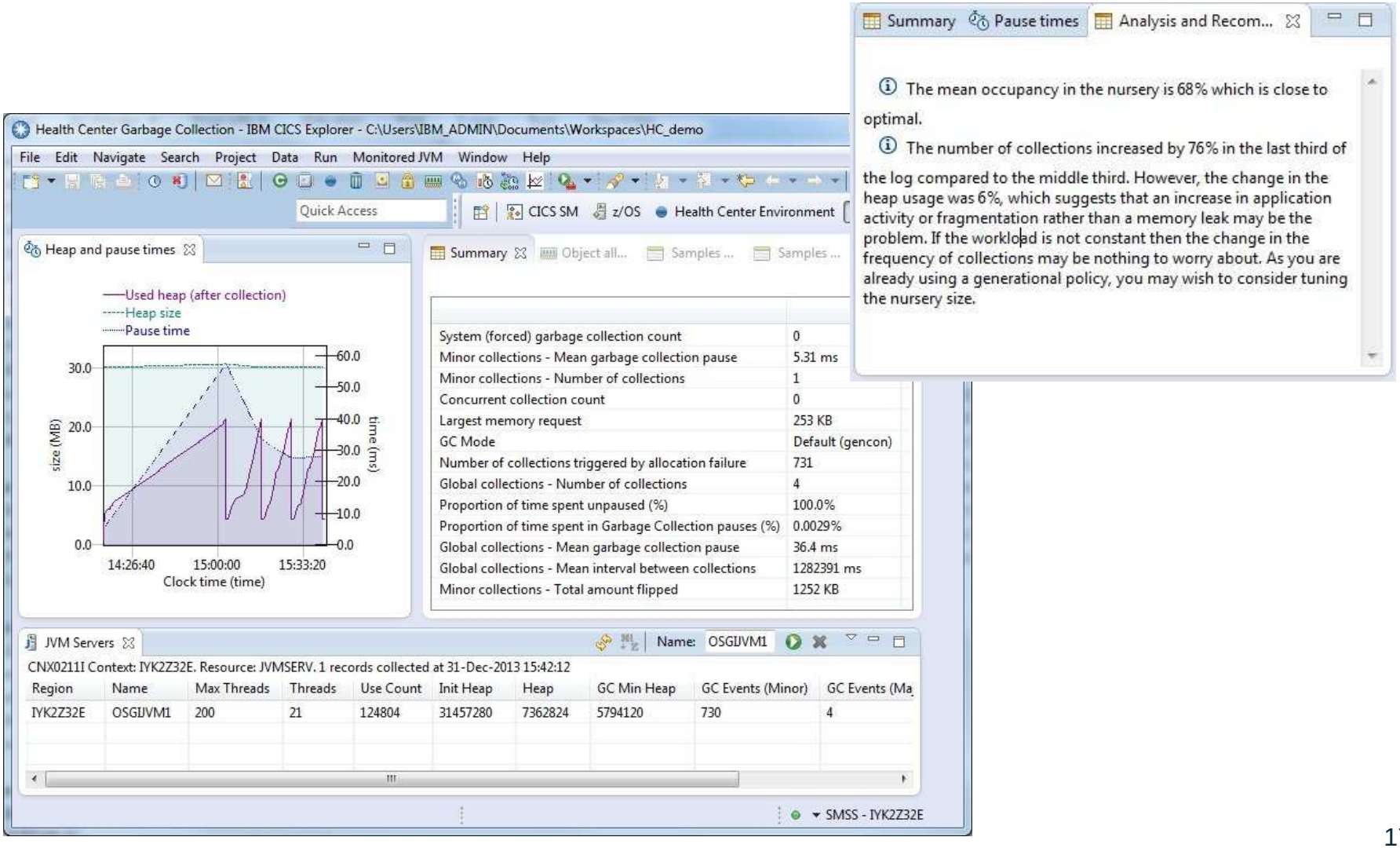
Java runtime environment

	Value
Process id	67174691
Health Center Agent library build date	Oct 3 2013 12:03:18
Health Center Agent version	2.2.0.20131003

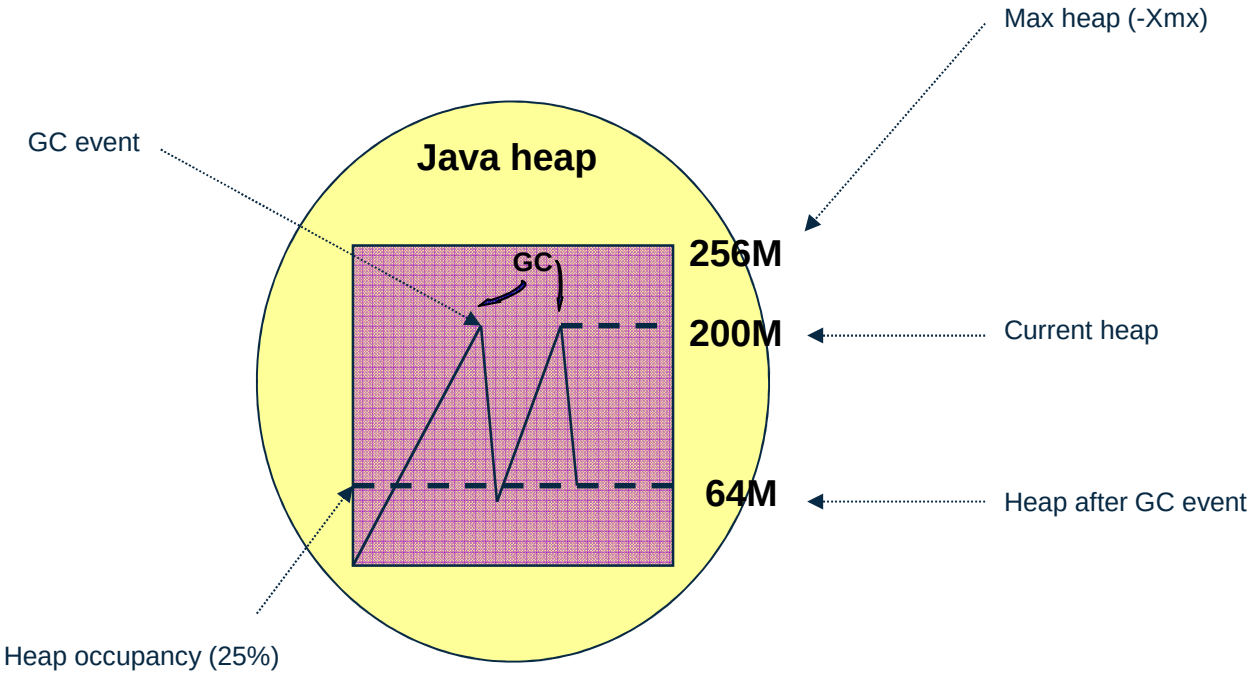
System

	Value
Number of available processors	4

GC Analysis


Summary
Pause times
Analysis and Recom...
The mean occupancy in the nursery is 68% which is close to optimal.
The number of collections increased by 76% in the last third of the log compared to the middle third. However, the change in the heap usage was 6%, which suggests that an increase in application activity or fragmentation rather than a memory leak may be the problem. If the workload is not constant then the change in the frequency of collections may be nothing to worry about. As you are already using a generational policy, you may wish to consider tuning the nursery size.
Health Center Garbage Collection - IBM CICS Explorer - C:\Users\IBM_ADMIN\Documents\Workspaces\HC_demo
File Edit Navigate Search Project Data Run Monitored JVM Window Help
Quick Access
CICS SM
z/OS
Health Center Environment
Heap and pause times
Used heap (after collection)
Heap size
Pause time
size (MB)
time (ms)
Clock time (time)
Summary
Object all...
Samples ...
System (forced) garbage collection count 0
Minor collections - Mean garbage collection pause 5.31 ms
Minor collections - Number of collections 1
Concurrent collection count 0
Largest memory request 253 KB
GC Mode Default (gencon)
Number of collections triggered by allocation failure 731
Global collections - Number of collections 4
Proportion of time spent unpaused (%) 100.0%
Proportion of time spent in Garbage Collection pauses (%) 0.0029%
Global collections - Mean garbage collection pause 36.4 ms
Global collections - Mean interval between collections 1282391 ms
Minor collections - Total amount flipped 1252 KB
JVM Servers
Name: OSGIJVM1
CNX0211I Context: IYK2Z32E. Resource: JVMSERV. 1 records collected at 31-Dec-2013 15:42:12
Region Name Max Threads Threads Use Count Init Heap Heap GC Min Heap GC Events (Minor) GC Events (Ma
IYK2Z32E OSGIJVM1 200 21 124804 31457280 7362824 5794120 730 4
SMSS - IYK2Z32E

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

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

Heap and pause times | Object allocations | Samples by request site | Samples by object

Filter class names: Apply Clear

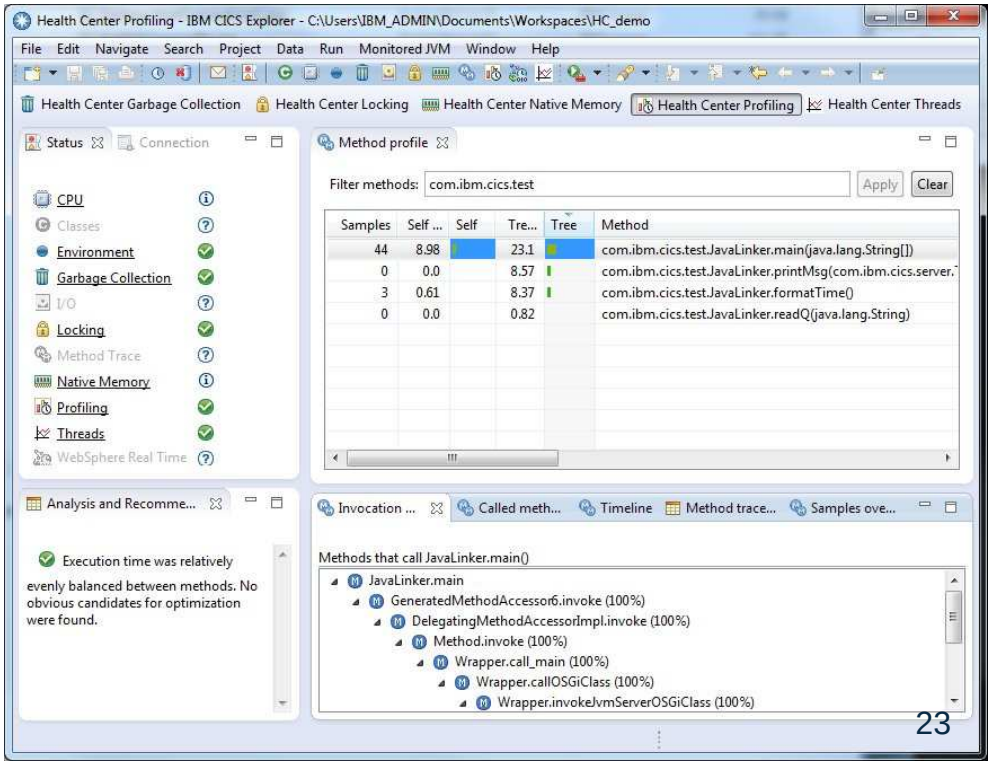
Count	%	%	Total size (KB)	Allocated Object
4	23.5		0.13	java/lang/StringBuffer
3	17.6		0.12	com/ibm/rmi/util/buffer/ByteBuffer$Position
3	17.6		0.12	java/util/HashMap$Entry
1	5.88		0.1	com/ibm/ws/genericbnf/internal/HeaderElement
1	5.88		0.023	com/ibm/rmi/iiop/CDRReader$PendingReadMarker
1	5.88		0.07	java/util/HashMap
1	5.88		0.031	java/lang/StringBuilder
1	5.88		0.023	java/util/HashMap$KeySet
1	5.88		0.055	java/lang/StackTraceElement
1	5.88		0.19	com/ibm/rmi/iiop/RequestMessage

Summary | Call hierarchy | Timeline

- Calls to allocation of class java/lang/StringBuffer
  - com.ibm.ws.http.channel.internal.HttpRequestMessageImpl.getRequestURL (HttpRequestMessageImpl.java:686) (100%)
    - com.ibm.ws.http.dispatcher.internal.channel.HttpRequestImpl.getURL (HttpRequestImpl.java:143) (100%)
      - com.ibm.cics.wlp.impl.CICSHttpRunnable.getURL (CICSHttpRunnable.java:419) (100%)
        - com.ibm.cics.server.internal.CICSThreadExecutor$DefaultThreadFactory$3.run (CICSThreadExecutor.java:571) (100%)
          - com.ibm.cics.server.internal.CICSThreadExecutor$DefaultThreadFactory$3.run (CICSThreadExecutor.java:534) (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.



Thread analysis

Health Center Threads - IBM CICS Explorer - C:\Users\IBM_ADMIN\Documents\Workspaces\HC_demo

File Edit Navigate Search Project Data Run Monitored JVM Window Help

Health Center Garbage Collection | Health Center Locking | Health Center Native Memory | Health Center Profiling | Health Center Threads

Status | Connection

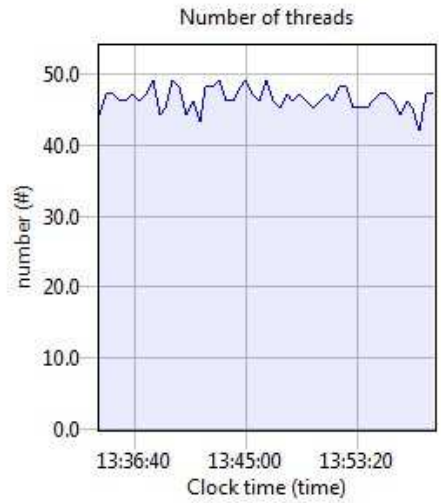
- CPU
- Classes
- Environment
- Garbage Collection
- I/O
- Locking
- Method Trace
- Native Memory
- Profiling
- Threads
- WebSphere Real Time

Current threads

Thread name filter: *.TASK* Apply Clear

Thread name	Thread
LINKER.TASK53520.CSMI	TIMED_
LINKER.TASK53521.CSMI	TIMED_
LINKER.TASK53522.CSMI	TIMED_
LINKER.TASK53523.CSMI	TIMED_
LINKER.TASK53524.CSMI	TIMED_
LINKER.TASK53525.CSMI	TIMED_

Number of threads



Analysis and Recomme...

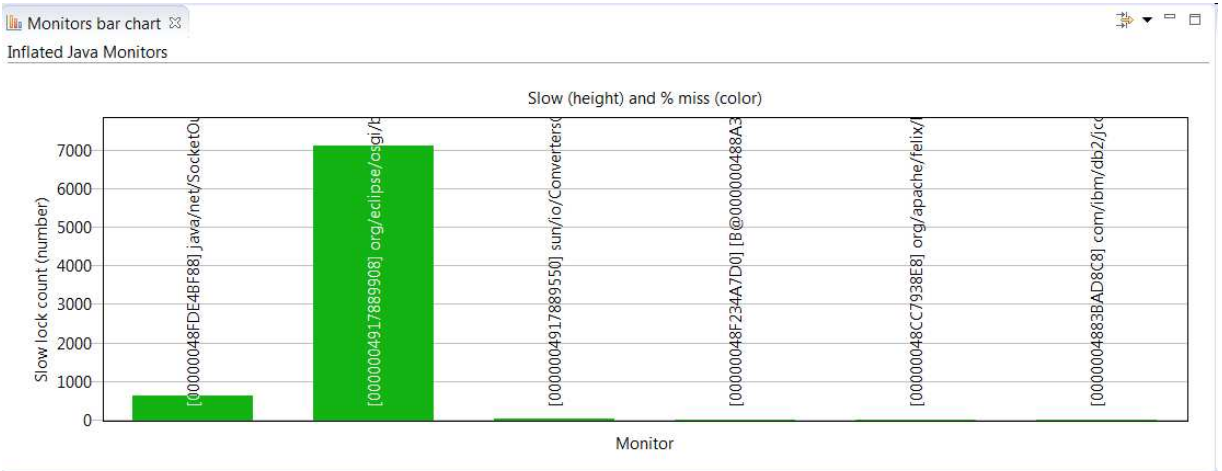
Your application has 47 threads

Thread stack

- LINKER.TASK53525.CSMI
  - java.lang.Thread.sleep(Native Method)
    - java.lang.Thread.sleep(Thread.java:977)
      - com.ibm.cics.test.JavaLinker.main(JavaLinker.java:168)

Lock analysis

- The Locking perspective profiles Java lock (aka monitors in Java) usage and helps identify points of contention in the application or Java™ runtime environment that prevent the application from scaling
- Useful metrics are:
  - % miss:- percentage of non-recursive requests that had to wait for the lock
  - Slow:- number of times a requests had to wait
  - % util:- percentage of time this lock was held during the measurement interval

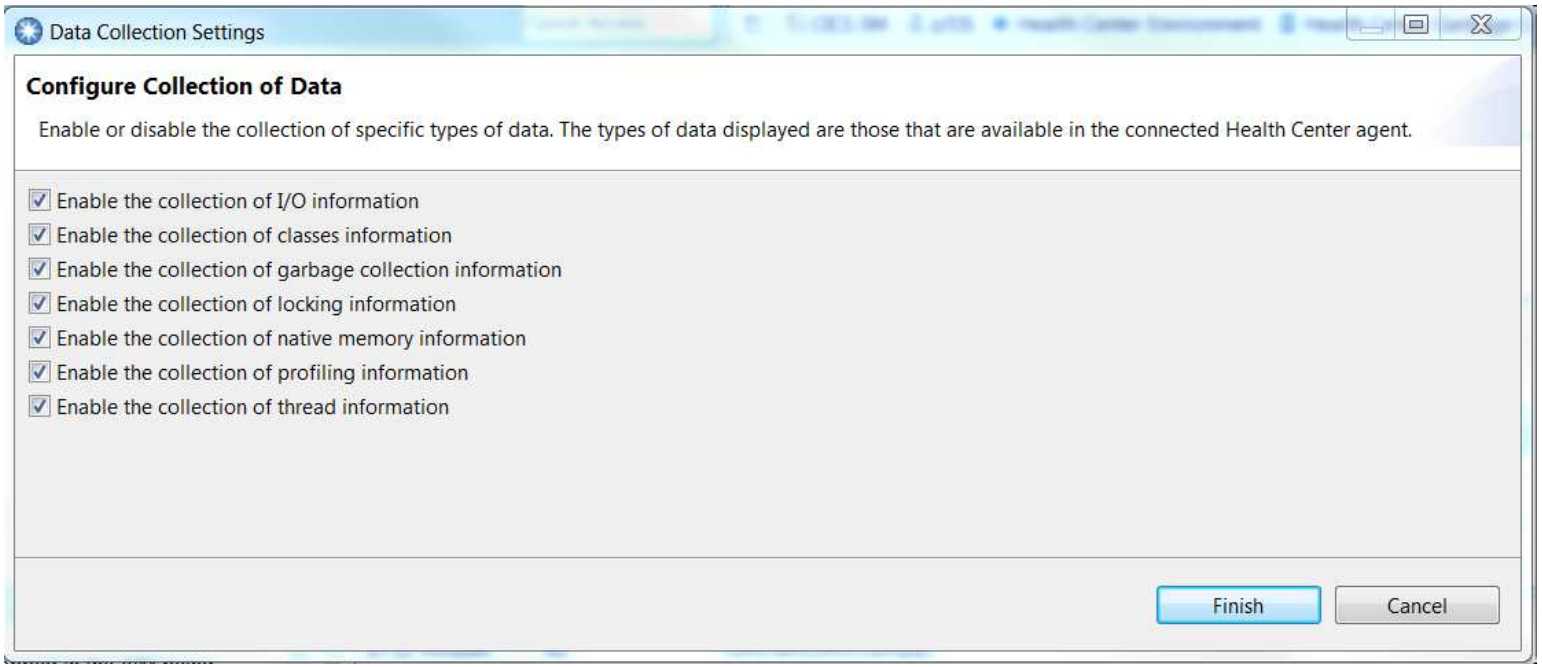


Customising data collection

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.

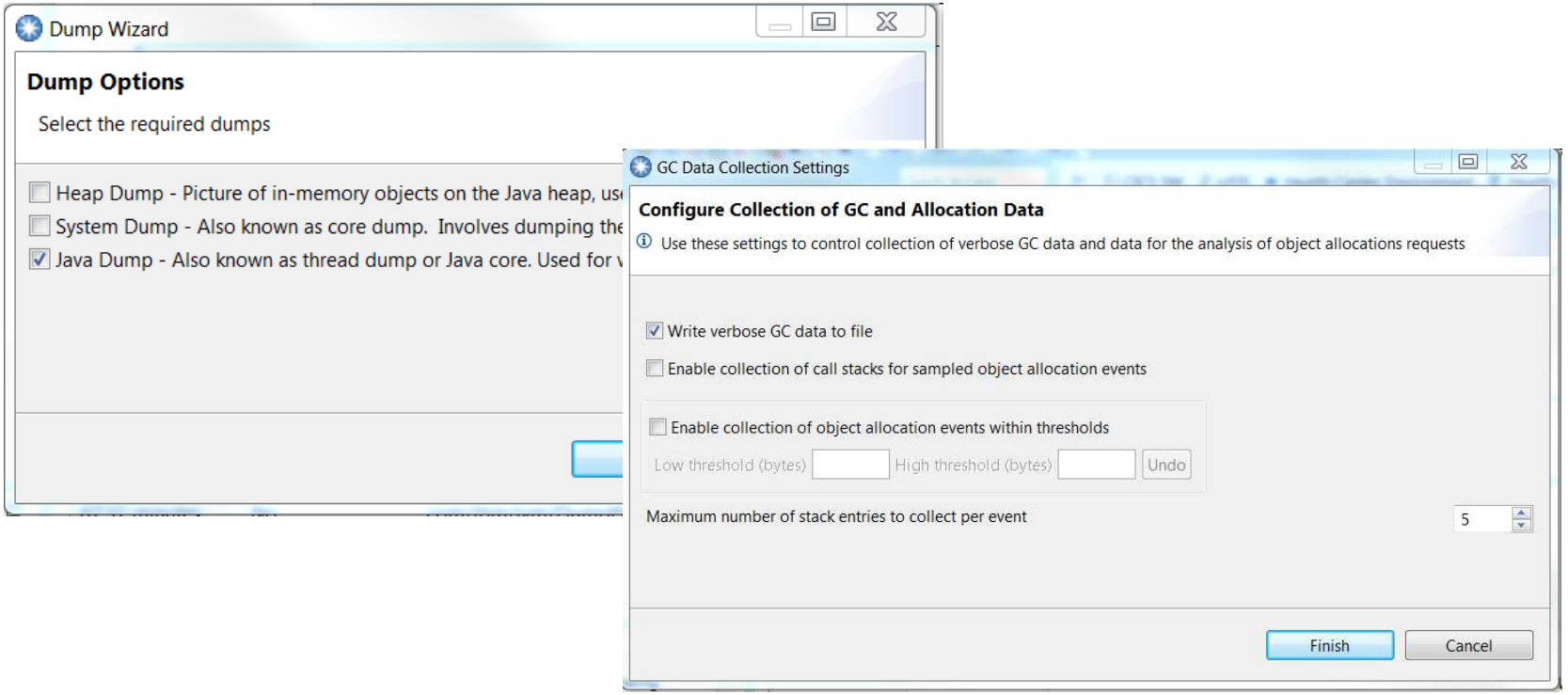
You can reduce the likelihood of losing data by turning off the collection of data from areas that you are not interested in.

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

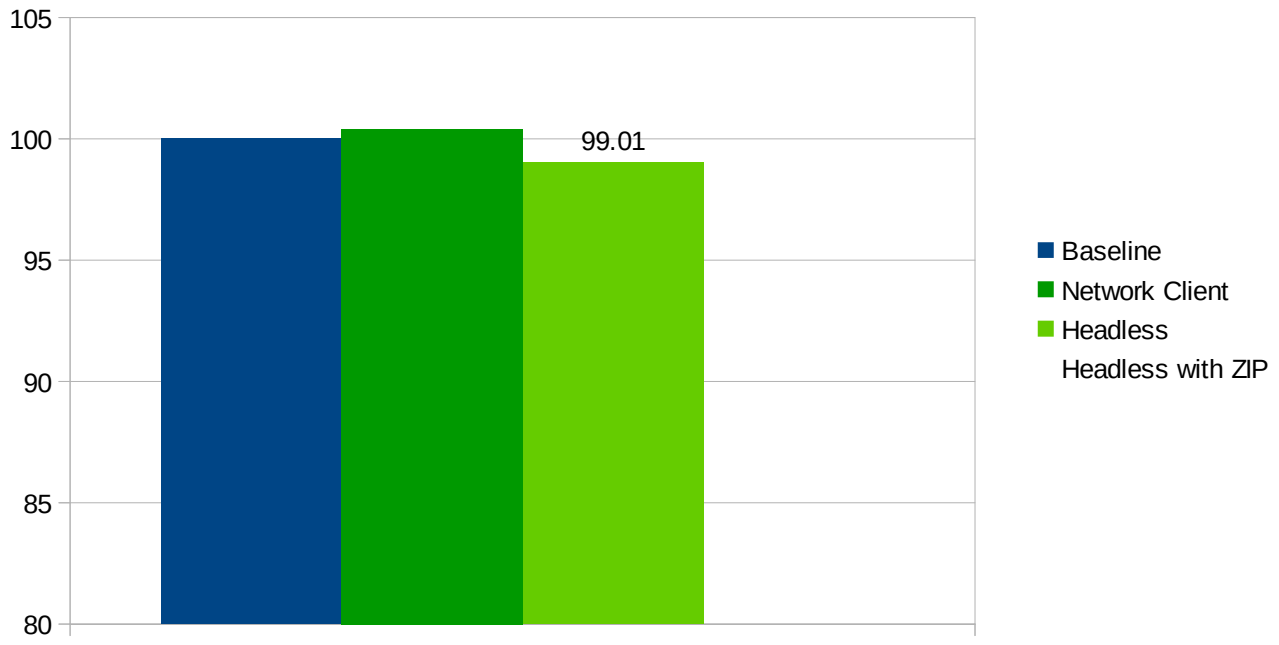


Gathering trace

- 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



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:
  https://www.ibm.com/developerworks/java/jdk/tools/healthcenter/
- 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