Zing Vision

Production performance tuning and diagnosis tool for the Zing runtime

Most Java performance measurement tools have too much overhead to use in production. Zing® Vision is different. It's designed for production use and has negligible performance overhead. Zing Vision gives you insight into your running Java application and Zing, Azul's innovative JVM, without any separate installation or configuration. The tool provides information on Java methods and native functions, thread-level views, lock contention diagnosis, garbage collection behavior and Java heap live object and object type growth rate information.

Java Method and Native Function Profiling

Zing Vision's Java method and native function profiling quickly identifies the 'hot' methods, the ones consuming the most CPU time. Unlike other tools, Zing Vision does not require you to know ahead of time which area of the application might contain the hot methods and does not use byte code instrumentation. You can also quickly drill down to the instruction level to look at details of the method's execution. Of course, Zing Vision also includes a call graph to show the call path(s) to the hot method. The tick profiler is always on. It has almost no overhead and collects data even when you're not looking at the generated metrics.

Timer Tick Profile

Percent	Ticks	Source
2.7%	870	new_stub (vm stub code)
2.4%	768	com.sun.tools.javac.util.Name.fromUtf (codeblob)
2.1%	682	HeapRefBufferList::grab(HeapRefBuffer**) (PC ref)
1.5%	486	com.sun.tools.javac.comp.Resolve.findMethod (codeblob)
1.3%	431	com.sun.tools.javac.comp.Resolve.findType (codeblob)

Zing Vision Tick Profiler Showing the Hot Methods in the Java Process

AZUL SYSTEMS®

Thread-level Views

Most tools can only gather thread-level information after all the threads have been simultaneously brought to a safepoint. When a subset of the threads continue running before stopping, the threads that were stopped early and could have been doing work are stalled.

Zing is different, each thread is stopped individually to collect information, allowing the other threads to continue working. Zing Vision's thread-level view provides useful information including the threads executing in the application (including JVM housekeeping threads), the stack trace and profile for a thread, what the threads are doing, and where the application is stalled waiting for locks including the method with the lock and the call path to that method.

Name	State	Details
ARTA Thread	I/O wait (0:0:6.358)	Stack Ticks
ARTA Thread	running	Stack Ticks
BenchmarkThread compiler.compiler 1	running	Stack Ticks
BenchmarkThread compiler.compiler 2	running	Stack Ticks
BenchmarkThread compiler.compiler 3	running	Stack Ticks
BenchmarkThread compiler.compiler 4	running	Stack Ticks
Finalizer	waiting on VM lock 'java.lang.ref.ReferenceQueue$Lock' (0:0:1.789)	Stack Ticks
main	waiting on VM lock 'spec.harness.ProgramRunner' (0:0:12.010)	Stack Ticks

Zing Vision Thread View

Java Heap Object Information

The Live Objects tab shows objects in the Old Generation sorted by the cumulative size of all objects of a specific type. The Old Generation Type Velocity tab shows the object instances of each type sorted by the growth rate in memory occupancy of that type. Zing Vision refers to this object type as the type with the highest velocity. If you observe that the size of the Old Generation is continuously growing then the object type with the highest velocity is likely to be involved. Each object type can be expanded to view the references to that object from other objects in the heap. You can also choose the time interval over which you'd like to calculate the growth rate.

Memory - Old Generation Type Velocity

Requested time interval : 00:03:00

Best available match interval : 00:02:35 interval-start : 00:01:13 interval-end : 00:03:48

VM uptime : 00:03:52

Interval: 3 min

Class Name	Latest Count	Latest Size (bytes)	Prior Count	Prior Size (bytes)	Delta Count	Delta Size (bytes)	Growth Rate (bytes/min)
Total	**450,656**	**44,847,072**	**447,846**	**44,646,688**	**2,810**	**200,384**	**77,483**
java.lang.Object[]	13,004	1,589,312	11,941	1,491,744	1,063	97,568	37,727
spec.harness.results.LoopResult	856	41,088	198	9,504	658	31,584	12,212
[VM Internal Type] methodCodeKlass	4,344	556,032	4,118	527,104	226	28,928	11,185
java.util.LinkedList$Node	862	27,584	203	6,496	659	21,088	8,154

Old Generation Object Velocity

Garbage Collection Behavior

Zing Vision captures and displays in-depth information about the Zing runtime's garbage collection and Java heap space use. Users can view garbage collection history that covers the last 50 collections and a detailed summary of New Generation and Old Generation collections with calculated averages for the metrics. Zing Vision also reports on application threads that were blocked due to inability to create a new object, which often indicates that the heap size reserved at application start-up was set too small.

Memory - Summary

Java heap overview and collection count

Metric	Value	Description
Reserved	20.00 GB	The amount of memory requested at VM launch (-Xmx)
Application heap	17.52 GB	The amount of Reserved memory available for holding application objects
Application heap used	8.76 GB	The amount of memory occupied by application objects
Code cache	256.00 MB	The amount of memory reserved for the code cache
Code cache used	17.12 MB	The amount of code cache used
Collection count	22	Number of garbage collections performed since program start

Process's Zing memory

Name	Used	Reserved	Contingency Used
Java system	2.00 MB	1.20 GB	0.00 B
Java heap	10.07 GB	18.80 GB	0.00 B
Java pause prevention	0.00 B	-	-
Total	10.07 GB	20.00 GB	0.00 B

Memory Usage Summary Screen

Lock Contention Diagnosis

Zing Vision provides monitor locking metrics for the total and max times for lock acquisition and the number of lock acquires that were blocked because of lock contention. Users can drill down on the methods to see the call tree and determine where in the application lock contention is occurring.

Monitors - Contention

	Acquire time (ms)[1]		Blocking acquires	Waits		
Name	Total	Max	Count	Count	Max (ms)	Total (ms)
MonitorShowSigDis	8,384,655	14,073	1,937	0	0	0
PGCTaskManager start monitor	1,567	13	6,844	6,821	104,379	17,226,147
PGCTaskManager start monitor	541	6	4,460	5,906	104,178	10,332,159
Threads_lock	307	128	5	0	0	0

Zing Vision Monitor Contention View

ZVRobot

ZVRobot is a monitoring tool that saves Zing Vision "snapshots" that you would see in your browser at time intervals you choose. Each of these snapshots is then available for you to view in a web browser. The information looks just like Zing Vision screens, but drill down is not available.

Summary

Zing Vision is your "flashlight" into a running Java application. Unlike other tools, Zing Vision is designed for use in production and has negligible performance overhead. It helps you find-and fix-production issues quickly. Zing Vision ships with Zing, Azul's innovative Java Virtual Machine. To try Zing Vision, request a trial copy of Zing. It requires no changes to your application and is Java SE compliant.

Contact Us

Email info@azulsystems.com
Phone +1.650.230.6500
www.azulsystems.com/zingvision

AZUL SYSTEMS®
