Инструментарий диагностики java-приложений

Alexander Mokhov
amokhov@naumen.ru

Цель:

- Рассмотреть один из возможных подходов к диагностике проблемы в зависимости от ее проявления
- *Познакомиться* с основными инструментами диагностики и профилирования java-приложений

Что на входе?

Запрос от клиента или извещение от системы мониторинга о неработоспособности приложения.

Что требуется?

- *Быстро* локализовать проблему и восстановить работоспособность приложения
- Детально разобраться с причинами и исключить дальнейшее возникновение подобных проблем

Источники данных для диагностики

Источники данных для диагностики

- Логи: приложения, web-сервера, ОС, gc-логи, сохраненные периодические снимки стеков потоков и т.д.;
- Метрики приложения, переданные в систему мониторинга;
- Если процесс запущен:
  - непосредственно сам процесс приложения (подключиться по JMX, attach API (с той же машины), в случае web-приложения - по http (используя встроенные средства диагностики (например, javamelody));
  - средства мониторинга состояния ресурсов ОС (top, vmstat и т. д.);
- Если процесс аварийно завершен:
  - core-файл;
  - heap-dump

Тестовое приложение

Параметры JVM:

-Xmx128m -Xms50m -XX:+UseG1GC -XX:G1HeapRegionSize=1
-XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath="$app_home/dumps"
-Xloggc:"$app_home/logs/`date +%F_%H-%M-%S`-gc.log" -XX:+PrintGCDetails
-XX:+PrintGCDateStamps -XX:+PrintGCCause
-Dlog4j.configurationFile="$app_home/config/log4j2.xml"
-XX:+UnlockCommercialFeatures -XX:+FlightRecorder

Javamelody

Библиотека, встраиваемая в web-приложение, позволяющая собирать статистику по реальным операциям приложения в production.

С ее помощью можно увидеть:

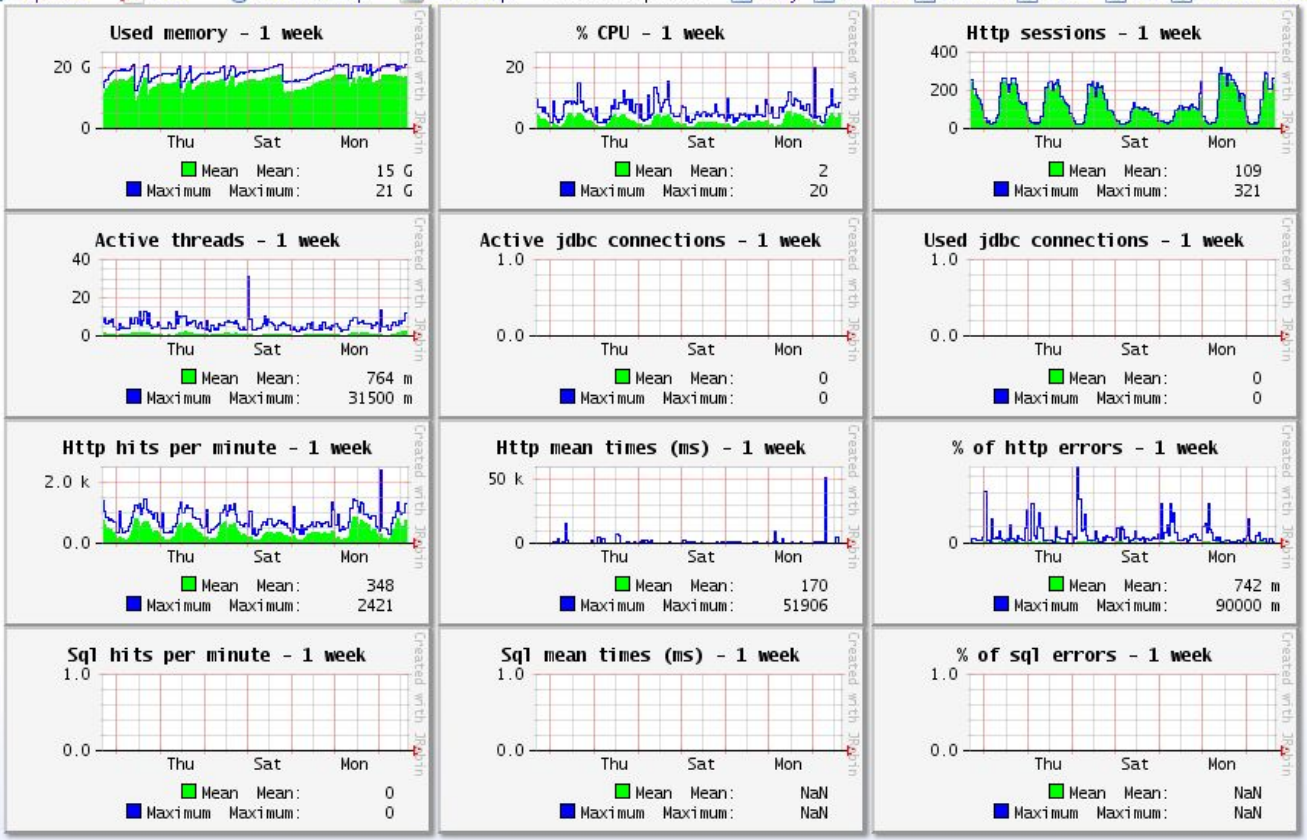
- количество HTTP запросов
- среднее время выполнения
- процент ошибок по HTTP-запросам
- использование памяти
- загрузку CPU
- количество пользовательских сессий и т.д.

Monitoring JavaMelody on /

/monitoring

Statistics of JavaMelody monitoring taken at 9/23/14 9:36 PM on /

Update | PDF | Online help | Desktop | Choice of period : Day | Week | Month | Year | All | Customized



Other charts

1 hits/min on 2 errors Details Last errors

Current requests

Thread	Request	Elapsed time (ms)	Mean time (ms)	Cpu time (ms)	Mean cpu time (ms)	Executed method	Kill
http-bio-9001-exec-111	/$guic/ru.naumen.guic.components.forms.form_jsp?uuid=corebo185180g0000it6l43fiotccj7s&activeComponent=Parameters.EmployeeParameters.EditEmployee POST	3 945 675	2 728	48	588	java.net.SocketInputStream.socketRead0(Native Method)	

57 current requests PDF View in a new page Details

Thread	Request	Elapsed time (ms)	Mean time (ms)	Cpu time (ms)	Mean cpu time (ms)	Executed method	Kill
http-bio-9001-exec-111	/$guic/ru.naumen.guic.components.forms.form_jsp?uuid=corebo185180g0000it6l43fiotccj7s&activeComponent=Parameters.EmployeeParameters.EditEmployee POST	3 945 675	2 728	48	588	java.net.SocketInputStream.socketRead0(Native Method)	
http-bio-9001-exec-10	/$guic/ru.naumen.guic.components.forms.form_jsp?uuid=corebo185180g0000it6l43fiotccj7s&activeComponent=Parameters.EmployeeParameters.EditEmployee POST	3 245 178	2 728	19	588	java.net.SocketInputStream.socketRead0(Native Method)	
http-bio-9001-exec-142	/$guic/ru.naumen.guic.components.forms.form_jsp?uuid=corebo185180g0000it6l43fiotccj7s&activeComponent=Parameters.EmployeeParameters.EditEmployeePhones POST	3 177 414	2 728	12	588	java.net.SocketInputStream.socketRead0(Native Method)	
http-bio-9001-exec-20	/$guic/ru.naumen.guic.components.forms.form_jsp?uuid=corebo185180g0000it6l43fiotccj7s&activeComponent=Parameters.EmployeeParameters.EditEmployee POST	3 095 638	2 728	16	588	java.net.SocketInputStream.socketRead0(Native Method)	
http-bio-9001-exec-119	/$guic/ru.naumen.guic.components.forms.form_jsp?uuid=corebo185180g0000it6l43fiotccj7s&activeComponent=Parameters.EmployeeParameters.EditEmployee POST	3 012 229	2 728	188	588	java.net.SocketInputStream.socketRead0(Native Method)	
http-bio-9001-exec-88	/$guic/ru.naumen.guic.components.forms.form_jsp?uuid=corebo185180g0000it6l43fiotccj7s&activeComponent=Parameters.EmployeeParameters.EditEmployee POST	2 502 245	2 728	16	588	java.net.SocketInputStream.socketRead0(Native Method)	
http-bio-9001-exec-115	/$guic/ru.naumen.guic.components.forms.form_jsp?uuid=employ18518280000k9mk222vhdsndjs&activeComponent=Parameters.EmployeeParameters.EditEmployee POST	2 242 972	2 728	17	588	java.net.SocketInputStream.socketRead0(Native Method)	
http-bio-9001-exec-110	/$volia/servlet.ru.naumen.volia.templates.RenderDocumentTemplateMassive GET	57 186	39 348	1 744	26 032	java.lang.Object.hashCode(Native Method)	
http-bio-9001-exec-90	/voliagwt/servlet.ru.naumen.volia.ui.gwt.rpc.CrashRpcImpl POST	12 844	5 668	142	82	java.net.SocketInputStream.socketRead0(Native Method)	
http-bio-9001-exec-206	/voliagwt/servlet.ru.naumen.volia.ui.gwt.rpc.InquiryRpcImpl POST	2 761	1 905	42	38	sun.misc.Unsafe.park(Native Method)	
http-bio-9001-exec-132	/voliagwt/servlet.ru.naumen.volia.ui.gwt.rpc.ContractRpcImpl POST	2 503	678	31	39	sun.misc.Unsafe.park(Native Method)	
http-bio-9001-exec-157	/voliagwt/servlet.ru.naumen.volia.ui.gwt.rpc.ContractRpcImpl POST	2 304	678	8	39	sun.misc.Unsafe.park(Native Method)	
http-bio-9001-exec-176	/voliagwt/servlet.ru.naumen.volia.ui.gwt.rpc.InquiryRpcImpl POST	1 858	1 905	34	38	java.net.SocketInputStream.socketRead0(Native Method)	
http-bio-9001-exec-150	/voliagwt/servlet.ru.naumen.volia.ui.gwt.rpc.InquiryRpcImpl POST	1 805	977	14	24	sun.misc.Unsafe.park(Native Method)	
http-bio-9001-exec-94	/voliagwt/servlet.ru.naumen.volia.ui.gwt.rpc.InquiryRpcImpl POST	1 776	1 905	40	38	sun.misc.Unsafe.park(Native Method)	
http-bio-9001-exec-135	/voliagwt/servlet.ru.naumen.volia.ui.gwt.rpc.InquiryRpcImpl POST	1 666	1 905	33	38	sun.misc.Unsafe.park(Native Method)	
http-bio-9001-exec-191	/voliagwt/servlet.ru.naumen.volia.ui.gwt.rpc.InquiryRpcImpl POST	1 591	977	14	24	sun.misc.Unsafe.park(Native Method)	
http-bio-9001-exec-72	/rest/exec?authKey=autkey18518so0000lr0tc21jr180fk0&func=VoliaRestUtils.getProposalsByContract POST	1 495	409	113	69	sun.misc.Unsafe.park(Native Method)	
http-bio-9001-exec-137	/voliagwt/servlet.ru.naumen.volia.ui.gwt.rpc.InquiryRpcImpl POST	1 235	977	34	24	java.net.PlainSocketImpl.socketConnect(Native Method)	
http-bio-9001-exec-113	/voliagwt/servlet.ru.naumen.volia.ui.gwt.rpc.ContractRpcImpl POST	1 208	678	16	39	java.net.SocketInputStream.socketRead0(Native Method)	
http-bio-9001-exec-46	/voliagwt/servlet.ru.naumen.volia.ui.gwt.rpc.InquiryRpcImpl POST	1 107	1 905	24	38	sun.misc.Unsafe.park(Native Method)	
http-bio-9001-exec-117	/voliagwt/servlet.ru.naumen.volia.ui.gwt.rpc.InquiryRpcImpl POST	1 007	110	258	26	java.net.PlainSocketImpl.socketConnect(Native Method)	
http-bio-9001-exec-28	/voliagwt/servlet.ru.naumen.volia.ui.gwt.rpc.InquiryRpcImpl POST	995	110	250	26	sun.misc.Unsafe.park(Native Method)	
http-bio-9001-exec-98	/voliagwt/servlet.ru.naumen.volia.ui.gwt.rpc.SalesRpcImpl POST	939	1 266	2	197	sun.misc.Unsafe.park(Native Method)	
http-bio-9001-exec-33	/voliagwt/servlet.ru.naumen.volia.ui.gwt.rpc.InquiryRpcImpl POST	936	296	2	7	java.net.SocketInputStream.socketRead0(Native Method)	
http-bio-9001-exec-80	/voliagwt/servlet.ru.naumen.volia.ui.gwt.rpc.InquiryRpcImpl POST	883	1 905	7	38	java.net.SocketInputStream.socketRead0(Native Method)	
http-bio-9001-exec-99	/voliagwt/servlet.ru.naumen.volia.ui.gwt.rpc.SalesRpcImpl POST	795	1 266	2	197	java.net.SocketInputStream.socketRead0(Native Method)	
http-bio-9001-exec-19	/voliagwt/servlet.ru.naumen.volia.ui.gwt.rpc.InquiryRpcImpl POST	786	977	13	24	sun.misc.Unsafe.park(Native Method)	
http-bio-9001-exec-85	/voliagwt/servlet.ru.naumen.volia.ui.gwt.rpc.InquiryRpcImpl POST	704	1 905	10	38	sun.misc.Unsafe.park(Native Method)	
http-bio-9001-exec-58	/voliagwt/servlet.ru.naumen.volia.ui.gwt.rpc.SalesRpcImpl POST	416	1 266	75	197	java.net.PlainSocketImpl.socketConnect(Native Method)	
http-bio-9001-exec-112	/voliagwt/servlet.ru.naumen.volia.ui.gwt.rpc.InquiryRpcImpl POST	388	977	13	24	sun.misc.Unsafe.park(Native Method)	
http-bio-9001-exec-100	/voliagwt/servlet.ru.naumen.volia.ui.gwt.rpc.ContractRpcImpl POST	321	1 222	70	175	sun.misc.Unsafe.park(Native Method)	
http-bio-9001-exec-69	/voliagwt/servlet.ru.naumen.volia.ui.gwt.rpc.InquiryRpcImpl POST	283	1 905	13	38	sun.misc.Unsafe.park(Native Method)	
http-bio-9001-exec-166	/voliagwt/servlet.ru.naumen.volia.ui.gwt.rpc.InquiryRpcImpl POST	215	977	9	24	java.net.PlainSocketImpl.socketConnect(Native Method)	
http-bio-9001-exec-105	/voliagwt/servlet.ru.naumen.volia.ui.gwt.rpc.InquiryRpcImpl POST	177	977	10	24	org.hibernate.internal.AbstractQueryImpl.setParameter(AbstractQueryImpl.java:479)	
http-bio-9001-exec-207	/voliagwt/servlet.ru.naumen.volia.ui.gwt.rpc.InquiryRpcImpl POST	175	1 905	9	38	sun.misc.Unsafe.park(Native Method)	
http-bio-9001-exec-101	/voliagwt/servlet.ru.naumen.volia.ui.gwt.rpc.ContractRpcImpl POST	157	678	25	39	sun.misc.Unsafe.park(Native Method)	

JConsole и VisualVM

JConsole – простой встроенный профайлер производительности Java, позволяет получить информацию о производительности JVM и потреблении ресурсов. Может подключаться либо к локальным java-процессам, либо удаленно по JMX.

VisualVM — усовершенствованный профайлер, визуально предоставляющий подробную информацию о состоянии java-процесса. Объединяет функциональность нескольких инструментов: jmap, jinfo, jstat, и jstack. Позволяет производить простую профилировку памяти и CPU

Также есть возможность расширять функциональность Java VisualVM при помощи плагинов: функциональность jconsole доступна через соответствующие плагины.

jcmd, jstack, ibm tda

jcmd - консольное приложение для посылки диагностических команд JVM. Позволяет реализовать почти все функции команд jstack, jmap, jinfo и т.д. и является предпочтительным вариантом использования. Далее мы будем говорить про конкретную команду и, где возможно, указывать альтернативу с использованием jcmd.

Пример:

- вывести список всех запущенных java-процессов: *jcmd*
- вывести список доступных команд для java-процесса*: jcmd <PID> help*

jcmd, **jstack**, ibm tda

jstack - выводит stack trace java-потоков для java-процесса или core-файла. Аналогом является jcmd <PID> Thread.print

Пример:

- вывести стек потоков процесса с PID = 4057 с дополнительной информацией о блокировках (например, списком владельцев):

  jstack -l 4057

jcmd, jstack, **ibm tda**

ibm tda - IBM Thread Dump Analyzer - инструмент, позволяющий увидеть зависания, deadlock’и, соперничество потоков за ресурсы и узкие места Java-потоков в удобном графическом виде.

gcviewer, jmap, jhat, Eclipse MAT

gcviewer - небольшой инструмент с открытым исходным кодом, позволяющий анализировать генерируемые JVM gc-логи и представлять их в удобном графическом виде.

gcviewer, **jmap**, jhat, Eclipse MAT

jmap - инструмент, позволяющий сделать снимок текущего состояния java-heap, распечатать маппинг разделяемых библиотек или обобщенную информацию по состоянию java-heap.

Пример:

- вывести java heap summary для процесса PID = 4569 на 64-битной JVM:
  sudo jmap -J-d64 -heap 4569
- сделать heap-dump живых объектов в бинарном формате процесса с PID = 6079:
  jmap -dump:live,file=heapdump.bin 6079

gcviewer, jmap, **jhat**, Eclipse MAT

jhat - Java Heap Analysis Tool, инструмент для анализа файлов java heap dump. После разбора файла запускает web-сервер, что позволяет анализировать результаты при помощи браузера.

jhat поддерживает OQL (Object Query Language). Страница помощи по OQL доступна там же по адресу http://localhost:7000/oqlhelp/ (по умолчанию)

Пример:

- проанализировать файл “java_pid5157.hprof” и запустить web-сервер на порту 7001: *jhat -port 7001 java_pid5157.hprof*
- Пример OQL (все int[] с длиной >= 256): *select a from [I a where a.length >= 256*

gcviewer, jhat, jmap, **Eclipse MAT**

Eclipse MAT - Memory Analyzer Tool, инструмент с открытым исходным кодом, позволяющий диагностировать проблемы с памятью. Его можно использовать для анализа дампов памяти, содержащих сотни миллионов объектов, быстро рассчитывать удерживаемый объектами объем памяти, понимать, что мешает сборщику мусора освобождать память, а также формировать отчеты по вероятным причинам утечек.

Java Flight Recorder и Java Mission Control

JFR и JMC - набор инструментов для сбора низкоуровневой информации во время выполнения приложения с возможностью ее дальнейшего анализа.

JFR - встроенный в Oracle JDK профилировщик и сборщик событий, позволяющий собирать подробную информацию о поведении JVM. Как заявляется, для большинства приложений overhead на запись событий составляют менее 2%.

JMC - набор инструментов, позволяющий осуществлять эффективный анализ данных, собранных JFR.

Резюме

- Познакомились со следующими инструментами:

javamelody, jvisualvm, jconsole, jcmd, jstack, ibm tda, eclipse MAT, gcviewer, jmap, jhat, jfr+jmc

- Использование этих инструментов не является панацеей

Зачастую, все совсем не так очевидно

- Необходимо вырабатывать системный подход

Вопросы?