常用工具及常见问题分析（PDF测试岗位课程）

课程概览

课程名称 *性能测试常用工具及常见问题分析方法介绍*

基本描述

本课程介绍性能测试常用的工具及常见的性能问题分析方法

课程目标：

- 掌握常用工具的基本使用
- 掌握常见性能问题的分析思路

主要学习内容/要点：

- 监控工具nmon的基本使用，nmon监控，生成及分析nmon文件的方法等
- 使用工具分析javacore，gc日志，heapdump等文件
- 常见问题分析思路，TPS上不去，负载不均，TPS忽高忽低等

前言

性能测试过程中使用到的工具有很多，除了需要使用工具发起压力；还经常需要用到工具对应用、系统进行监控，发现问题时要生成各种日志文件并使用工具对其进行分析

本次针对测试过程中常用的分析性能的工具使用进行讲解，并列举几个实例辅助梳理一下分析性能问题的基本思路

第一部分 nmon工具基本使用

第二部分 ISA工具基本使用

第三部分 常见性能问题分析方法

第一部分 nmon工具基本使用

nmon介绍

系统监控是检查系统问题或优化系统性能工作上必不可少的一个环节。通过监视操作系统各项资源的使用情况，间接地反映了服务器程序的运行情况。通过对监控结果的分析可以帮助我们快速定位系统问题或者瓶颈。因此操作系统的监控是不容忽视的，我们通常使用监控工具对操作系统（linux）进行监控。

操作系统监控工具有很多种，包括系统自带的一些基于命令行的工具：top，vmstat，free，iostat，sar，netstat等，以及一些基于web界面的监控工具如zabbix等。我们这次针对我司主要使用的nmon工具的使用进行讲解。

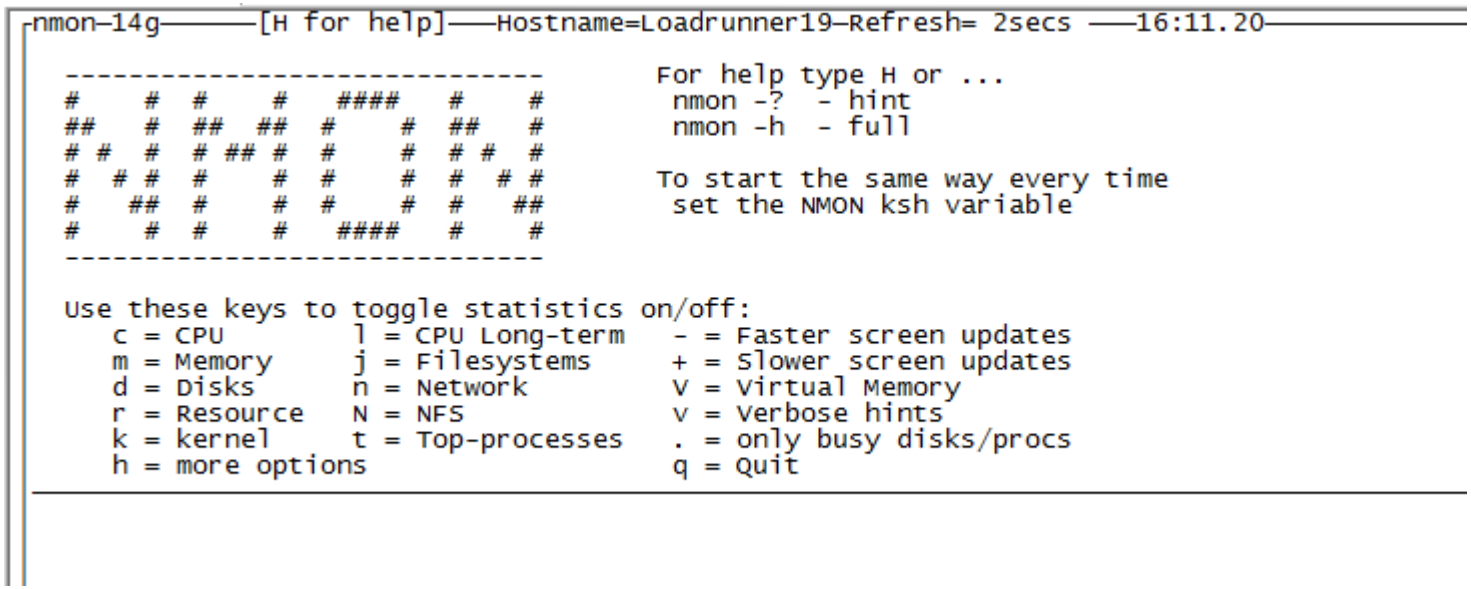
nmon是一种在Aix与各种Linux操作系统上广泛使用的监控与与分析工具，相对于操作系统自带的一些监控工具来说，nmon所记录的信息是比较全面的，它能在系统运行过程中实时地捕捉系统资源的使用情况，并且能输出结果到文件中，然后通过nmon_analyzer工具产生数据文件，与图形化结果。

Nmon可以监控的数据主要包括：CPU 使用情况，内存使用情况，内核统计信息和运行队列信息，磁盘 I/O 速度、传输和读/写比率，网络 I/O 速度、传输和读/写比率，消耗资源最多的进程，虚拟内存使用情况等

第一部分 nmon工具基本使用

nmon安装

nmon是一个独立的二进制文件，从官网下载下来nmon文件，直接解压上传到Linux服务器上，并使用命令chmod给文件赋可执行权即可使用。此处举例的版本是nmon_x86_rhel54；进入到nmon执行文件所在目录，./nmon_x86_rhel54，出现如下界面表示安装成功

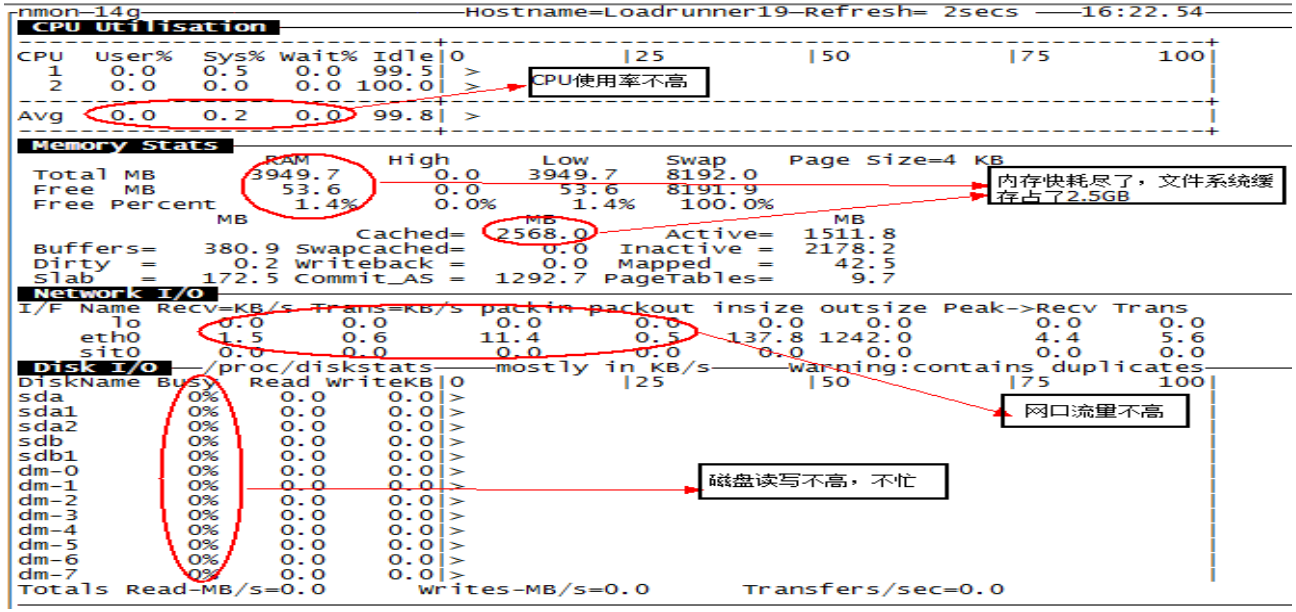


第一部分 nmon工具基本使用

nmon使用

- 实时监控

  执行./nmon_x86_rhel54，可以看到“Use these keys to toggle statistics on/off”提示，根据需要输入对应key；也可以输入“h”打开帮助页面。通常输入c（或者l）、m、n、d、t，可以监控系统cpu，内存，网络，磁盘，以及列出消耗CPU最高的进程，这些是我们通常监控的信息。再次输入对应的key表示收起该监控信息。



第一部分 nmon工具基本使用

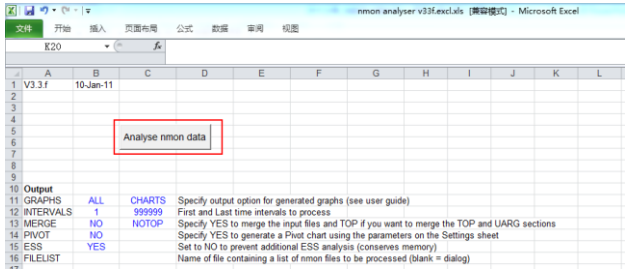
- 生成记录文件

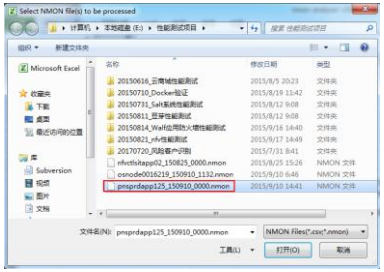
  测试过程中的实时监控通常只能看到一屏的系统信息，性能测试通常需要采集一段时间的操作系统资源使用情况

1. 生成文件：输入命令./nmon_x86_rhel54 –s 15 –c 240 –f，即可在当前目录下生成nmon记录文件。“-s 15”表示每间隔15秒采集一次，“-c 240”表示总共采集240次，计算下来就是3600秒也即采集一个小时的数据。也可以通过-F指定生成文件名称等，可以通过帮助文档查看

```
[root@Loadrunner19 nmon]# ./nmon_x86_rhel54 -s 15 -c 240 -f
[root@Loadrunner19 nmon]# ls -rlth
total 208K
-rwxr-xr-x 1 root root 180K Sep 20  2012 nmon_x86_rhel54
-rw-r--r-- 1 root root  22K Sep 23 16:46 Loadrunner19_150923_1646.nmon
```

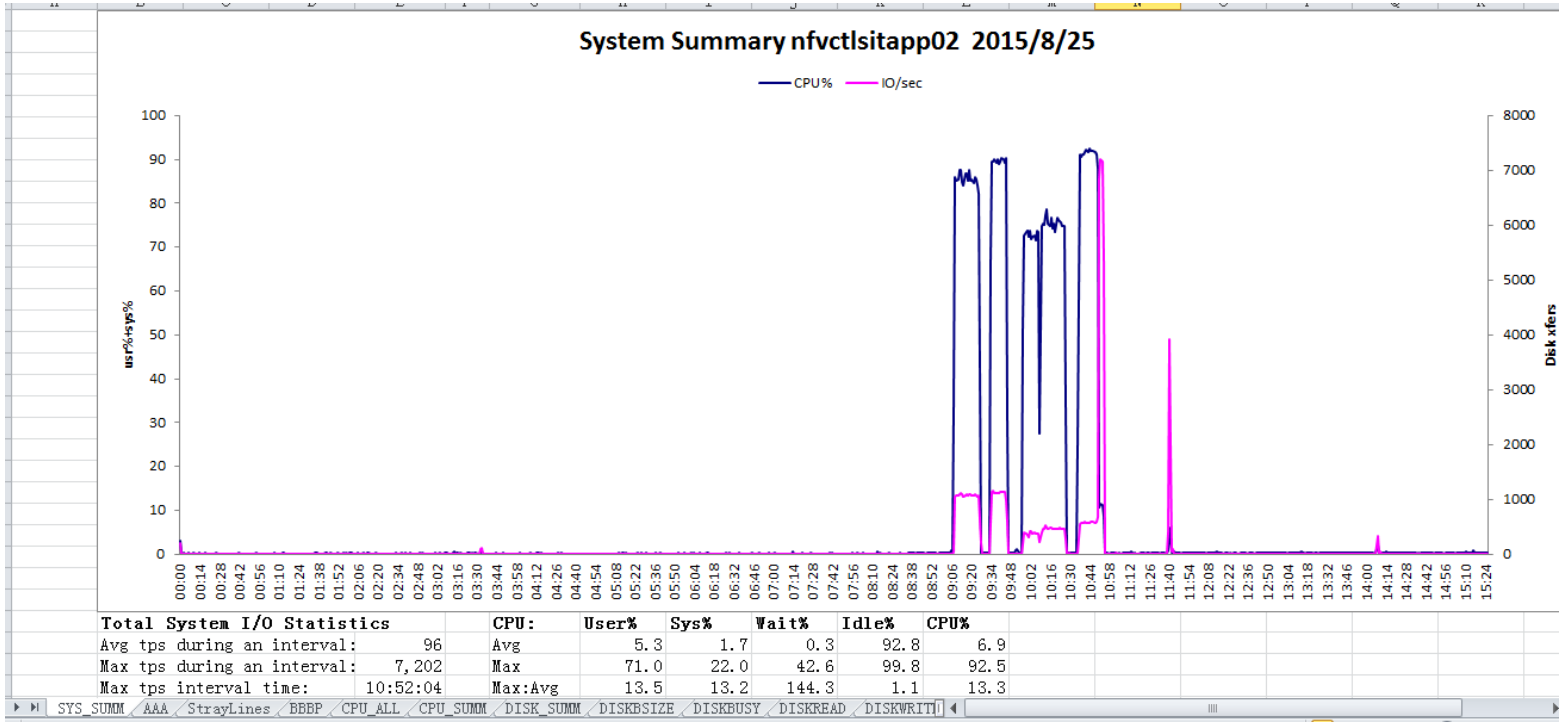
2. 分析文件：生成的文件通过sz或者sftp命令传输到本地，使用工具nmon analyser进行解析，生成可读性较高的excel文件





第一部分 nmon工具基本使用

见到如下文件打开界面表示nmon文件解析成功



Total System I/O Statistics		CPU:	User%	Sys%	Wait%	Idle%	CPU%
Avg tps during an interval:	96	Avg	5.3	1.7	0.3	92.8	6.9
Max tps during an interval:	7,202	Max	71.0	22.0	42.6	99.8	92.5
Max tps interval time:	10:52:04	Max:Avg	13.5	13.2	144.3	1.1	13.3

SYS_SUMM | AAA | StrayLines | BBBP | CPU_ALL | CPU_SUMM | DISK_SUMM | DISKBSIZE | DISKBUSY | DISKREAD | DISKWRIT

第一部分 nmon工具基本使用

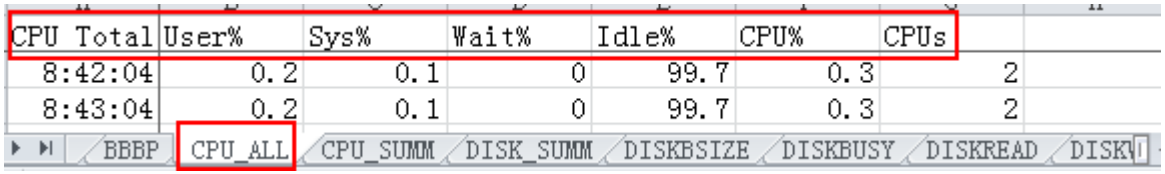
nmon结果文件分析

nmon文件解析出来之后生成的excel表格，有几十个sheet；下面就这些sheet所列指标的含义进行分析。

- 常用指标分析

  监控操作系统，最关注的几个指标分别是CPU、内存、磁盘IO、网络等。下面对几张关键图表进行分析。

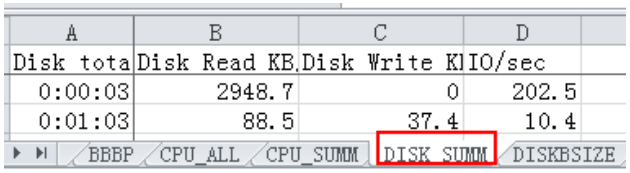
  1. CPU_ALL：所有CPU概述，显示监控系统所有CPU的平均占用情况，包含User/Sys/Wait/Idle状态
     a. User%，用户模式下执行的程序所使用的CPU百分比
     b. Sys%，内核模式下执行的程序所使用的CPU百分比
     c. Wait%，等待 IO 所花的时间百分比
     d. Idel%，CPU的空闲时间百分比，此值和User%，Sys%，Wait%之和等于1
     e. CPU%，CPU总体占用情况，这个值通常等于User%+Sys%+Wait%
     f. CPUs，CPU核数，即操作系统是多少C的



CPU Total	User%	Sys%	Wait%	Idle%	CPU%	CPUs
8:42:04	0.2	0.1	0	99.7	0.3	2
8:43:04	0.2	0.1	0	99.7	0.3	2

第一部分 nmon工具基本使用

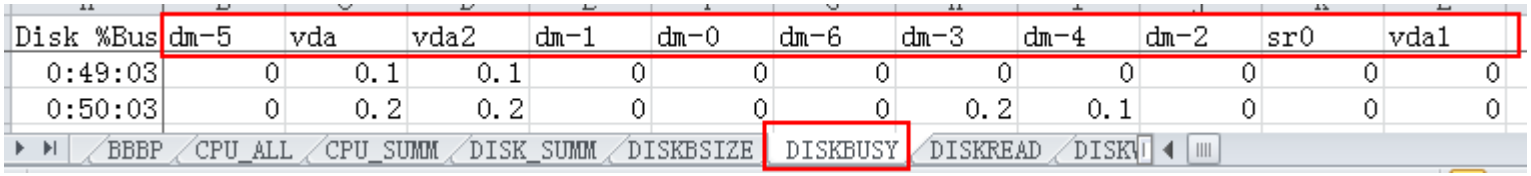
2. DISK_SUM：总体disk读、写以及I/O操作
   a. Disk Read KB/s ，每个磁盘执行采样数据（磁盘设备的读速率）
   b. Disk Write KB/s ，每个磁盘执行采样数据（磁盘设备的写速率）
   c. IO/sec，每秒钟输出到物理磁盘的传输次数



	A	B	C	D
	Disk tota	Disk Read KB	Disk Write K	IO/sec
	0:00:03	2948.7	0	202.5
	0:01:03	88.5	37.4	10.4

BBBP | CPU_ALL | CPU_SUMM | DISK_SUMM | DISKBSIZE

3. DISKBUSY：每个hdisk设备平均占用情况
   单位为%（百分比）



Disk %Bus	dm-5	vda	vda2	dm-1	dm-0	dm-6	dm-3	dm-4	dm-2	sr0	vda1
0:49:03	0	0.1	0.1	0	0	0	0	0	0	0	0
0:50:03	0	0.2	0.2	0	0	0	0.2	0.1	0	0	0

BBBP | CPU_ALL | CPU_SUMM | DISK_SUMM | DISKBSIZE | DISKBUSY | DISKREAD | DISKW

第一部分 nmon工具基本使用

4. MEM：内存使用情况描述，包括物理内存和虚拟内存
   a. memtotal，物理内存总大小
   b. swaptotal，虚拟内存（即交换空间）的总大小
   c. memfree，剩余物理内存大小
   d. swapfree，剩余虚拟内存大小
   e. cached，已占用的文件系统缓存大小，由物理内存分配
   f. buffers，文件系统缓冲区大小
   g. swapcached，虚拟内存中已分配出来的内存大小
   h. inactive，最近不常使用的内存大小

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
Memory MB	memtotal	hightotal	lowtotal	swaptotal	memfree	highfree	lowfree	swapfree	memshared	cached	active	bigfree	buffers	swapcached	inactive
0:15:03	3833.2	0	0	10240	119.6	0	0	10070.3	0	1236	2320	-1	24.3	4.7	1225.7
0:16:03	3833.2	0	0	10240	119.7	0	0	10070.3	0	1236	2319.9	-1	24.4	4.7	1225.8

DISKBUSY / DISKREAD / DISKWRITE / DISKXFER / JFSFILE / MEM / NET / NETPACKET / PROC / VM / ZZZZ / CPU0

5. NET：系统中每个网络适配器的数据传输速率（千字节/秒）
   a. Total-Read，网络适配器每秒接收的数据包总大小，单位是KB/sec
   b. Total-Write (-ve)，网络适配器每秒发送的数据包总大小，单位是KB/sec
   c. eth0-total，网络适配器每秒接收和发送的数据包总大小，单位是KB/sec

A	B	C	D	E	F	G	H	I	J
Network I	lo-read	eth0-rea	lo-write	eth0-writ	lo-total	eth0-tota	Total-Rea	Total-Write	(-ve)
15:08:04	0	24.8	0	4.9	0	29.7	24.8	-4.9	
15:09:04	0	0.5	0	0.7	0	1.2	0.5	-0.7	

DISKBUSY / DISKREAD / DISKWRITE / DISKXFER / JFSFILE / MEM / NET / NETPACKET / PROC / VM / ZZZZ / CPU0

第一部分 nmon工具基本使用

- 其他指标图表

Sheet名称	描述
SYS_SUMM	系统汇总,蓝线为cpu占有率变化情况,粉线为磁盘IO的变化情况；
AAA	关于操作系统以及nmon本身的一些信息；
BBBP	vmtune，schedtune, emstat和lsattr命令的输出信息；
CPUnn	显示执行之间内CPU占用情况，其中包含user%、sys%、wait%和idle%；
CPU_SUMM	每一个CPU在执行时间内的占用情况，其中包含user%、sys%、wait%和idle%；
DISKBSIZE	执行时间内每个hdisk的传输块大小；
DISKREAD	每个hdisk的平均读情况；
DISKWRITE	每个hdisk的平均写情况；
DISKXFER	每个hdisk的I/O每秒操作；
JFSFILE	本sheet显示对于每一个文件系统中，在每个间隔区间正在被使用的空间百分比
NETPACKET	本sheet统计每个适配器网络读写包的数量
PROC	本sheet包含nmon内核内部的统计信息。其中RunQueue和Swap-in域是使用的平均时间间隔，其他项的单位是比率/秒
ZZZZ	本sheet自动转换所有nmon的时间戳为现在真实的时间，方便更容易的分析

第一部分 nmon工具基本使用

第二部分 ISA工具基本使用

第三部分 常见性能问题分析方法

第二部分 ISA工具基本使用

ISA-基本介绍

ISA（全名IBM Support Assistant）是 IBM 提供的一款免费产品。是一个针对JVM环境下JAVA应用进程的调试平台，包含但不局限于对Websphere服务器的性能诊断；通过ISA平台下载各类的调试工具，我们可以分析JVM产生的一些系统文件（ThreadDump，gc.log，HeapDump等）。

ISA主要提供了如下功能：

- 查找信息：可以轻松查找需要的信息，包括特定于产品的信息和搜索功能。
- 分析问题：通过可维护性工具、一系列诊断工件，提供诊断和分析问题的工具，我们可以通过这个功能来分析JVM的一些系统文件。
- 收集和发送数据：使用此功能收集问题确定文件，使用这些文件进行自助问题确定，或者使用“服务请求”功能将这些文件随服务请求一起发送给IBM

下面主要针对ISA的分析问题部分组件进行分享。这是我们诊断和分析性能问题最常用的功能。以IBM Support Assistant 4.1举例。

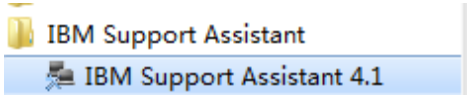
第二部分 ISA工具基本使用

ISA-下载和安装

ISA官网下载地址：http://www-01.ibm.com/software/support/isa/index.html；下载过程不赘述。可以选择语言为简体中文，根据提示操作即可

下载之后解压，进入目录，双击setupwin32.exe；根据指引选择目录，完成安装

安装完成后，启动程序里看到：



点击运行“IBM Support Assistant 4.1”，看到如下页面，表示安装成功

第二部分 ISA工具基本使用

第二部分 ISA工具基本使用

ISA-分析问题-查找及安装工具

打开ISA页面，点击“查找问题”，这时候看到工具目录里面是空的。需要通过查找新的工具加载项来进行安装。操作如下：

1、点击“查找新的工具加载项”，弹出要安装的工具加载项列表

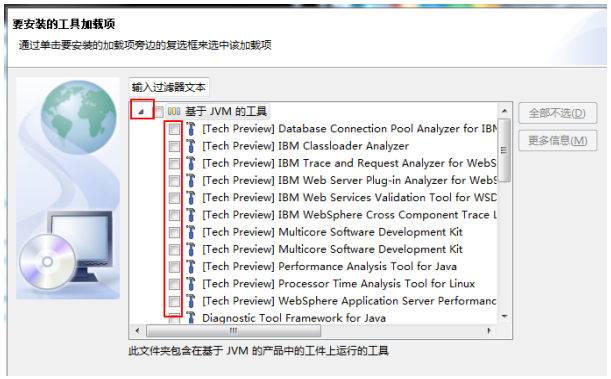
2、点击“基于JVM的工具”，勾选需要安装的工具，点击下一步

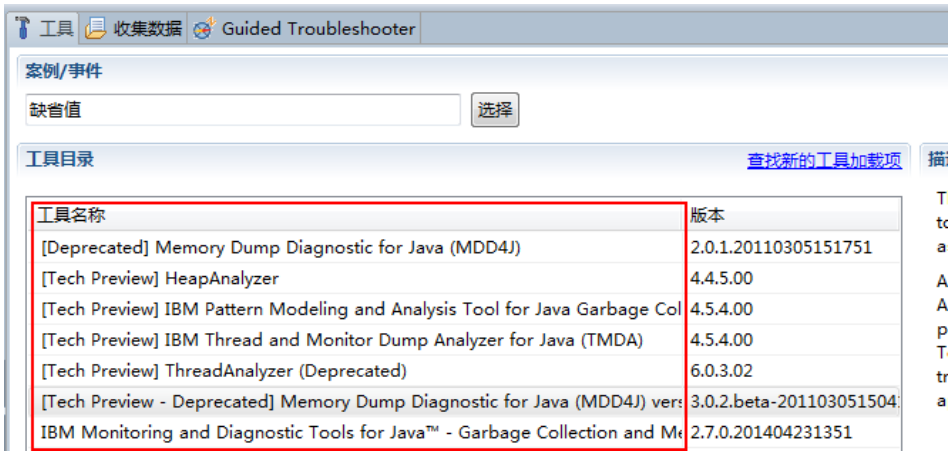
3、选择“我接受许可协议中的条款（A）”，点击下一步

4、展示要安装的工具，点击安装

5、等待下载安装（下载过程比较缓慢），根据提示重启之后可以在列表中看到添加的工具

第二部分 ISA工具基本使用





第二部分 ISA工具基本使用

ISA-常用工具介绍

我们分析性能问题或诊断系统性能瓶颈时，最经常需要分析事项有：JVM垃圾回收情况，线程执行情况，内存使用情况等；而这些过程中产生的系统日志分别有：gc.log，javacore，heapdump。所以最常需要在ISA下载安装的工具有：PMAT，TMDA，HeapAnalyzer等。下面针对这几个最常用的工具使用介绍

- PMAT（Pattern Modeling and Analysis Tool）：可用于分析 IBM 详细 GC 跟踪，分析 Java 堆使用情况，并基于Java堆使用情况的模式建模提供重要配置建议，并提供了一个可能相当有用的不同透视图。

- TMDA（Thread and Monitor Dump Analyzer）：分析一个或多个Java线程转储或javacore，并诊断监视器锁和线程活动，以便确定挂起、死锁和资源争用或监视器瓶颈的根源。

- HeapAnalyzer：分析Java堆转储文件，能够浏览转储以查看其内容。即可以用于分析JVM堆对象。

第二部分 ISA工具基本使用

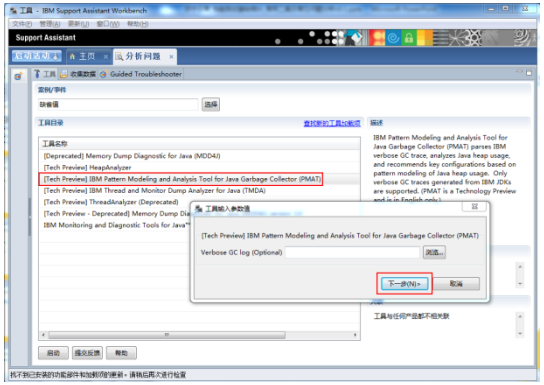
ISA-PMAT使用方法

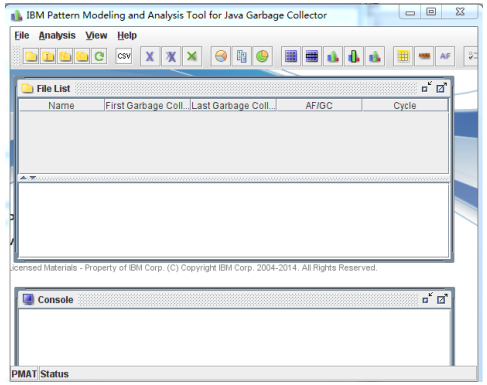
- 分析对象：gc日志
  输出：图形表示形式、html、jpeg 或 csv 文件

- 操作步骤
  1. 从服务器上获取gc日志到本地

```
sftp> cd /opt/jboss/domain/servers/bserver1/log
sftp> get verbose.gc
正在从 /opt/jboss/domain/servers/bserver1/log/verbose.gc 下载 verbose.gc
  100% 39734KB   9933KB/s 00:00:04
sftp> lpwd
C:/Documents and Settings/Administrator
sftp>
```

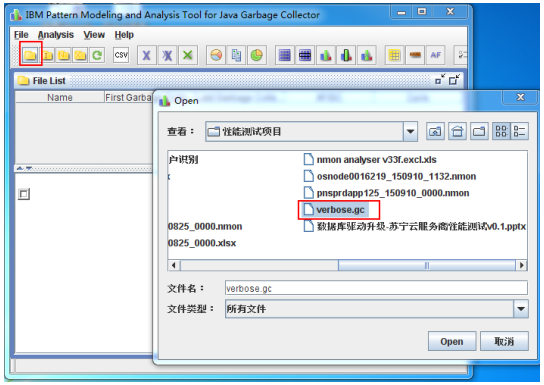
  2. 双击运行，打开PMAT

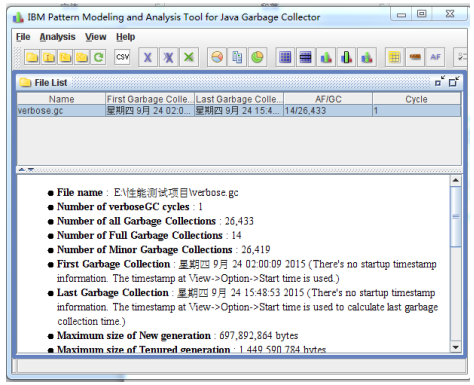




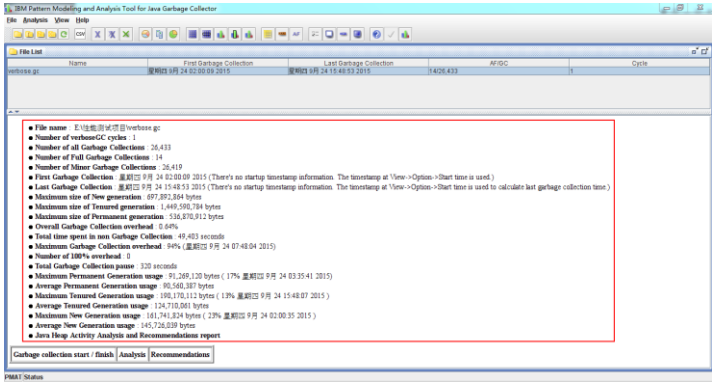
第二部分 ISA工具基本使用

3. 浏览选择gc日志，点击“Open”



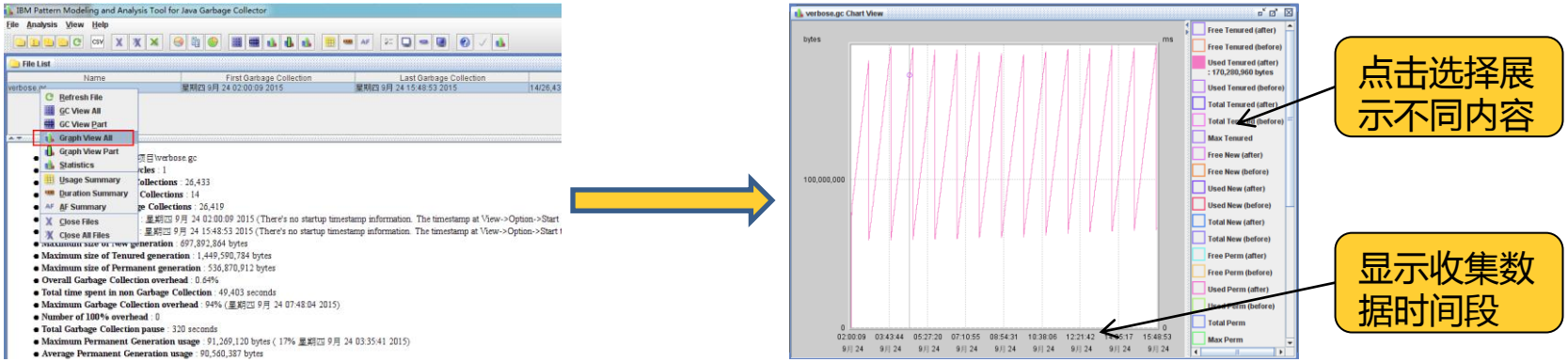


4. GC概况：展示GC文件采集的时间段，GC的基本情况，堆内存的使用情况等信息



第二部分 ISA工具基本使用

5. Graph View All（最常用功能），展示GC情况



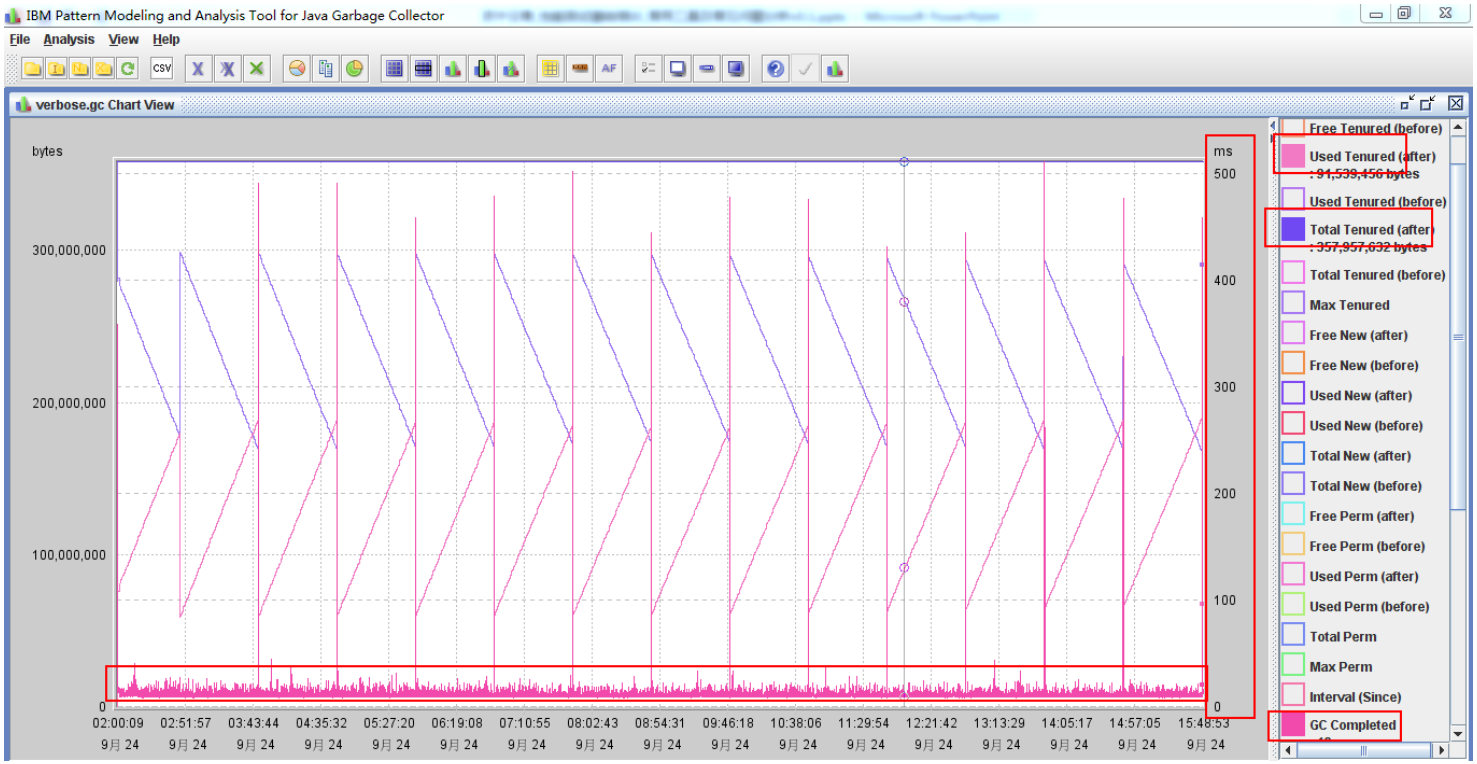
➢ 图表分析：

图表右侧显示可供选择展示项，包括JVM各部分的GC前、GC后的大小变化；GC各个过程消耗的时间等选项。根据需要选择展示。下面对其中一些名词加以说明（不同GC策略的GC日志展示内容略有差异）：

✓ Total 、Used、Free ：这里是指JVM某部分空间总共、已用、剩余部分的大小
✓ before、after：这里是指gc之前和gc之后，通常可以结合比较
✓ Tenured、New、Perm：这里是指JVM内存划分的区域“年老代”、“年轻代”、“永久代”等
✓ GC Completed：GC开始到结束总共花费的时间，单位毫秒

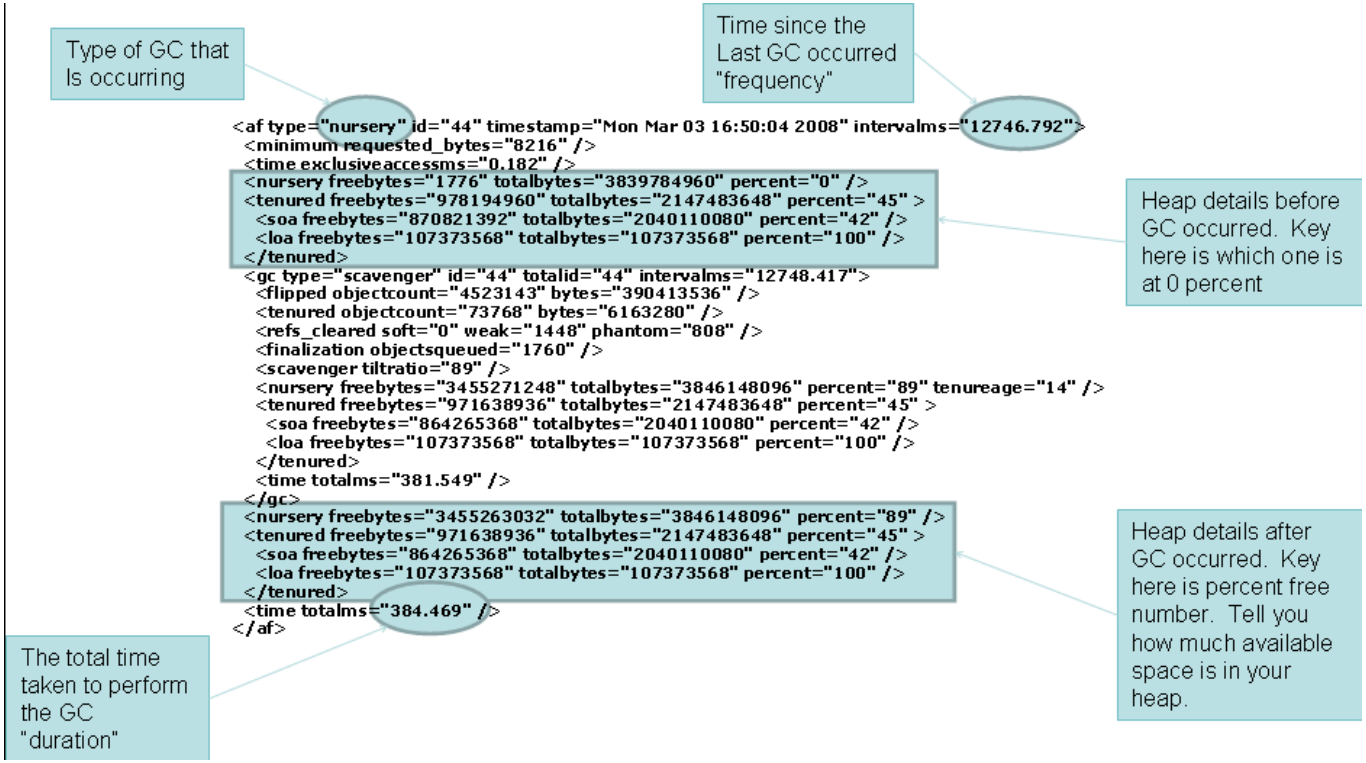
第二部分 ISA工具基本使用

- 图表示例：如图，点选展示GC后JVM年老代部分总共大小、以及已使用的大小；GC花费的时间等信息



第二部分 ISA工具基本使用

- IBM Verbose GC日志源文件：



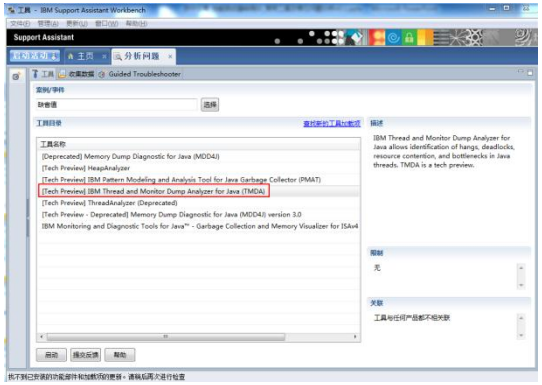
第二部分 ISA工具基本使用

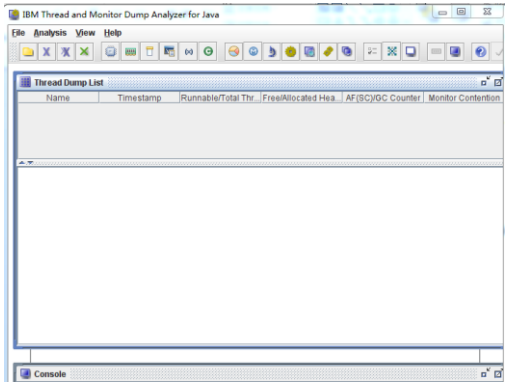
ISA-TMDA使用方法

- 分析对象：javacore或者ThreadDump文件
  输出：基于GUI 的视图展示线程处理情况

- 操作步骤
  1. 从服务器上获取javacore文件，通常在CPU表现不正常或者需要分析的时候，使用kill -3 PID，或者jstack PID>filename生成。可使用sftp或者sz命令传输到本地

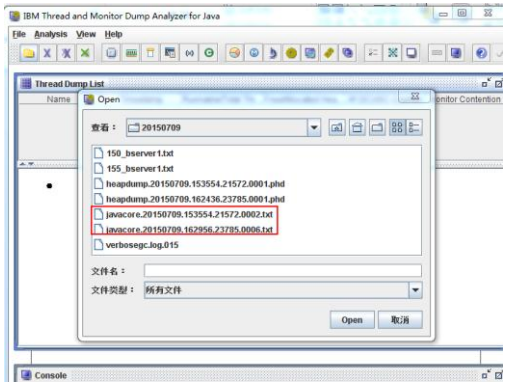
  2. 双击运行，打开TMDA

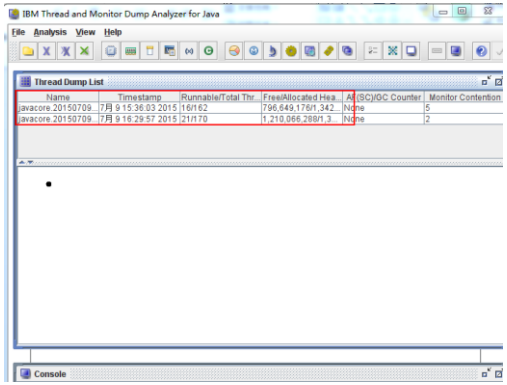




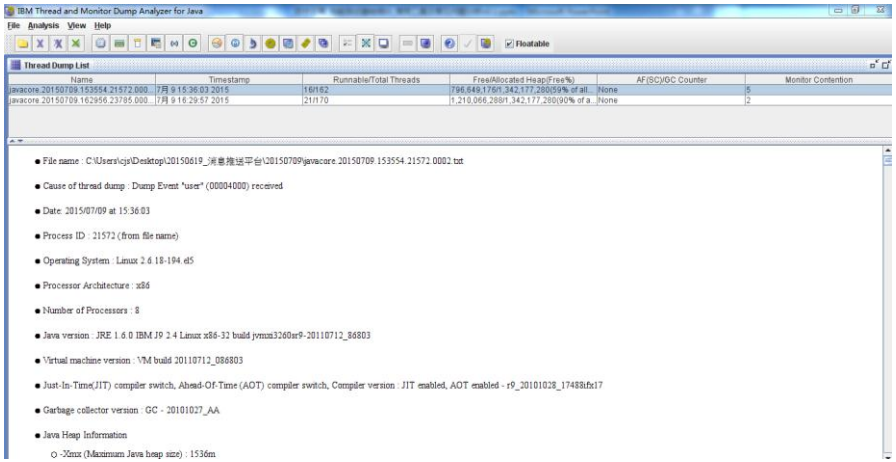
第二部分 ISA工具基本使用

3. 浏览选择javacore或者ThreadDump文件，点击“Open”



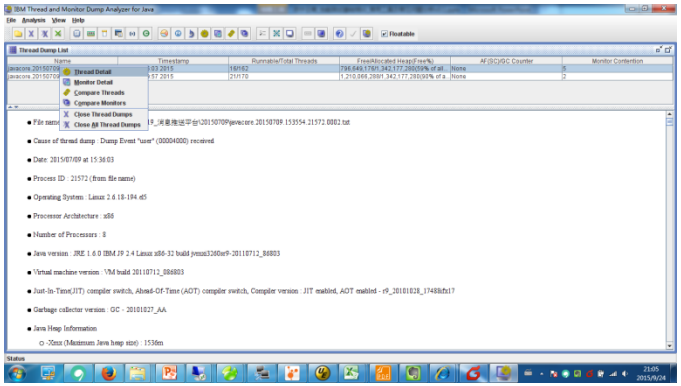


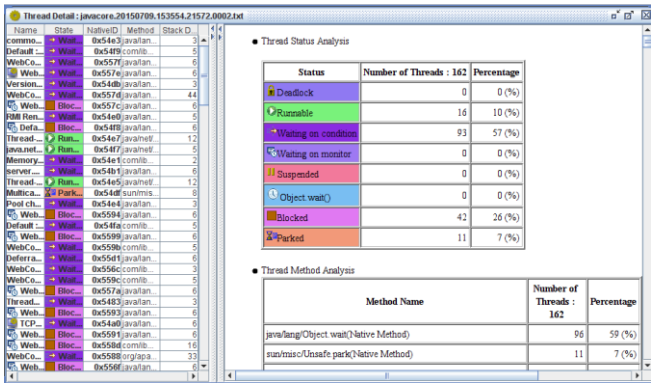
4. javacore概况：展示文件收集时间，JVM基本信息，以及这一刻线程执行的概况及统计信息等



第二部分 ISA工具基本使用

5. Thread Detail（最常用功能），展示线程详细状态





➢ 图表分析：

图表左侧显示线程基本信息，包括线程名称，状态，ID，函数名等，可供选择；右侧默认展示线程概况，左侧线程选中时右侧展示线程执行的状态及具体信息。下面对一些名词加以解释：

✓ 线程状态：Deadlock（死锁），Runnable（执行中），Waiting on condition（等待资源），Waiting on monitor（等待获取监视器），Suspended（暂停），Object.wait()或TIMED_WAITING（对象等待中），Blocked（阻塞），Parked（停止）；下面对各种线程状态进行解释说明

第二部分 ISA工具基本使用

线程状态

状态	状态解释	处理
Deadlock	死锁线程，一般指多个线程调用间，进入相互资源占用，导致一直等待无法释放的情况	重点关注
Runnable	一般指该线程正在执行状态中，该线程占用了资源，正在处理某个请求，有可能正在传递SQL到数据库执行，有可能在对某个文件操作	分析CPU消耗时需要关注
Waiting on condition	等待资源，或等待某个条件的发生；有时候线程空闲时也是这个状态	重点关注
Waiting on monitor	等待获取监视器	重点关注
Suspended	线程挂起	一般不关注
Object.wait()	对象等待中	重点关注
TIMED_WAITING	对象等待中	重点关注
Blocked	线程阻塞，是指当前线程执行过程中，所需要的资源长时间等待却一直未能获取到，被容器的线程管理器标识为阻塞状态	重点关注
Parked	停止	一般不关注

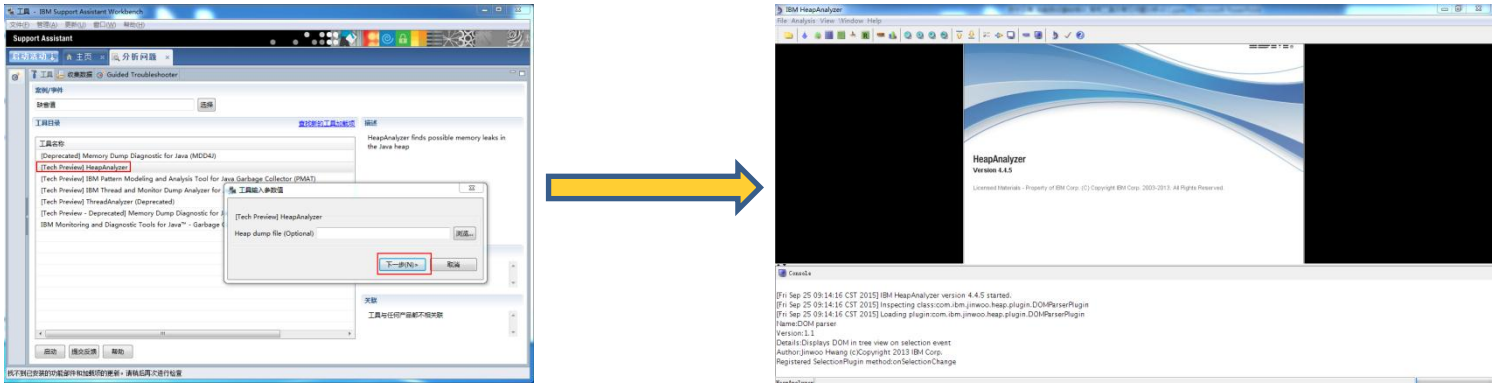
第二部分 ISA工具基本使用

ISA-HeapAnalyzer使用方法

- 分析对象：HeapDump文件
  输出：可疑对象的图形视图和堆浏览器

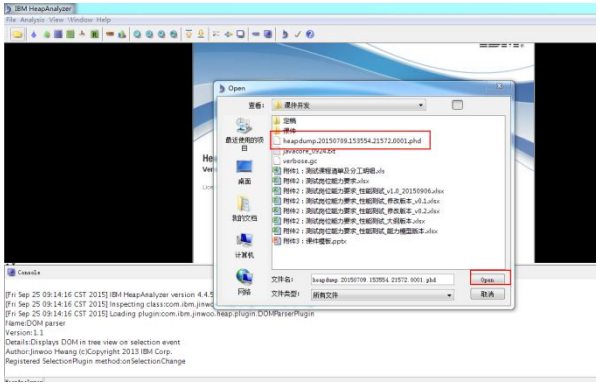
- 操作步骤
  1. 从服务器上获取HeapDump文件，通常在内存出现问题或者需要分析的时候，was需要进行参数配置。使用kill -3 PID，或者jmap -F -dump:format=b,file=filename PID。可使用sftp或者sz命令传输到本地

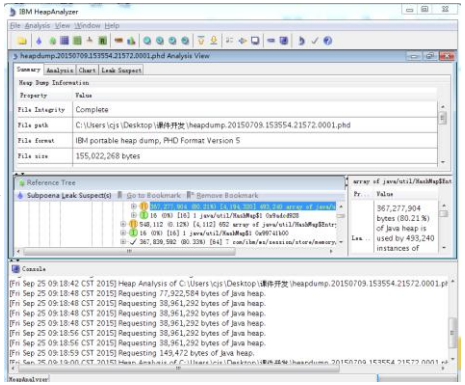
  2. 双击运行，打开HeapAnalyzer



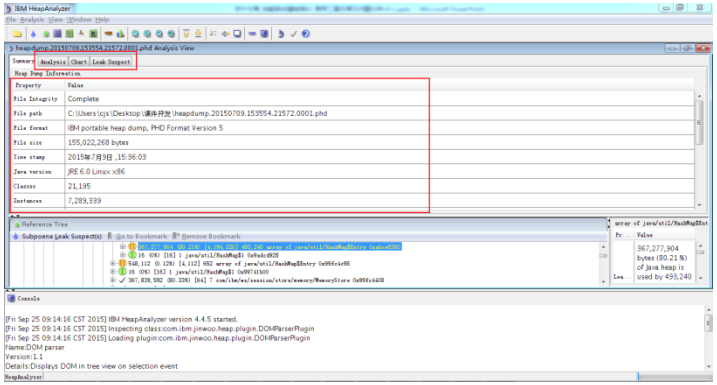
第二部分 ISA工具基本使用

3. 浏览选择HeapDump文件，点击“Open”



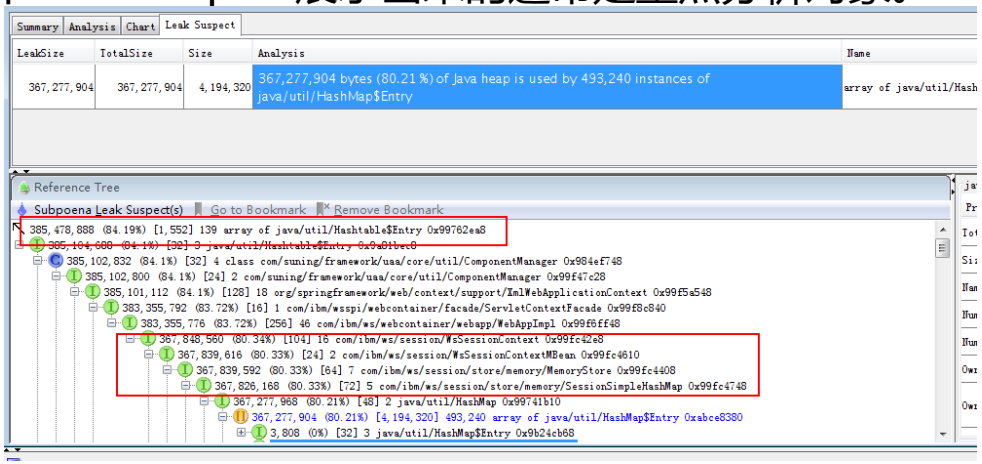


4. HeapDump基本信息：展示文件收集时间，JVM中class数目，实例个数等信息



第二部分 ISA工具基本使用

5. HeapDump分析（最常用功能），展示java堆中对象以及可能泄露对象的分析。点击“Analysis”，“Chart”，“Leak Suspect”可以分别展示一些堆内存中对象总数量以及大小所占的比例等信息，并提供一个“泄露嫌疑”对象的列表。这个Leak Suspect展示出来的通常是重点分析对象。



- Leak Suspect图表分析：

图表上半边模块显示怀疑泄露的类的基本信息以及统计信息，下半边模块展示每个节点树的大小占总的堆栈大小，然后是这个类的在内存中的大小，以及“包含”的子对象，注意这边的子对象是指在这个类中定义的，而不是继承关系中的子类对象。点击占比最大的或者数量最多的，并进一步分析出可能泄露对象。上面的图表中稍有经验的工程师不难看出是由于Session对象占用了大量的内存（注意：这里泄露嫌疑对象不一定就肯定泄露，得持续观察看是否能够被GC）

第二部分 ISA工具基本使用

ISA-分析工具的其他启动方式

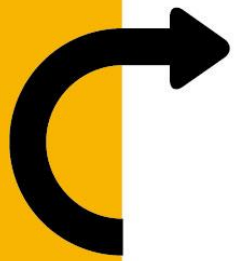
通过ISA启动工具分析gc.log，javacore，HeapDump，其实就是通过java调用工具的jar包。我们可以通过命令行直接调用。

获取对应的jar包。ga435.jar，jca433.jar，ha39.jar等，数字表示不同的版本号。

启动方式：

1. PMAT：java –Xmx[heapsize] –jar ga435.jar
2. TMDA：Java –Xmx[heapsize] –jar jca433.jar
3. HeapAnalyzer：java –Xmx[heapsize] –jar ha39.jar

其他分析工具，根据需要获取使用。这边不再赘述。



第一部分 nmon工具基本使用

第二部分 ISA工具基本使用

第三部分 常见性能问题分析方法

第三部分 常见性能问题分析方法

什么是性能？

性能，即系统处理功能或者业务的能力。下面从不同角度阐述软件性能。

用户角度，关心系统的响应时间，即访问浏览网站或者提交登陆，多长时间展示出来或登陆成功；不关心有多少人和“我”同时访问、提交

系统角度，关心每个用户的响应时间（平均响应时间），以及系统能支撑多少用户访问（用并发用户或者TPS来衡量），服务器消耗多少资源（CPU、内存、磁盘IO、网络等）以及系统会不会宕机等

什么是性能问题？

性能问题（现象），通常是指应用系统出现的响应慢，响应错误（这里需要区别于功能），甚至不响应等情况；这类问题不局限于并发的情况下出现。

常见的性能问题（原因）有：内存溢出（即OOM）、CPU负载较高、磁盘IO高、负载不均（集群模式）等等；通常会导致上述现象是由这些原因导致的，而进一步分析，这些问题也是有更深层的原因引发。下面通过举例的方式分析几种问题的分析方法（指在压测过程中）

第三部分 常见性能问题分析方法

内存耗尽（JVM）

问题现象：用户感觉系统响应非常慢，甚至出现不响应，即“卡死”的现象

系统表现：JVM内存得不到释放甚至溢出

分析步骤：

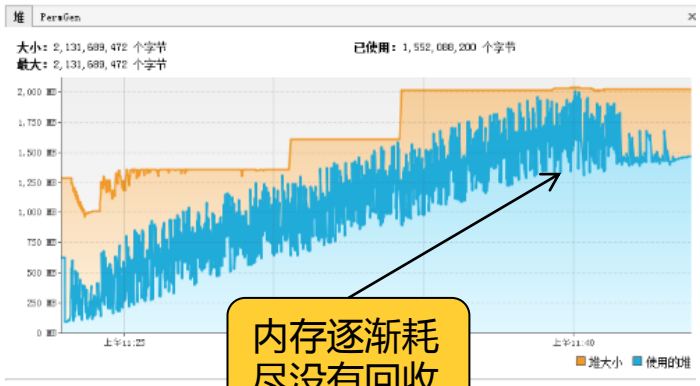
1. 实时监控服务器：包括通过gc日志或者其他手段（jconsole、jvisualvm等）监控JVM表现，如果发现JVM剩余空间逐渐减少，继续压测观察内存能否释放，如果不能释放，或者大量的full gc（global gc），说明JVM内存出现问题

2. 检查相关配置，机器物理内存总大小、使用大小，JVM大小配置是否合理，gc策略配置（一般默认gc配置没问题）等

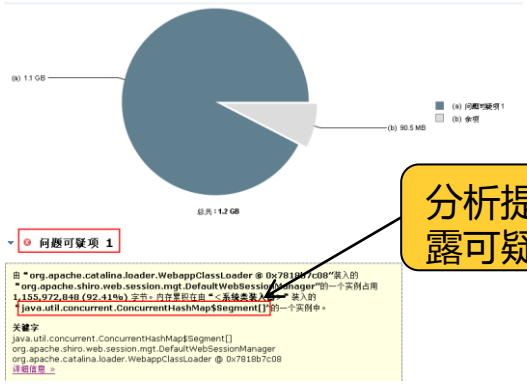
3. 收集相关日志：收集gc日志，HeapDump文件等，保存到本地。通过使用ISA工具解析gc日志，HeapDump文件进一步分析出问题根本原因，并提出优化意见

第三部分 常见性能问题分析方法

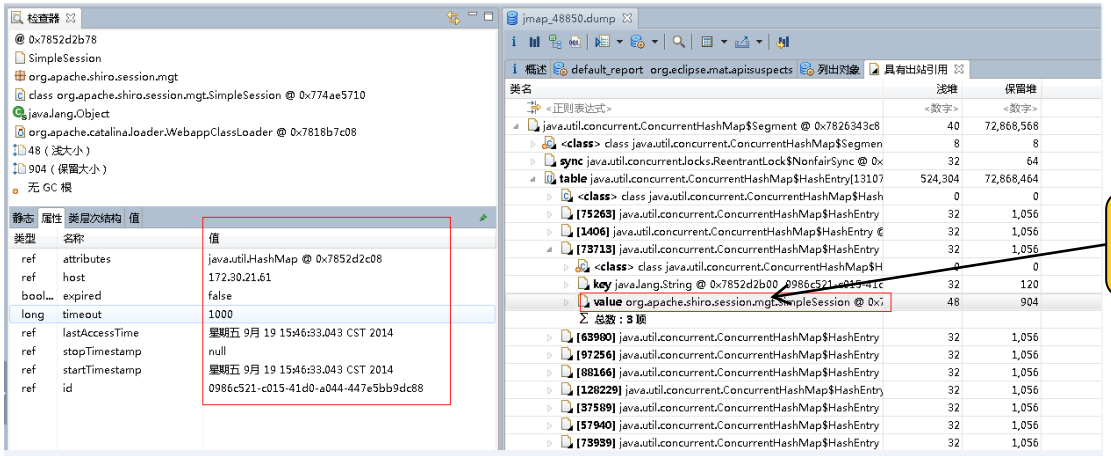
下面是一个OOM的示例



内存逐渐耗尽没有回收



分析提示泄露可疑项



进一步分析定位

第三部分 常见性能问题分析方法

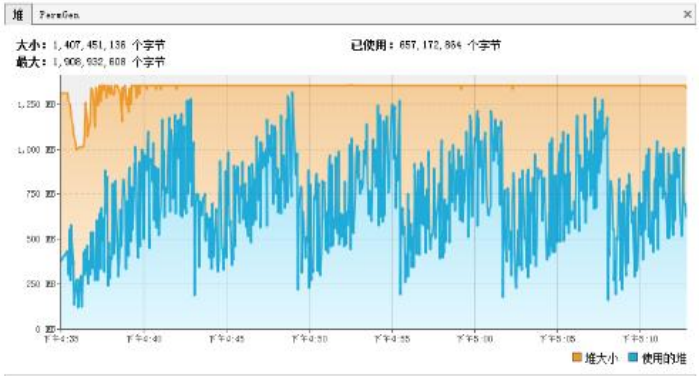
解决过程：

1. 通过分析发现JVM内存溢出，年老代渐渐耗尽，导致频繁的Full GC

2. HeapDump分析，发现是由于org.apache.shiro.web.session.mgt.DefaultWebSessionManager引用的java.util.concurrent.ConcurrentHashMap$Segment导致

3. 调整工程下的WEB-INF/classes/config/spring下的beans.xml中对Session对象管理的配置，调整为每60秒调度一次，将ConcurrentHashMap中的失效的Session对象remove掉。

调整后再次压测效果（GC回收稳定，系统响应正常）：



第三部分 常见性能问题分析方法

CPU负载上不去：

问题现象：增加负载，TPS不变，响应时间变慢

系统表现：服务器的CPU消耗均不高

分析步骤：

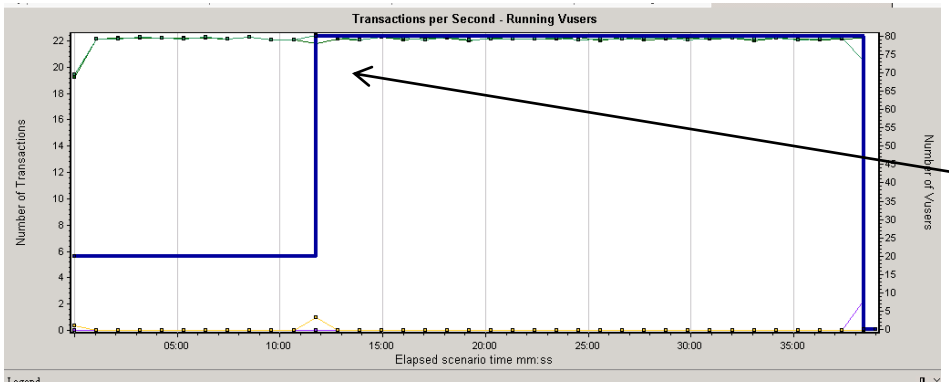
1. 实时监控服务器：通过nmon，top等监控服务器CPU，继续增加负载，TPS上不去的时候如果服务器CPU也上不去，说明系统存在性能瓶颈（前提压力发生器不是瓶颈）

2. 检查相关配置：比如压力发送的地址是否正确，服务器中调用数据库、缓存等服务器ip端口是不是配置有误等

3. 收集相关日志：包括应用日志，ThreadDump文件等；对这些文件进行分析，应用日志是否有报错提示，分析ThreadDump采集的线程信息，进一步找到原因

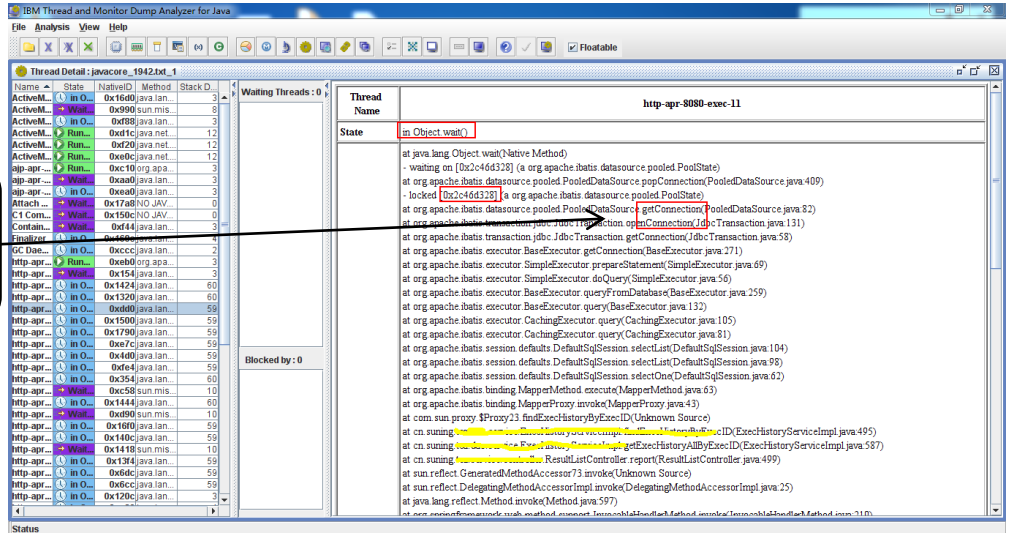
第三部分 常见性能问题分析方法

下面是一个TPS上不去的示例



增加负载，TPS没有变化；监控服务器CPU也没变化

分析ThreadDump，大量线程在等待数据源连接



第三部分 常见性能问题分析方法

解决过程：

1. 分析发现大量线程等待获取数据源连接，检查数据源连接池配置，发现没有配置，使用的是默认的配置；查询官网得知默认配置连接池大小是8

2. 根据官网说明文档配置，将连接池大小设置为200，再次压测，TPS能上去了。（这边200不一定是最佳配置）

第三部分 常见性能问题分析方法

补充说明

- ✓ 上述分析问题的思路均是对较简单的单独的系统进行分析，不包含系统与系统之间的调用分析。
- ✓ 上述问题分析过程中，排除了性能测试工具、基础网络环境等因素。测试人员根据情况判断
- ✓ 诊断性能问题之前，性能测试人员需要对系统整体的架构、调用关系甚至处理逻辑有一定的了解，这有助于提高分析发现性能问题原因的效率
- ✓ 建议性能问题分析整体思路：
  - 从前往后：根据请求流转路径从前往后分析
  - 从整体到局部：根据系统整体表现判断瓶颈所在，并进一步分析该问题

Thanks!