Created by: Raúl Castillo

Using linear regression to detect memory leaks

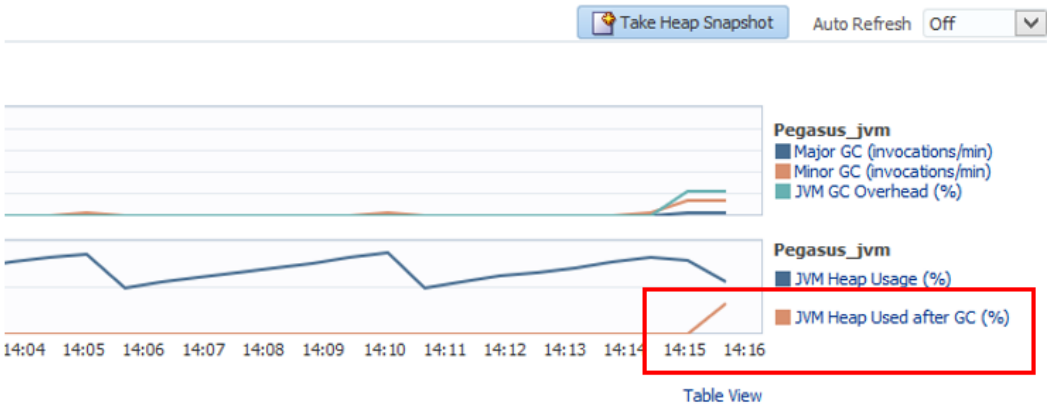
The aim of this document is to talk about one of the most important problems that affect application servers based on the Java Virtual Machine, memory leaks. In order to have a detailed explanation of this phenomenon, you could review [1]. According to this reference: *"…a memory leak belongs to a class of situations called software aging problems. Software aging defines the loss of performance over the time because of the gradual accumulation of little problems. Other term within this kind of problem is the rejuvenation process. For example, rejuvenation happens when the system has to be restarted in order to free the memory accumulated by a Java Virtual Machine (JVM) process after a period of time…"*

Based on the experience dealing with these kinds of problems, it would be interesting to use software to monitor the garbage collector in order to recognize the memory leak's symptoms in advance. In this way, a system administrator could program a restart process (rejuvenation) avoiding a crash. Of course there are systems that could execute tasks like these, but many times they are not used (this affirmation is based on the author's experience) because of the cost of licenses and the lack of experience.

Therefore, in the following lines a demonstration about how to use Java, JMX and linear regression to detect memory leaks will be presented.

An important fact from a previous post

In [1] an important picture was included, this picture shows that if the Java Virtual Machine is affected by a memory leak problem, the full garbage collector will not free enough memory from the heap.



Source: [1]

Based on this observation, monitoring the activity of the full garbage collector and analysing its trends over the time could be useful to detect memory leaks.

Created by: Raúl Castillo

The elements needed to construct the monitor

In order to construct this monitor it is necessary to detect the Garbage Collection execution, save the data about the execution, create a time series and analyse it to detect some trends and features.

- Monitoring the garbage collector. In order to monitor the garbage collector two ways arise. The first is reading the JVM memory consumption and the number of collections executed based on a period. The second one is installing a listener on the JVM that notifies each garbage collection event to a Java client, this way was selected and the code recommended by [2] was used.

- The time series data structure. Each time the garbage collection is executed, a notification is generated. This data is saved on a time series built using JFreeChart [3]. It is important to remark that each point of the time series represents the time when the garbage collector is executed (x-axis) and the amount of memory used after that execution (y-axis).

- The linear regression model. One of the matters that this kind of software has to face is related to the model to be used on the time series. According to [4] it is possible to use linear models such as linear regression, but it is also stated that non-linear models can be useful. For simplicity linear regression was selected in order to detect trends.

- The analysis. The analysis made by this demonstration software is based on the information given by the line obtained using linear regression. Thus, the slope is analysed to detect whether there is a decreasing, increasing or constant trend.

Testing the software

- ### Test

  In order to execute this test, this configuration was used:

```
set USER_MEM_ARGS=-Xms2048M -Xmx2048M -Xmn1024M -XX:PermSize=512M -
XX:MaxPermSize=512M -Xss256K -XX:SurvivorRatio=8 -XX:TargetSurvivorRatio=90 -
XX:+UseParallelOldGC -XX:+AlwaysPreTouch -XX:+ParallelRefProcEnabled -
XX:MaxTenuringThreshold=15 -XX:-UseAdaptiveSizePolicy -XX:+DisableExplicitGC
-Dweblogic.threadpool.MinPoolSize=50 -XX:+HeapDumpOnOutOfMemoryError -
XX:+PrintGCDetails -XX:+PrintGCTimeStamps -XX:+UseGCLogFileRotation -
XX:NumberOfGCLogFiles=10 -XX:GCLogFileSize=10M -
Xloggc:C:\u02\oracle\config\domains\Testdomain\GClogs\gclog_WLS_01.txt
```

  These charts show the results of this test.

Chart 01

The following screen shows the memory consumption after the execution of minor collections on the Eden Space. This constant slope is according to the expected

Created by: Raúl Castillo

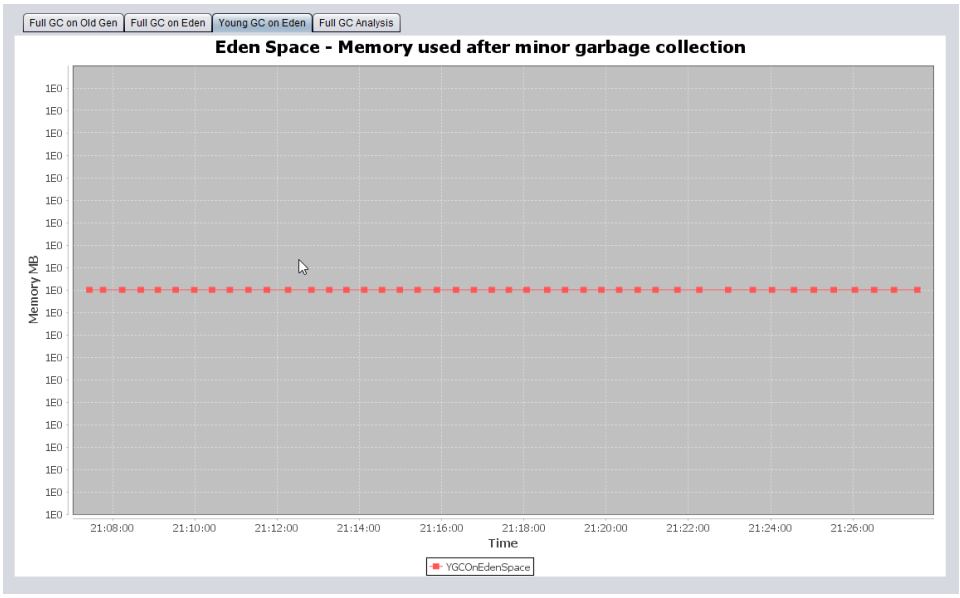


Chart 01: Memory consumption on the Eden Space after minor collections

Chart 02

Here there is one of the important charts given by this program. If you see "Chart 01" again you could realize the last minor collection happened at 21:29 and as can be seen in "Chart 02" since that time (21:29) full collections were executed several times. However, the memory used in the tenured space continued growing up until filling the whole space available.

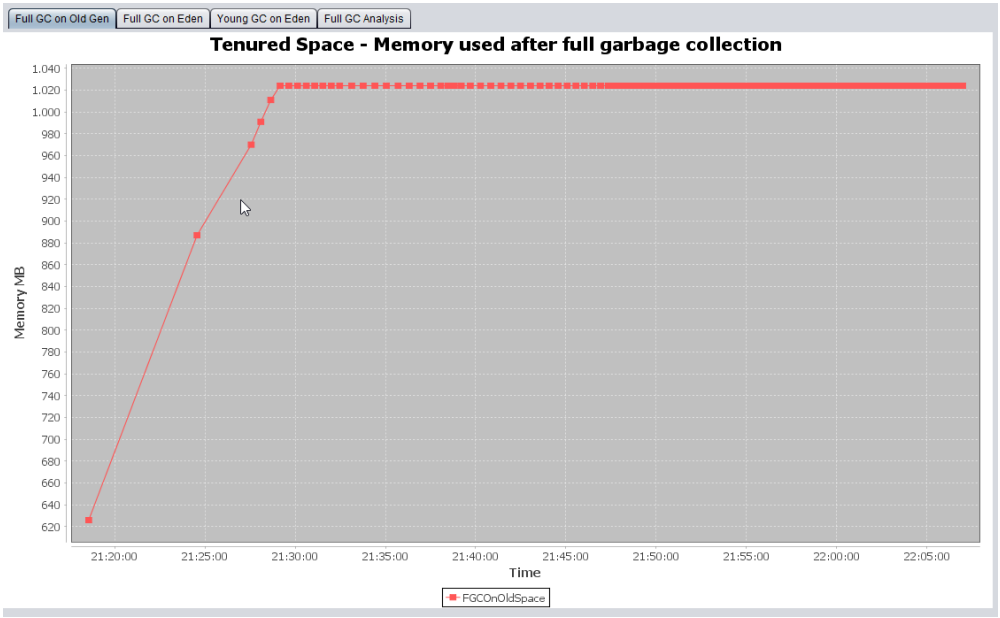


Chart 02: Memory consumption on the Tenured Space after full collections

Created by: Raúl Castillo

Chart 03

As in the previous case, it is possible to see that after 21:29 (when the tenured space was filled completely) the consecutive executions of full garbage collections were not enough to free the Eden Space. This situation will lead to the crash of the JVM due to a memory leak.

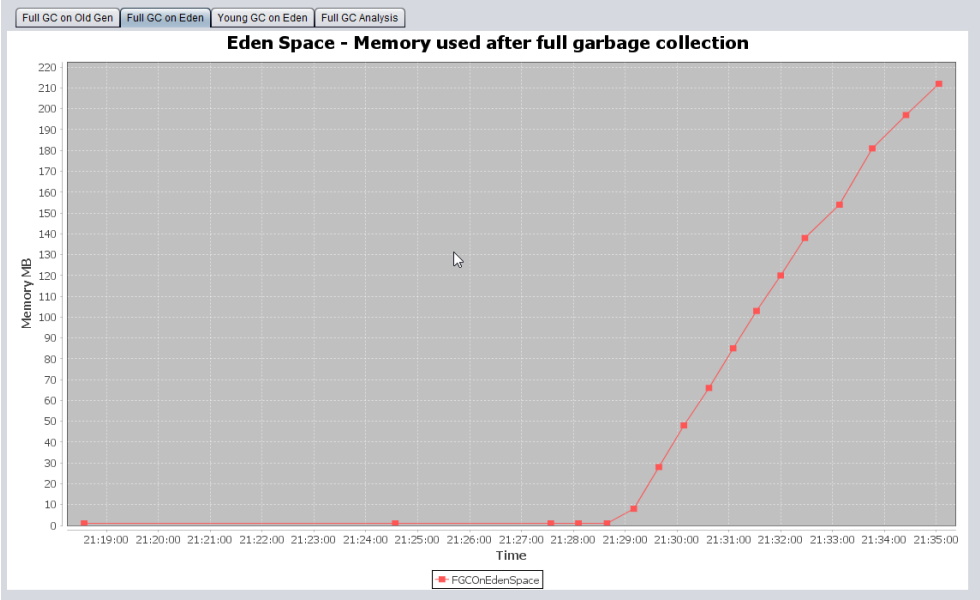
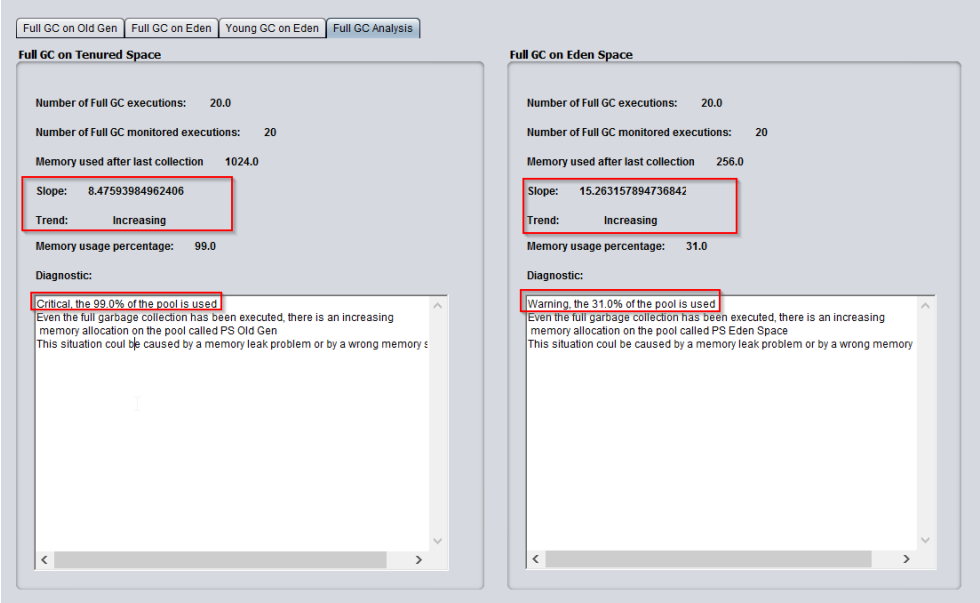


Chart 03: Memory consumption on the Eden Space after full collections

Created by: Raúl Castillo

The analysis tab

With enough experience, an analyst could see these charts to identify that there is a problem in progress. However, with lots of application servers to analyse it is useful to have tools that alert administrators in order to investigate these situations quickly. For this reason, this tab shows the result of the analysis made using the slope and the execution of full garbage collections. Based on this information an alert is triggered in order to avoid unexpected crashes.



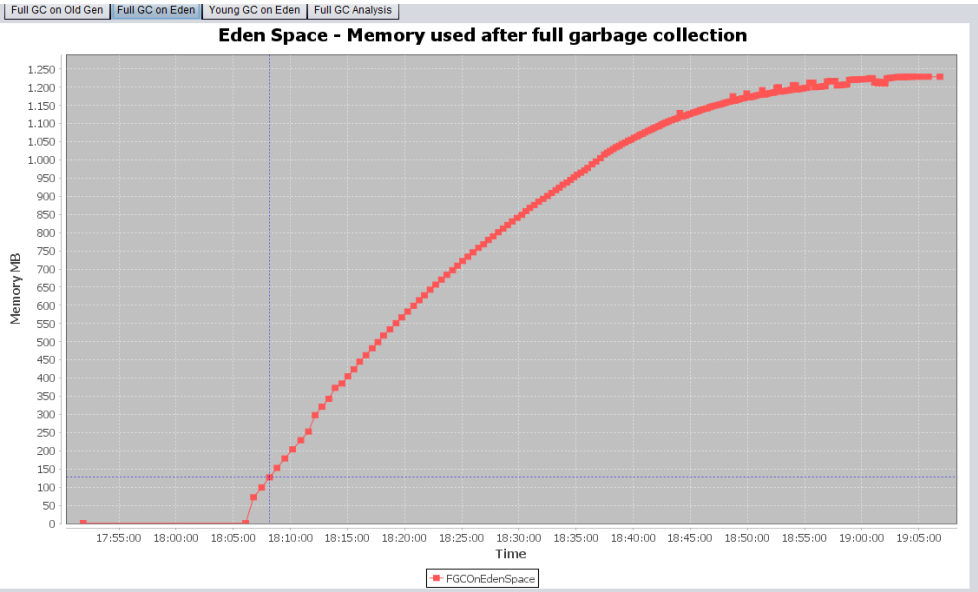
The analysis tab: As can be seen the text indicates the occurrence of a problem based on the slope analysis.

Limitations.

This is just a demonstration about using time series and linear regression to find trends, which should allow system administrator to avoid problems within the application server. With this in mind, the following limitations are remarked:

1. The time series data is not saved in any database or file.
2. It is not possible to select the time series model to identify a trend.
3. It is not possible to use seasonal data. For example, consumption caused during some periods such as Christmas or any other important date or time.
4. Even line regression was useful for this demonstration. It seems to be other models could be applied. It is suggested by the time series given for the software over a longer time.

Created by: Raúl Castillo



Conclusion

As we are focused on business applications, we often forget how useful mathematics are to model some phenomena. With this in mind, developing software to analyse situations such as software aging could give us an important advantage with our customers. Beyond this, there are other methods such as artificial neural networks, Markov chains, etc. that could be very useful to improve our IT operations.

This document would not be written without [5]

References list

[1] Using Oracle Cloud Control 12C to Analyse a Memory Leak Problem
http://blog.sysco.no/memory/leak/memory-leak/

[2] Garbage Collection JMX Notifications http://www.fasterj.com/articles/gcnotifs.shtml

[3] JFreeChart: Time Series Demo 8 : Time Series Chart « Chart « Java
http://www.java2s.com/Code/Java/Chart/JFreeChartTimeSeriesDemo8.htm

[4] Using IT Analytics Cloud Service
https://docs.oracle.com/cloud/latest/em_home/ITACS/GUID-812BEEFC-3E54-41B8-BCB3-929E4BA404CF.htm#ITACS-GUID-812BEEFC-3E54-41B8-BCB3-929E4BA404CF

[5] AC/DC - For Those About to Rock (We Salute You)
https://www.youtube.com/watch?v=RtMGoU9NcMo