Der MemoryWatcher

Überblick über das MemoryWatcher-Plugin

by Thorsten Kamann, Peter Roßbach

NOTICE:

Der MemoryWatcher ist für die Kontrolle des Speicherbedarfs des gesamten Servers zuständig. Er führt automatisch ab einer bestimmten Speicherauslastung regelmäßig einen GarbageCollector und wenn eine einstellbare kritische Marke überschritten ist einen Restart aus.

1. Idee der Speicherüberwachung

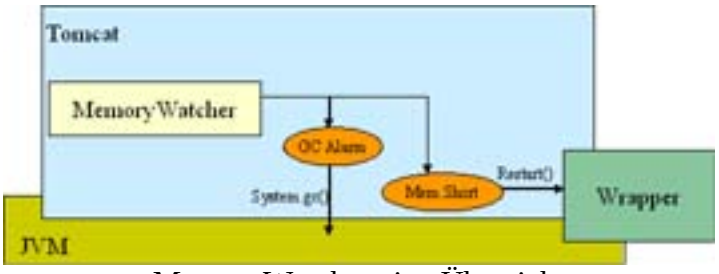
Auf einem Server in einem Hosting-Umfeld werden für gewöhnlicher die unterschiedlichsten Anwendungen installiert. Jede Anwendung bringt meist viele Klassen und Archive mit, die Speicher durch die Nutzung brauchen.
So unterschiedlich die Anwendungen, so unterschiedlich die Erfahrung und das Können der oder des Programmierer(s). Damit direkt verbunden ist auch die Qualität der Anwendungen.
Um einen Server in arge Speicherprobleme zu manövrieren reicht schon der folgende Code:

```
int i=0; while (i<10){ do anything; }
```

Hier wurde vergessen den Zähler `i` in jedem Durchgang zu erhöhen. Die Bedingung ist auf alle Zeit `true`.
Je nachdem wie anspruchsvoll der Code innerhalb der Schleife ist, verbraucht der Server Unmengen an Speicher und irgendwann ist dieser verbraucht.
Das Ergebnis: Der Server stürzt unkontrolliert ab und reisst aktive Sessions unwiederbringlich mit in den Abgrund.
Das Problem: Nicht nur Ihre Anwendung steht, sondern der gesamte Server verweigert seinen Dienst.

2. Die Lösung ist ein aktive Speicherüberwachung

Solche Endlosschleifen kann man nie verhindern. Die Folgen solcher Fehlprogrammierungen können aber gemildert werden. Wichtiger noch ist, dass auch Lastspitzen einen Server mehr belasten können als Sie geplant haben. Innerhalb einer JVM ist der Status des Speicherverbrauchs des Servers jederzeit abfragbar.
Der MemoryWatcher kontrolliert in Intervallen den noch freien Speicher. Fällt der freie - noch verfügbare - Speicher unter eine Grenze, wird zuerst der GarbageCollector aufgerufen und - wenn der Speicher unter eine weitere Grenze fällt - der Server kontrolliert neu gestartet. Requests werden dadurch normal beendet und Sessions können - soweit es so konfiguriert ist - gespeichert werden.



MemoryWatcher eine Übersicht

Klicken Sie auf das Bild für die Vollansicht

3. Funktionsweise

Der MemoryWatcher ist als Server-Listener realisiert, d.h. er wird beim Start der Centaurus-Platform mitgestartet und beim Beenden mit heruntergefahren. Darüberhinaus ist er auch ein MBean und kann damit über eine JMX-Konsole (MX4J und MC4J) kontrolliert werden.

Der MemoryWatcher hat einige Parameter, die Sie an Ihre Bedürfnisse anpassen können. Dies passiert in der `${centaurus.base}/conf/server.xml`.
Suchen Sie dort den entsprechenden Listenereintrag:

```
<Listener
className="de.planetes.centaurus.plugins.memory.MemoryWatcherLifecycleLis
... />
```

Dort können Sie folgende Parameter konfigurieren:

checkInterval	Das Intervall, in dem der MemoryWatcher den Speicher überprüfen soll. Die Angabe erfolgt in Millisekunden (**Standard: 10000**).
gcRate	Ab wieviel Prozent freier Speicher soll der GarbageCollector aufgerufen werden (**Standard: 40**)?

restartRate	Ab wieviel Prozent freier Speicher soll der Server neu gestartet werden (**Standard: 20**)? Wählen Sie den Wert nicht zu klein, da auch der Shutdown noch zusätzlichen Speicher braucht.
permSize	Der permanente Speicher der JVM (**Standard: 64**).
debug	Sollen die Speicherwerte angezeigt werden, dann setzen Sie diesen Parameter auf 1, ansonsten auf **0**

Table 1:

4. Spaß mit dem Java GC

Java GC Tunnig	Tuning Garbage Collection with the 1.4.2 Java[tm] Virtual Machine

Die RMIRegisty ruft den GC alle 60 Sekunden	Setzen Sie dafür in der conf/wrapper.xml die entsprechenden JVM Parameter, um die zusätzlichen GC Aufrufe zu verhindern.
	`<property name="wrapper.java.additional.9" value="-Dsun.rmi.dgc.client.gcInterval=3600000"><`
	`<property name="wrapper.java.additional.10" value="-Dsun.rmi.dgc.server.gcInterval=3600000"><`
	Dies Einstellungen werden benötigt, wenn Sie den MX4J RMI Adaptor (jrmp) aktivieren oder ein Backend System via RMI ansprechen.

Deaktivierung des Full GC	Mit dieser Einstellung verhindern Sie das der GC Aufuf die gesamte JVM für Sekunden lahm legt. IN der Praxis kommt es dann manchmal dazu, dass nicht alle Objekte abgeräumt werden.. `<property name="wrapper.java.additional.11" value="-XX:+DisableExplicitGC"></property>`

Table 1: