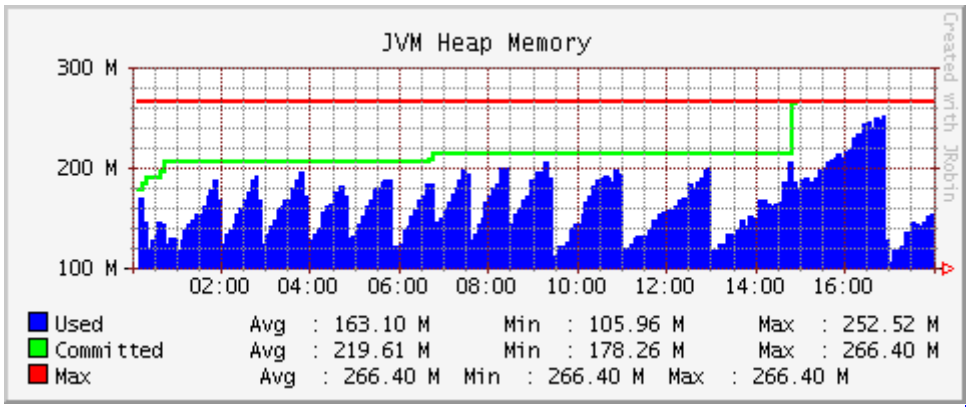
Monitoring the JVM with SNMP

By Roger Keays, 21 January 2007

Since Java 1.5, Sun's JVM has included an SNMP agent which is quite handy for keeping an eye on your Java apps using your existing monitoring toolset. Here's how to set up OpenNMS to monitor an app server and produce pretty graphs such as the one below, alongside your other SNMP collected metrics like CPU load and memory usage.



1. Install OpenNMS

First you need to install OpenNMS [1] onto the machine which is going to do the monitoring. OpenNMS is not difficult to install if you've had any experience with maven and Tomcat. Their website covers installation fairly accurately, so I'll skip this step and go straight into how to setup the JVM.

2. Enable Java SNMP

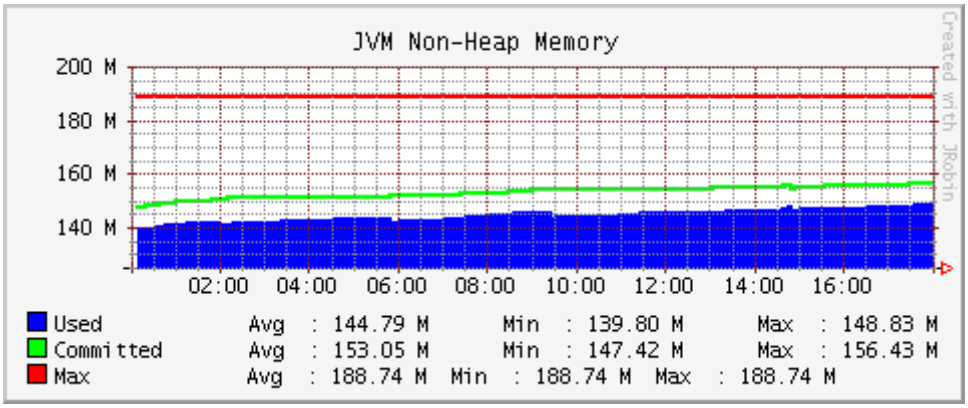
Enabling the SNMP agent on the JVM is pretty simple [2], and can be done adding either -Dcom.sun.management.snmp.port=161 or -Dcom.sun.management.config.file=snmp.properties to the java command line. Using the first method gives you an SNMP agent with all the defaults, which includes only listening on the loopback interface. Using the second method allows you to specify a few more properties in the snmp.properties file (or any other file of your choice).

For this task, I wanted SNMP on port 1161 since the app server runs as an underprivileged user and I already had another agent on port 161. I also needed to listen on all interfaces because the machine is being monitored remotely. Finally, since my firewall keeps out unwanted guests, I've also disabled the access-control list. The complete snmp.properties file looks like this:

```
com.sun.management.snmp.interface=0.0.0.0
com.sun.management.snmp.port=1161
com.sun.management.snmp.acl=false
```

Test your configuration by trying to query the Java SNMP agent first from the localhost, then from the monitoring machine:

```
$ snmpwalk -c public -v2c java.example.com:1161 .1.3.6.1.4.1.42.2.145.3.163.1.1
SNMPv2-SMI::enterprises.42.2.145.3.163.1.1.4.1.0 = STRING: "pid@host"
```



3. Configure OpenNMS

Next, we need to configure OpenNMS, which is a little more involved. Conceptually, there are four steps:

1. Configure the capabilities daemon to look for the snmp-jvm service on your chosen port (capsd-configuration.xml).
2. Configure the collect daemon to fetch data for this service (collectd-configuration.xml).
3. Configure the OIDs for the data you want to collect (datacollection-config.xml).
4. Configure the snmp graphs to display the collected values (snmp-graph.properties).

capsd configuration

This tells OpenNMS to look for the SNMP agent you have set up. Add the following configuration to capsd-configuration.xml, restart OpenNMS and rescan the host's services using the web interface. You should see a new service 'SNMP-JVM' on the host.

```
<protocol-plugin protocol="SNMP-JVM" class-name="org.opennms.netmgt.capsd.
                 scan="on" user-defined="false">
    <property key="timeout" value="5000" />
    <property key="retry" value="3" />
    <property key="port" value="1161" />
    <property key="vbname" value=".1.3.6.1.4.1.42.2.145.3.163.1.1.4.1" />
</protocol-plugin>
```

collectd configuration

To tell OpenNMS to actually collect the data from this service, you have to add the following to the collectd-configuration.xml file.

```
<service name="SNMP-JVM" interval="300000" user-defined="false" status="on
    <parameter key="collection" value="jvm"/>
    <parameter key="port" value="1161"/>
    <parameter key="retry" value="3"/>
    <parameter key="timeout" value="3000"/>
    <parameter key="oid" value=".1.3.6.1.4.1.42.2.145.3.163.1.1.4.1"/>
</service>
```

OID configuration

You also need to decide what data you want to monitor by reading the Java MIB [3]. Here are the values I chose to monitor:

Variable	OID
jvmMemoryHeapUsed	1.3.6.1.4.1.42.2.145.3.163.1.1.2.11
jvmMemoryHeapCommitted	1.3.6.1.4.1.42.2.145.3.163.1.1.2.12
jvmMemoryHeapMaxSize	1.3.6.1.4.1.42.2.145.3.163.1.1.2.13
jvmMemoryNonHeapUsed	1.3.6.1.4.1.42.2.145.3.163.1.1.2.21
jvmMemoryNonHeapCommited	1.3.6.1.4.1.42.2.145.3.163.1.1.2.22
jvmMemoryNonHeapMaxSize	1.3.6.1.4.1.42.2.145.3.163.1.1.2.23
jvmThreadCount	1.3.6.1.4.1.42.2.145.3.163.1.1.3.1

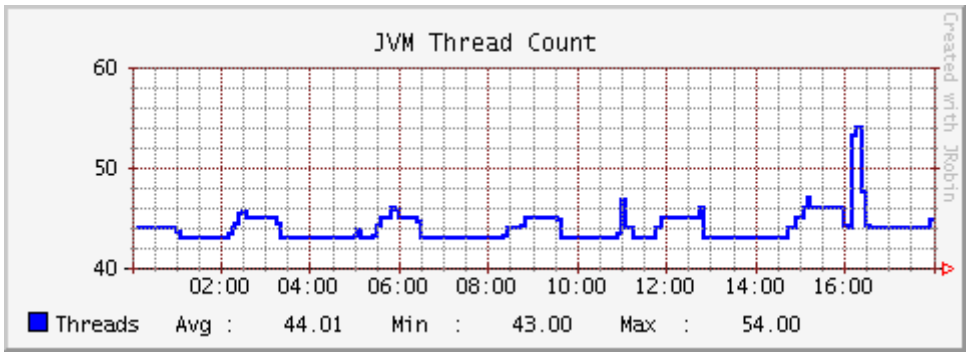
This information is added to datacollection-config.xml with the snippet below. An entire <snmp-collection/> needs to be created, because the jvm SNMP agent doesn't include a system table that might be used to distinguish it as a unique system in the default data collection.

```
<snmp-collection name="jvm" maxVarsPerPdu="10" snmpStorageFlag="primary">
  <rrd step="300">
    <rra>RRA:AVERAGE:0.5:1:8928</rra>
    <rra>RRA:AVERAGE:0.5:12:8784</rra>
    <rra>RRA:MIN:0.5:12:8784</rra>
    <rra>RRA:MAX:0.5:12:8784</rra>
  </rrd>

  <groups>
    <group name="jvm" ifType="all">
      <mibObj oid=".1.3.6.1.4.1.42.2.145.3.163.1.1.2.11" instance="0" alia
      <mibObj oid=".1.3.6.1.4.1.42.2.145.3.163.1.1.2.12" instance="0" alia
      <mibObj oid=".1.3.6.1.4.1.42.2.145.3.163.1.1.2.13" instance="0" alia
      <mibObj oid=".1.3.6.1.4.1.42.2.145.3.163.1.1.2.21" instance="0" alia
```

```
        <mibObj oid=".1.3.6.1.4.1.42.2.145.3.163.1.1.2.22" instance="0" alia
        <mibObj oid=".1.3.6.1.4.1.42.2.145.3.163.1.1.2.23" instance="0" alia
        <mibObj oid=".1.3.6.1.4.1.42.2.145.3.163.1.1.3.1" instance="0" alias
      </group>
    </groups>

    <systems>
      <systemDef name="JVM">
        <sysoidMask></sysoidMask>
        <collect>
          <includeGroup>jvm</includeGroup>
        </collect>
      </systemDef>
    </systems>
  </snmp-collection>
```

Now restart OpenNMS and look for the new RRD data files in share/rrd/**/jvm* to make sure the collection is working. Also check the collectd.log file for error messages.

Creating graphs



Phew! Almost there. If your RRD files are being created all you have to do is edit the snmp-graph.properties config file and reload the graphs page. Here's the configuration I used to create the graphs in this blog:

```
report.jvm.heap.name=JVM Heap Memory
report.jvm.heap.columns=jvmHeapUsed, jvmHeapCommitted, jvmHeapMax
report.jvm.heap.type=nodeSnmp
report.jvm.heap.command=--title="JVM Heap Memory" \
 DEF:used={rrd1}:jvmHeapUsed:AVERAGE \
 DEF:comm={rrd2}:jvmHeapCommitted:AVERAGE \
 DEF:max={rrd3}:jvmHeapMax:AVERAGE \
 AREA:used#0000ff:"Used     " \
 GPRINT:used:AVERAGE:" Avg  \\: %5.2lf %s" \
 GPRINT:used:MIN:"Min  \\: %5.2lf %s" \
```

```
 GPRINT:used:MAX:"Max  \\: %5.2lf %s\\n" \
 LINE2:comm#00ff00:"Committed" \
 GPRINT:comm:AVERAGE:" Avg  \\: %5.2lf %s" \
 GPRINT:comm:MIN:"Min  \\: %5.2lf %s" \
 GPRINT:comm:MAX:"Max  \\: %5.2lf %s\\n" \
 LINE2:max#ff0000:"Max           " \
 GPRINT:max:AVERAGE:" Avg  \\: %5.2lf %s" \
 GPRINT:max:MIN:"Min  \\: %5.2lf %s" \
 GPRINT:max:MAX:"Max  \\: %5.2lf %s\\n"

report.jvm.nonheap.name=JVM Non-Heap Memory
report.jvm.nonheap.columns=jvmNonHeapUsed, jvmNonHeapCommitted, jvmNonHeapMax
report.jvm.nonheap.type=nodeSnmp
report.jvm.nonheap.command=--title="JVM Non-Heap Memory" \
 DEF:used={rrd1}:jvmNonHeapUsed:AVERAGE \
 DEF:comm={rrd2}:jvmNonHeapCommitted:AVERAGE \
 DEF:max={rrd3}:jvmNonHeapMax:AVERAGE \
 AREA:used#0000ff:"Used     " \
 GPRINT:used:AVERAGE:" Avg  \\: %5.2lf %s" \
 GPRINT:used:MIN:"Min  \\: %5.2lf %s" \
 GPRINT:used:MAX:"Max  \\: %5.2lf %s\\n" \
 LINE2:comm#00ff00:"Committed" \
 GPRINT:comm:AVERAGE:" Avg  \\: %5.2lf %s" \
 GPRINT:comm:MIN:"Min  \\: %5.2lf %s" \
 GPRINT:comm:MAX:"Max  \\: %5.2lf %s\\n" \
 LINE2:max#ff0000:"Max          " \
 GPRINT:max:AVERAGE:" Avg  \\: %5.2lf %s" \
 GPRINT:max:MIN:"Min  \\: %5.2lf %s" \
 GPRINT:max:MAX:"Max  \\: %5.2lf %s\\n"

report.jvm.threads.name=JVM Threads
report.jvm.threads.columns=jvmThreadCount
report.jvm.threads.type=nodeSnmp
report.jvm.threads.command=--title="JVM Thread Count" \
 DEF:threads={rrd1}:jvmThreadCount:AVERAGE \
 LINE2:threads#0000ff:"Threads" \
 GPRINT:threads:AVERAGE:" Avg \\: %8.2lf %s" \
 GPRINT:threads:MIN:"Min  \\: %8.2lf %s" \
 GPRINT:threads:MAX:"Max  \\: %8.2lf %s\\n"
```

Add these report definitions towards the end of the file, and their names (jvm.heap, jvm.nonheap and jvm.threads) to the list at the top of the file to have them display on the graphs page.

JMX Monitoring

OpenNMS also does JMX monitoring, but this is likely to be the subject of another blog post. Unfortunately, AFAICT, it does not yet have support for CompositeTypes which means we can't (yet) collect heap and non-heap usage via the Memory managed bean.

References

[1] http://www.opennms.org

[2] http://java.sun.com/j2se/1.5.0/docs/guide/management/SNMP.html

[3] http://java.sun.com/j2se/1.5.0/docs/guide/management/JVM-MANAGEMENT-MIB.mib

About Roger Keays

Roger Keays is an artist, an engineer, and a student of life. He has no fixed addressand has leftfootprints on 40-something different countries around the world. Roger is addicted to surfing. His other interests are music, psychology, languages, the proper use of semicolons, and finding good food.