Profile an application on the SAP NetWeaver Composition Environment edition

This tutorial will show you the necessary steps to setup a profiling analysis of a Java application on a SAP NetWeaver Composition Environment. As an example we will use one of the applications delivered with the application server (the System Info application) and the Administrator user.

Prerequisites

- You have access to a SAP NetWeaver Composition Environment 7.3 trial edition system (or a newer version), either local or remote. The system is started.
- You know the password of the system's Administrator user.
- You have installed SAP NetWeaver Developer Studio provided with the SAP NetWeaver Composition Environment or any Eclipse IDE 3.3 or above.
- You installed the SAP JVM Profiler 2.0 plug-in in your NetWeaver Developer Studio or your Eclipse IDE.

Attach the Server JVM

If not already done, start the Eclipse environment and open the Profiling perspective. To accomplish this, press the button Open Perspective in the main toolbar (alternatively: Window Open Perspective Other ...) and choose Profiling in the list of available perspectives (depicted list may differ from yours):

Initially, the VM Explorer View should be opened (you may (re-)open it by pressing Show VM Explorer in the Profile View at any time). The VM Explorer is designed to give you a quick overview about all SAP JVMs that are running on your local system as well as on remote hosts.

- Local scenario

If the SAP NetWeaver system you want to profile runs on the same host as the profiler, the server instance should be listed in the VM Explorer automatically:
Remote scenario

Whereas local SAP JVMs can be investigated without any additional work, some steps has to be performed to monitor JVMs on remote hosts:

1. Start a `jvmmond` daemon on the host you want to register to the VM Explorer. You find the jvmmond command tool in the bin directory of the SAP JDK. If no specific port is chosen with option `-port`, jvmmond will try to use default RMI port 1099.

   **WARNING:** Please be aware that jvmmond opens a port in order to offer remote access. Depending on your network setup (firewalls etc.) this might be a security issue comparable to an open debugging port.

2. In the profiler on your local host now open the Hosts dialog by pressing the toolbar button `Manage Hosts...` (see screenshot below)

3. Add the hostname or IP address of the remote host and the port, jvmmond is listening to.

4. A successful registration is acknowledged with `Valid host`. 

Now the SAP JVM running your server instance (named LKG_00_server in this example) should be visible in the VM Explorer just as depicted in the screenshot above.

**Monitoring the JVM**

The VM Explorer lists all SAP JVMs that are visible, i.e. which are either local or which are visible through a registered jvmmond daemon on a remote host. By default, there are some table columns displaying basic attributes like the PID, CPU load, memory usage or the name of the host. This should help to easily identify the JVM you want to profile.

If you like to get some more information about a specific JVM, just select it from the explorer list and choose among the offered attribute groups represented by the tabs *General*, *System*, *Performance*, *Garbage Collection* and *JIT*. Dynamic values like the number of current threads or the Java Heap sizes are updated frequently. Please note that observing the JVMs in the explorer does not induce any overhead for these processes.

**Starting a Profiling Run**

Profiling the server JVM requires a so-called Profiling connection, because the profiling backend which resides in the JVM instance needs to send profiling data packets continuously to the profiling frontend (i.e. the Eclipse plug-in). Former Profiler releases established a special debugging connection to the target JVM to provide the basic communication infrastructure. This implied an open debug port and also could interfere with existing breakpoints. The current release still supports this kind of 1:1 - connections (press button *Connect Host via Debugging Port* ... in explorer's toolbar), but it's recommended to make usage of the new profiling connections which are based on shared memory communication (to be more precisely sockets defined on shared files). The advantage of this approach is that the target JVM doesn't need an open port anymore (but be aware that in remote scenarios jvmmond still need) and moreover there can't be any interferences with the debugging functionality. Also multiple profiling connections at the same time are possible.

To open a profiling connection to the target JVM just double-click it in the VM Explorer or choose *Profile...* from its context menu. As a result, the Connection View appears:
The Connection View represents the starting point for all different kinds of profiling analysis. Users activate one of the offered profiling traces for a specific period of time, the analysis period. All the profiling data gathered during such a collection period are stored in a so-called snapshot.

After the collection period a new trace of arbitrary type may be started. While collecting new profiling data, you may continue the analysis in parallel by investigating already retrieved snapshots (i.e. explore their statistics and graphs).

In this example we start a Performance Hotspot Analysis in order to get an impression of the server's runtime behaviour when the System Information is opened by a browser. A helpful profiler feature is to filter the profiling data a priori by the offered categories User, Session, Request or Application. This means instead of filtering the snapshot data in the profiler, the profiler backend only creates and sends profiling data that matches the filter, e.g. only the method calls that are triggered by a special user. This way the amount of profiling data may be shrunked substantially.

Before you start the Performance Hotspot Analysis, configure the profiling filter in the configuration dialog which appears when you click on the small black triangle attached to link for the Performance Hotspot Analysis (see screenshot above).

In the configuration dialog, set the User filter “Administrator” in order to track only methods calls that are triggered in the server for processing tasks that are related to the Administrator user:
The performance trace starts right after having pressed OK, which is illustrated in the connection view. Beside the analysis duration and the number of taken samples, the view also shows the proportion of interpreted methods. The simple rule is: the lower the percentage of interpreted frames, the better the quality of profiling results. Therefore a performance trace should be started only after the system has processed a warming-up phase where most of the Java methods have been compiled.

Now we put some load on the server which is tracked by the active performance trace. Open the start page of your system in an arbitrary browser and press the button for the System Information application. The browser will launch a new window for the System Information application. You should be prompted for a user name and password before the application starts.
Now open the Components Info tab and trigger some arbitrary operations:

After some clicks, you may stop the profiling run by pressing the disconnect button in the Connection View.

As a result you get the Snapshot Overview page where all taken snapshots are listed (obviously there is only the just created one):
Evaluating Snapshots

You are free to analyze the stored snapshots online or offline. The SAP JVM Profiler supports the analysis with sophisticated statistics and graphs that help to track down the resource consumption issues. In the snapshot overview page select the single snapshot named "Performance Hotspot Snapshot 1" in order to enter the corresponding snapshot view:

Depending on the snapshot several statistics are offered. In case of the example there are - amongst others - the Methods (Flat) and Methods (Hierarchical) statistics. Whereas the flat statistic simply displays the self and total running times of executes methods, the hierarchical statistic shows the values for all call paths in a tree-style manner:
Beginning from the root of the call hierarchy, you can follow the most CPU time consuming path and isolate possible performance bottlenecks.

Congratulations

You are now able to setup a profiling analysis for an application running on the SAP NetWeaver Composition Environment. Where to go from here? Explore the different statistics and make yourself familiar with the provided information. In the same way we did the application analysis you may also start an allocation analysis.

Of course there will be questions and it may be hard to quickly find resource bottlenecks. It may be helpful to check the SAP JVM Profiler’s documentation, which provides you some examples how to utilize the Profiler’s feature set to locate all the kinds or resource consumption problems.

Links

Java Profiling
What's New in SAP JVM Profiler 2.2
Features and Benefits
Detailed download instructions for SAP JVM Profiler
Download the SAP NetWeaver Composition Environment trial
View the SAP JVM Profiler documentation (PDF)
View the SAP JVM documentation
Provide feedback in our blog