Performance Tuning for SAP

In this tutorial I'm going to talk about performance tuning mainly on the web application, SAP have server side performance tool quite helpful, I also found Microsoft developer tool is quite useful on the browser side specially for BSP, WD on portal specially for dynamic programming for WD(A/J)

SAP server side performance tool

Transaction code STAD: analyse the system information like CPU time, processing time, database time, memory time.

Transaction code ST05: system->utility->performance trace activate the trace, after running the WDA, deactivate the trace. Trace list has a lot of information about the database and SQL statement, index

Transaction code SE30: examine the runtime analysis of any ABAP programs, such as reports, subroutines, function modules or classes, that you create in the ABAP Workbench

How to starting the runtime analysis:

Any screen: System -> Utilities ->Runtime Analysis -> Execute

This initial screen of ABAP workbench: Test -> Runtime Analysis

The initial screen of the ABAP Editor: Program -> Execute Runtime Analysis

The ABAP Editor: Utilities -> More Utilities -> Runtime Analysis

Here is the step of Using runtime analysis on Web dynpro ABAP:

1. In SE80 we select the required Web Dynpro ABAP application and display it (by double-clicking or by choosing Display).
2. Choose Goto HTTP Service Maintenance. You go to the service node in the SICF for your application.
3. Select the service node that belongs to your Web Dynpro application.
4. Choose Enter Runtime Analysis Activate.
5. Make the required settings and choose Activate. We recommend the setting for single entries as it is easier to trace the individual requests. To do this, under Processing change the default value Composite Entries to Single Entries.
6. Start your Web Dynpro application, make your user entries and execute the operations for which you would like to make a runtime analysis.
7. To display the results of the runtime analysis, in transaction SE30 or SAT under Performance Data File choose Other File... or choose tab Analyze.
8. You can display the overview and hit lists for the various performance data files and follow the individual steps.
9. Finish the runtime analysis in the SICF for the service node by choosing Edit Runtime Analysis

We can also create variant in SAT/SE30 depending the system, and run the variant in SICF and analysis the WDA.

Using memory inspector, you can get the information about how memory has be consumed during the runtime, this part is quite important for WDA.

Starting the memory inspector:

1. Call the transaction S_MEMORY_INSPECTOR
2. Choose the function Memory Analysis Compare Memory Snapshots in the ABAP Debug
3. Choose System ->Utilities ->Memory Analysis -> Compare Memory Snapshots in any transaction.

After starting the Memory Inspector, you can analyze memory snapshots on its initial screen. There are four ways of creating memory snapshots:

1. Call the function Development Memory Analysis Create Memory Snapshot in the ABAP Debugger.
2. Choose System ->utilities ->Memory Analysis ->Create Memory Snapshot in any transaction.
3. Call the static method WRITE_MEMORY_CONSUMPTION_FILE of the system class CL_ABAP_MEMORY_UTILITIES.
4. Enter /hmusa in the toolbar input field. You must be in a transaction to do this.

All options follow the same naming conventions for the files in which the memory snapshots are stored.

Using the Code Inspector (transaction code SCI), you can check individual objects or sets of objects for performance, security, syntax, and
adherence to name conventions. You can also determine statistical information or search for certain ABAP words (tokens). In the Code Inspector, you can define inspections that, with the help of check variants, examine certain sets of objects. As the result of an inspection, you receive information messages, warning messages, or error messages on different properties of the examined objects.

You can call the Code Inspector using transaction code SCI or through the menu path SAP Menu à Tools à ABAP Workbench à Test à Code Inspector. Also, you can call the Code Inspector from the following transactions:

- ABAP Dictionary (SE11) for DDIC tables
- Class Builder (SE24) for classes and interfaces
- Function Builder (SE37) for function groups
- ABAP Editor (SE38) for programs or reports
- ABAP Workbench (SE80)

**Client side performance tool**

I found Microsoft IE developer tool is quite useful tool to debug the BSP, WD, even portal page.

The Internet Explorer Developer Toolbar provides several features for exploring and understanding Web pages. These features enable you to:

- Select element by click: Explore and modify the document object model (DOM) of a Web page. Locate and select specific elements on a Web page through a variety of techniques.

Find: You can find all the property of UI element of WD or BSP, this is very useful for the dynamic program, you can identify the location, property by element, id, name class.

Here is the example of finding the property of UI element.

- **view:** View HTML object class names, ID's, and details such as link paths, tab index values, and access keys

Outline: Outline tables, table cells, images, or selected tags, you can also identify the position of UI element, it's float, fixed, absolute or relative

Images: Display image dimensions, file sizes, path information, and alternate (ALT) text. Immediately resize the browser window to a new resolution

Cache: Selectively clear the browser cache and saved cookies. Choose from all objects or those associated with a given domain, you can detect the cookie from server side and from the client side

Here is the sample of the cookie which has beed detected by IE developer tool on help.

- **https://weblogs.sdn.sap.com/weblogs/images/251978614/cookie.JPG** width=520 border=0>