ACTA TECHNICA CORVINIENSIS – Bulletin of Engineering
Tome V (Year 2012). FASCICULE 2 [April–June]. ISSN 2067-3809

[1.] Mihaela OSACI, [2.] Adela-Diana BERDIE, [3.] Ana-Daniela CRISTEA

CONTROL TOOLS FOR OPTIMIZING ABAP CODES

[1-2.] UNIVERSITY „POLITEHNICA" TIMISOARA, FACULTY ENGINEERING HUNEDOARA, ROMANIA
[3.] CELLENT AG STUTTGART, GERMANY

ABSTRACT: *Given current trends to computerized economic, integrated design development provides premises for a computer-based business processes in order to accommodate to current climate and increase competitiveness. This paper is a study on how to use control tools in ABAP SAP NetWeaver Application Server to perform the best code optimizing. In appropriate case studies, we will refer to the ABAP Runtime Analysis, Memory Analysis, and STAD transaction.*
KEYWORDS: *software integrated system, runtime analysis, memory analysis, business transaction analysis*

INTRODUCTION

SAP Netweaver Application Server ABAP is the SAP Netweaver Application Server from which applications can be programmed in ABAP [1], [2]. It provides the execution environment is a virtual machine ABAP to ABAP programs independent of hardware, operating system and database system. ABAP Workbench is used as the runtime environment for the development of applications. SAP Netweaver Application Server ABAP, client-server system 3-thier, can be used by people or other software. Users can access people using user interfaces which are available via a web browser (using ICM interface) or SAP GUI installed on desktop client computers. Software components can access the connection Remote Function Call protocol RCF (RCF) properly using RFC interface.

Native language of an ABAP system is a generation-4 programming language (4GL) that supports both procedural model and object oriented programming. Persistent data stored in relational databases and ABAP programs accessed through Open SQL instructions. Open SQL consists of the DML (Data Manipulation Language) standard SQL language.

CODE TESTING TOOL: ABAP OBJECTS RUNTIME ANALYSIS (SE30 TRANSACTION)

Running time is important information relative to the performance of code sequences [3]. The ABAP Runtime Analysis tool can analyze the execution time affecting the performance of ABAP software. The ABAP Runtime Analysis can test, for example, programs, methods and functional modules (global subroutines). Test results can be saved in graphic file server applications. These files are available to achieve the necessary optimizations. ABAP Runtime Analysis tool can be executed using transaction SE30 launch – Figure 1.

For example, we should test the YALLS_ARRAY_FETCH_DATA_REF software that uses array fetch reading technology in an internal table of data from a database table, access to data in internal table is made by reference variable time. We run the software from this window – Fig. 1. After returning from the execution window, the Runtime Analysis window turns as described in Figure 2 and it allows performance evaluation. Test program results are presented in Figure 3.

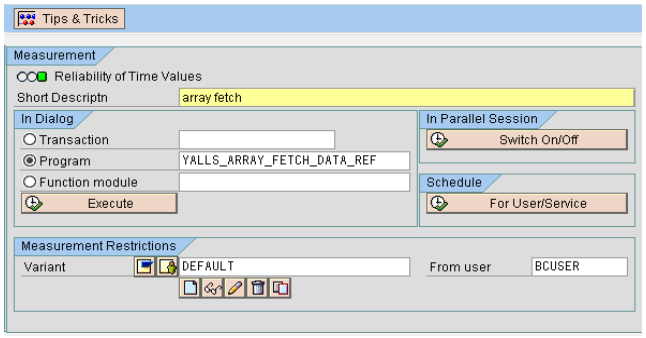

Figure 1. Main interface ABAP runtime analysis tool

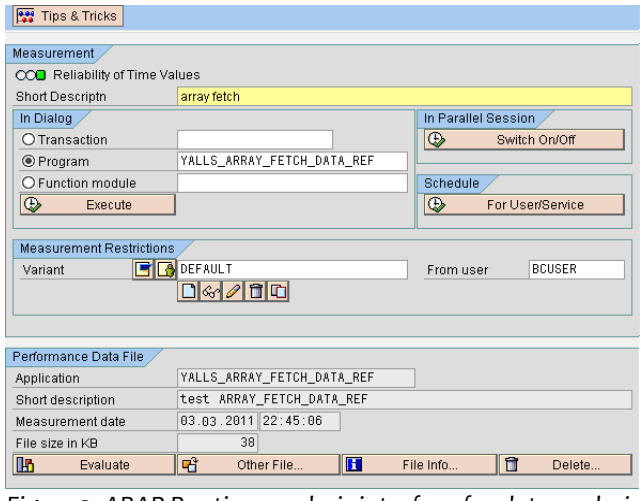

Figure 2. ABAP Runtime analysis interface for data analysis

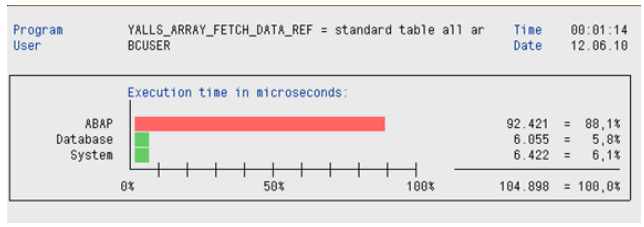

Figure 3. YALLS_ARRAY_FETCH_DATA_REF software test results

We can now read the ABAP instructions runtime time in microseconds, as well as the time to work with the database and runtime time operating system.

CODE TESTING TOOL: MEMORY ANALYSIS

An analysis of the software memory can be used by the Debugging process. We should use the Replace Tool from the Toolbar and choose to receive the Special Tool - Figure 4, and then we choose Memory Analysis option to analyze it in terms of memory used in the software code.

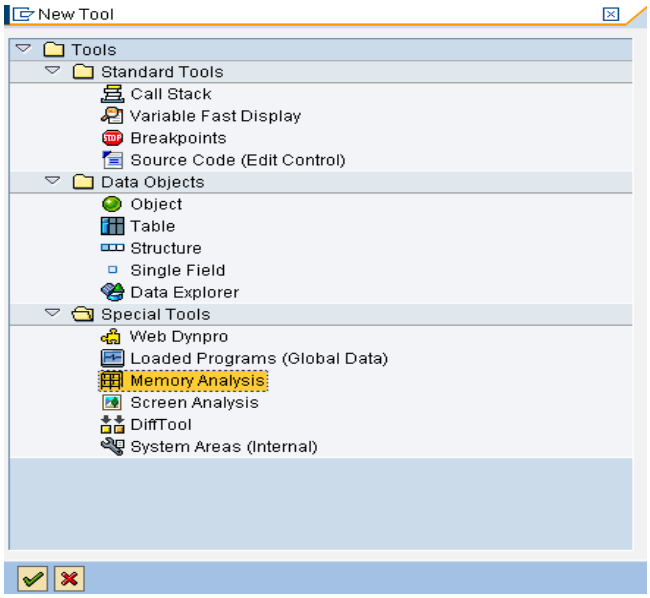

Figure 4. Memory analysis by debbuger

Figure 5 describes the memory test results of codes in YALLS_ARRAY_FETCH_DATA_REF software.

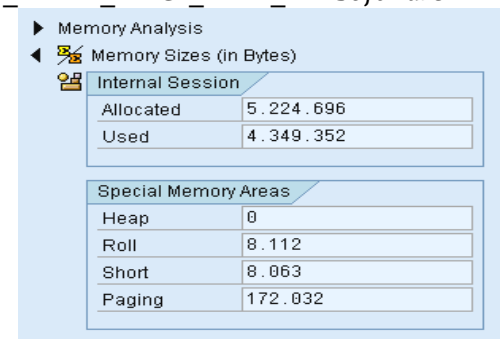

Figure 5. Memory analysis results

In Figure 5, memory size is specified in bytes. Special areas of memory are: Heap Memory -private memory reserved for runtime processes of the Application Server; Roll Memory is reserved for copying data sources such as roll file, if the runtime processes of application server change; Short memory - memory for storing intermediate results resulting from the use of such a screen is automatically cleaned after each dialog step.

STAD TRANSACTION (BUSINESS TRANSACTION ANALYSIS)

Business Transactions Analysis calculates the system resource usage of individual transactions for ABAP systems and provides a detailed analysis of a transaction and the dialog steps. The selection criteria include user, transaction, program, task type, start date, and start time. [1]

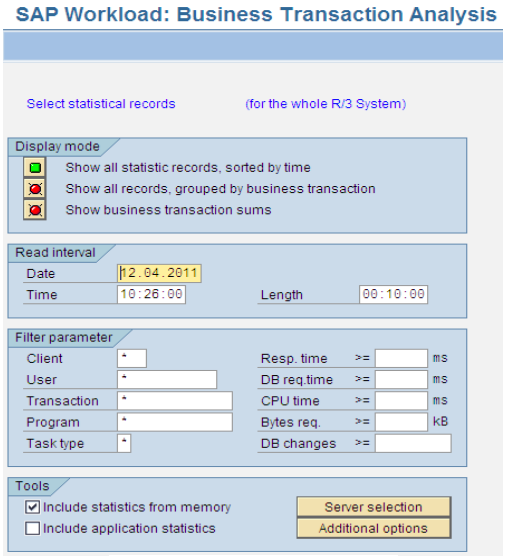

Figure 6. STAD transaction

This transaction enables us to set up the system through several variants - Figure 6: displaying single records as they are displayed in Transaction STAD, displaying the single records grouped by business transaction, or displaying business transaction or job totals.

The system always analyzes a time frame that is larger than the reading time, because the start date and time define the beginning of the period to be analyzed and the read time defines the duration of that period.

By choosing Include statistics from memory, the system analyzes statistical records that were not yet written to the statistic file, but which are contained in the statistics buffer. To analyze very recent time periods, is necessary to include the buffered records. With Server selection we can analyze only the statistics of specific servers. With Additional options we can influence the time frame mentioned above or the wait time for RFC's to be analyzed if we do not receive data from a server that was called as a result of RFC problems, or busy application servers that increase waiting time.

Testing it by STAD transactions for a transactional operation, the job name is SDOE_LOAD - Figure 7; the response time results, time in WP's, Waiting time, CPU time, DB req time, Memory used, and transfered kBytes.

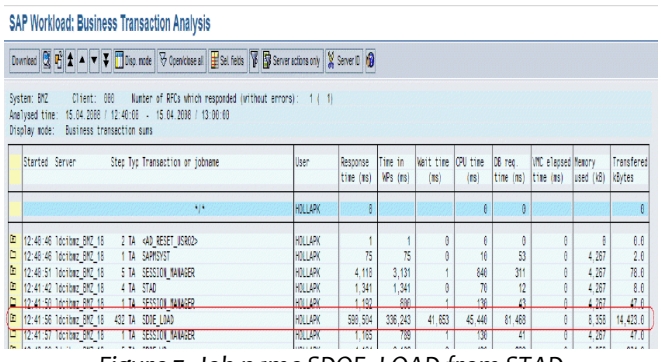

Figure 7. Job name SDOE_LOAD from STAD

RESULTS AND INTERPRETATION OF DATA PROCESSING EFFICIENCY INCREASE IN ABAP IN DATABASE TABLES

We will study various techniques [4], [5], [6] for processing data in a table. We suggest reading the records in a table according to selection criteria introduced by a screen with a single parameter selection and display records after processing a classic GU SAP list.

To read the data using two techniques: SELECT ... END SELECT cycle and ARRAY FETCH technique. SELECT ... END SELECT cycle requires sequential reading of information and attaching the internal table line by line. ARRAY FETCH technique involves reading the entire contents in an internal table. Data from internal table can be accessed in several ways: work area, field-symbol (label field) and variable time reference. The tests we perform at 10, 40, 70 and 100 table entries, watching the way ABAP instructions runtime, as well as runtime time of the database performance and total number of records in the table varies.

For the group of codes, we test two data reading techniques in database table, "select ... end select" and "array fetch"; and in case of the array fetch technique we look for effectiveness of three means of access to data from internal table: work area (a line-table type time object), every reference variable or field-symbols, and the results are described in Figure 8-10.

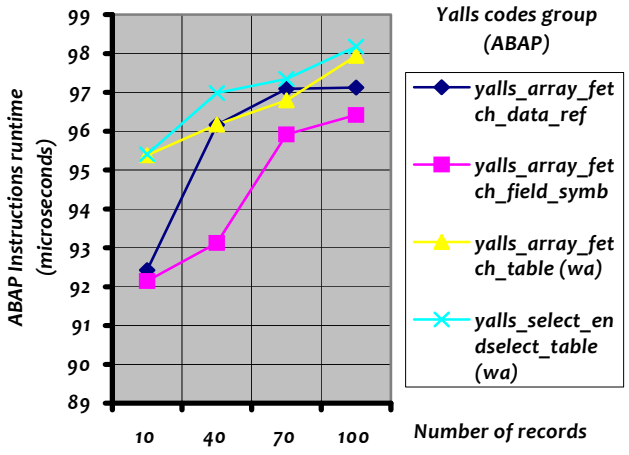

Figure 8. ABAP Runtime instructions depending on the number of records in the database table

In Figure 6, we describe the instructions execution of a lower ABAP when runtime with symbols for accessing field data from internal table. In this case, we use only one data object where we rewrite and process sequential values from the internal table. ABAP instructions runtime increases along with the number of records, because in all cases internal table reading and processing is done sequentially in a "loop ... end loop" cycle. The increase is more pronounced when using "select ... end select" technique that reads one line of the table and attaches it to the internal table. If case of array fetch technology data is read "globally".

In terms of database runtime, we can see - in Figure 7 - that it increases along with the number of records, which are almost similar in all three cases in which we used the array fetch technique, therefore the increase was more obvious when using "select ... end select" technique, with a slightly larger number of records.

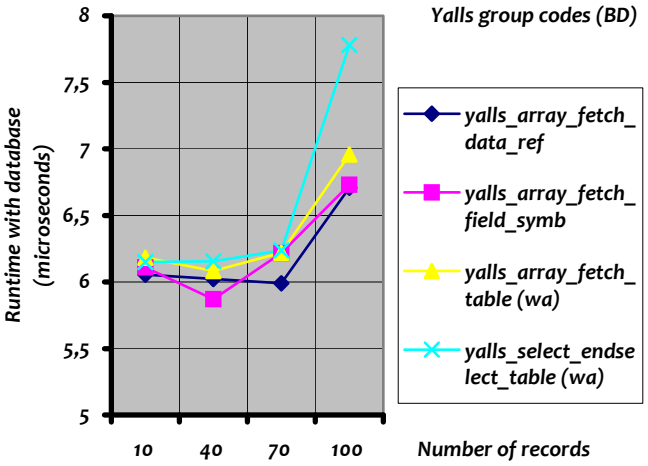

Figure 9. Runtime database records the number of database table

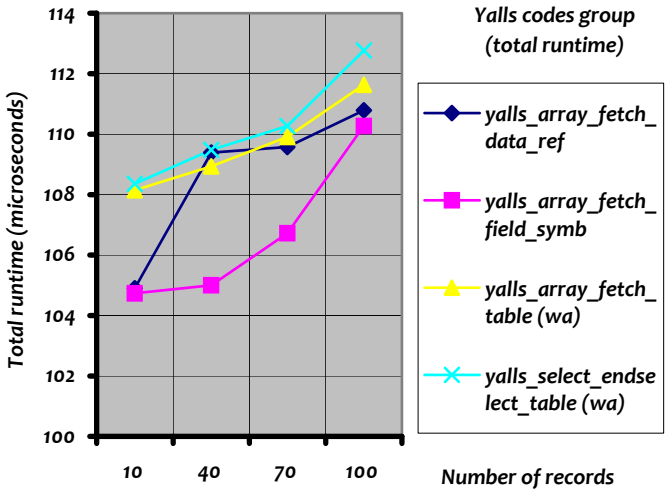

Figure 10. Total runtime of the software according to the number of records in the database table

Figure 8 reinforces the idea that using array fetch technology and accessing data from the internal table by the symbol field is a better solution over time. In terms of memory consumption, an analysis of codes from the yalls group revealed the results presented in Figures 11-13.

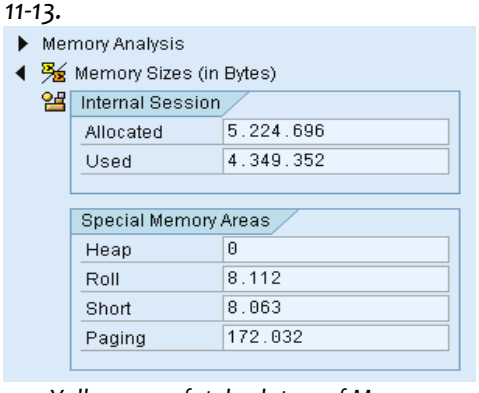

Figure 11. Yalls_array_fetch_data_ref Memory analysis

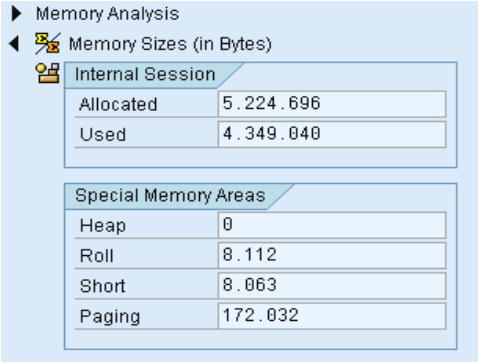

Figure 12. Yalls_array_fetch_field_symbols Memory analysis

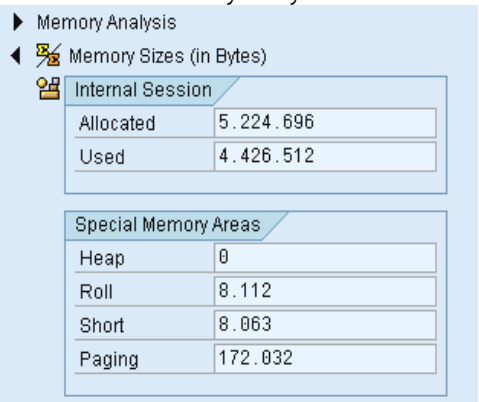

Figure 13. Yalls_array_fetch_table (wa) Memory analysis

And speaking of memory used, array fetch technique with accessing data from internal table with the symbol field is the most effective.

CONCLUSIONS

This paper collects several studies referring to the involvement of the control tools, such as ABAP Runtime Analysis and Memory Analysis in order to analyze all methods to optimize data processing codes of the level of persistence on SAP Netweaver 2004's, ABAP Application Server. The analysis performed showed that the most effective reading data technique from one data table is array fetch. As far as access to data in internal table is concerned, it is efficient to use the symbol field both in terms of processing time and memory used. These studies can be of great use for an efficient coding of this integrated system which is also an ERP system (Enterprise Resource Planning).

REFERENCES

[1.] *SAP Help Portal (http://help.sap.com),*
[2.] *SAP Developer Network (http://sdn.sap.com),*
[3.] *Horst Keller, Wolf Hagen Thummel, Official ABAP Programming Guidelines, Galileo Press, 2010*
[4.] *Horst Keller, Sascha Krüger, ABAP Objects, SAP Press, Bonn Germany, 2007*
[5.] *Ulrich Gellert, Ana Daniela Cristea, Web Dynpro ABAP for Practitioners, Springer Publishing House, 2010*
[6.] *Cristea Ana Daniela, Berdie Adela, Osaci Mihaela, Working With ABAP Persistent Data, ISIRR Hunedoara, 23-24 April, 2009, Proceedings Conference, CD Edition*

ACTA TECHNICA CORVINIENSIS – BULLETIN of ENGINEERING
ISSN: 2067-3809 [CD-Rom, online]