Capgemini

# ABAP Part III

## Lesson 8: Performance Tuning

# Lesson Objectives

After completing this lesson, participants will be able to understand performance tuning for -

- Selection Criteria
- Select Statements
- Internal Tables
- Typing
- Control Statements
- Field Conversion
- ABAP Objects

# Introduction

Performance Tuning is optimizing the performance of your application through various techniques thus increasing the productivity of the user.

Most systems respond to increased load with some degree of decreasing performance. A system's ability to accept higher load is called scalability, and modifying a system to handle a higher load is synonymous to performance tuning.

Systematic tuning follows these steps:

- Assess the problem and establish numeric values that categorize acceptable behavior.
- Measure the performance of the system before modification.
- Identify the part of the application that is critical for improving the performance. This is called the bottleneck.
- Modify that part of the system to remove the bottleneck.
- Measure the performance of the system after modification.

# Introduction

System architecture level factors to improve performance

- The R/3 System has a three-layer client/server architecture, with a Presentation layer, an Application layer and a Database layer.
- The presentation layer and the application layer are scalable.
- This means that if there is a hardware bottleneck, you can extend the system by adding more front ends and application servers.
- The database layer, as the central data repository, is not scalable.

# Introduction

ABAP performance analysis areas of focus

- One goal is to reduce the run time of programs on the application server, thereby reducing the CPU load.
- Another goal is to reduce the database load. Reducing the database load is particularly important since the database software is not scalable.
- An efficient program is one which delivers the required output to the user in a finite time as per the complexity of the program, rather than hearing the comment “I put the program to run , have my lunch and come back to check the results”.

# Introduction

A performance optimized ABAP Program has the following uses:

- 1.Saves the time of the end user.
- 2.Increases the productivity of the user.
- 3.In turn keeps the user and hence the management happy.

# General Definitions and Terms Involved

Response time: Time from the receipt of a user request to the sending of a response ( measured on the application server; does not include network time between the presentation server and the application server).

Dispatcher wait time: Time spent by the user request in the dispatcher queue.

Roll-in: Time required to roll the user context in to the R/3 work process.

Load time: Time required for loading and generating R/3 Repository or ABAP Dictionary objects.

Processing time: is response time - dispatcher wait time - roll-in - roll-out - load time - database time Enqueue time - roll wait time

# General Definitions and Terms Involved

Enqueue time: Time from sending an Enqueue request to the R/3 Enqueue server to the receipt of the results

Database time: Time from sending an SQL statement to the receipt of the results (measured on the application server; includes network time between the application server and the database server).

Roll wait time: Time in which the user context is rolled out of the work process pending response to an RFC call.

Roll-out: Time required to roll the user context in to the roll buffer.

CPU time: Time spent by the CPU in processing the transaction step (measured by the operating system; not an additive component of the response time).

# Selection Criteria

1. Restrict the data to the selection criteria itself, rather than filtering it out using the ABAP code or CHECK statement.
2. Select with selection list.

- Note: It is suggestible to make at least one field mandatory in Selection-Screen as mandatory fields restrict the data selection and hence increasing the performance.

# Select Statements

1.Avoid nested SELECT Statements.

2.Select all the records in a single shot using into table clause of select statement rather than to use Append statements.

3.When a base table has multiple indices, the where clause should be in the order of the index, either a primary or a secondary index.

4.For testing existence , use Select.. Up to 1 rows statement instead of a SELECT – ENDSELECT – LOOP with an Exit.

5.Use Select Single if all primary key fields are supplied in the Where condition.

6.Use column updates instead of single-row updates to update your database tables.

7.For all frequently used Select statements, try to use an index.

8.Using buffered tables improves the performance considerably

# Select Statements

Aggregate Functions

- 1.If you want to find the maximum, minimum, sum and average value or the count of a database column, use a select list with aggregate function instead of computing the aggregates yourself.
- 2.Some of the Aggregate functions allowed in SAP are MAX, MIN, AVG, SUM, COUNT, COUNT( * ).
- 3.Consider the following extract.

```
MAXNO = 0.
SELECT * FROM ZFLIGHT WHERE AIRLN = 'LF' AND CNTRY = 'IN'.
CHECK ZFLIGHT-FLIGH > MAXNO.
MAXNO = ZFLIGHT-FLIGH.
ENDSELECT.
```

- 4.The above mentioned code can be much more optimized by using the following code.
  - Select MAX( FLIGH ) from ZFLIGHT into MAXNO WHERE AIRLN = 'LF' AND CNTRY = 'IN'.

# Select Statements

FOR ALL ENTRIES

The for all entries creates a where clause, where all the entries in the driver table are combined with OR. If the number of entries in the driver table is larger than RSDB/MAX_BLOCKING_FACTOR, several similar SQL statements are executed to limit the length of the WHERE clause.

The plus (advantages)

- Large amount of data
- Mixing processing and reading of data
- Fast internal reprocessing of data
- Fast

The Minus (disadvantages)

- Difficult to program/understand
- Memory could be critical (use FREE or PACKAGE size)

# Select Statements

Points to be must considered FOR ALL ENTRIES

- Check that data is present in the driver table.
- Sorting the driver table and Removing duplicates from the driver table.
- Consider the following piece of extract

```
LOOP AT INT_CNTRY.
SELECT SINGLE * FROM ZFLIGH INTO INT_FLIGH
WHERE CNTRY = INT_CNTRY-CNTRY.
APPEND INT_FLIGH.
ENDLOOP.
```

.

# Select Statements

- The above mentioned can be more optimized by using the following code.

```
SORT INT_CNTRY BY CNTRY.
DELETE ADJACENT DUPLICATES FROM INT_CNTRY.
IF NOT INT_CNTRY[] IS INITIAL.
SELECT * FROM ZFLIGH APPENDING TABLE INT_FLIGH
FOR ALL ENTRIES IN INT_CNTRY
WHERE CNTRY = INT_CNTRY-CNTRY.
ENDIF.
```

# Select Statements

Select over more than one internal table

- Its better to use a views instead of nested Select statements.
- To read data from several logically connected tables use a join instead of nested Select statements. Joins are preferred only if all the primary key are available in WHERE clause for the tables that are joined. If the primary keys are not provided in join the Joining of tables itself takes time.
- Instead of using nested Select loops it is often better to use subqueries.

# Internal Tables

Table operations should be done using explicit work areas rather than via header lines.

Always try to use binary search instead of linear search. But don’t forget to sort your internal table before that.

A dynamic key access is slower than a static one, since the key specification must be evaluated at runtime.

A binary search using secondary index takes considerably less time.

LOOP ... WHERE is faster than LOOP/CHECK because LOOP ... WHERE evaluates the specified condition internally.

Modifying selected components using “ MODIFY itab …TRANSPORTING f1 f2.. “ accelerates the task of updating a line of an internal table.

# Internal Tables

Accessing the table entries directly in a "LOOP ... ASSIGNING ..." accelerates the task of updating a set of lines of an internal table considerably

If collect semantics is required, it is always better to use to COLLECT rather than READ BINARY and then ADD.

"APPEND LINES OF itab1 TO itab2" accelerates the task of appending a table to another table considerably as compared to " LOOP-APPEND-ENDLOOP."

"DELETE ADJACENT DUPLICATES" accelerates the task of deleting duplicate entries considerably as compared to " READ-LOOP-DELETE-ENDLOOP".

"DELETE itab FROM ... TO ..." accelerates the task of deleting a sequence of lines considerably as compared to " DO -DELETE-ENDDO".

# Internal Tables

Copying internal tables by using “ITAB2[ ] = ITAB1[ ]” as compared to “LOOP-APPEND-ENDLOOP”.

Specify the sort key as restrictively as possible to run the program faster.

For single read access hashed tables are more optimized as compared to sorted tables.

For partial sequential access sorted tables are more optimized as compared to hashed tables.

# Typing

Typed Parameters: Specifying the type for formal parameters in the source code, optimizes the code more thoroughly

Typed Field-Symbols: Specifying the type for formal parameters in the source code, optimizes the code more thoroughly

# Control Statements

CASE statements are precise and a little faster than IF-constructions.

Use of WHILE instead of a DO+EXIT-construction, is faster to execute.

# Field Conversion

Use fields of type I instead of P for typical integral variables like indices.

Use numeric literals or named constants with a number type instead of character strings if you are dealing with type-I or integral type-P fields.

Use properly typed constants instead of literals.

Use number types for arithmetic. Use type-N fields only for pure digit strings that are not intended for calculations e.g., Telephone nos etc.

Don't mix types unless absolutely necessary.

String operations can be made faster by specifying length of character field rather than defining as string

# ABAP Objects

Calling methods of global classes is faster than calling function modules

Calling global methods is slower as compared to calling local methods

**Point # 1**

“call method CL_PERFORMANCE_TEST=>M1.” is faster than “call function 'FUNCTION1'.”

**Point # 2**

“call method C1=>M1.” is faster than “call method CL_PERFORMANCE_TEST=>M1.”

# Tools for Performance Analysis

1. SCI - Code Inspector
2. ST05 – Performance Trace
3. SE30 ABAP Runtime Analysis
4. Extended Program Check (EPC)

# Tools for Performance Analysis

SCI – Code Inspector

- Static analysis of the quality of the code. Scans your program code and reports on potential problems particularly in the area of performance and security.

Benefits:

- Analysis of SQL WHERE clause of SELECT, UPDATE and DELETE statements.
- Identifies select statements that do not handle SY-SUBRC return codes
- Use of statements that infer ominous database access
- Identifies statements that bypass buffering
- Drill through support from results screen to program source statement.

Disadvantages:

- Static checks cannot identify the relevancy of a piece of code
- Cannot determine the frequency of execution of a statement of code.

# Tools for Performance Analysis

Code Inspector - Standard Checks Performed

Syntax checks and program generation

- 1. Normal ABAP syntax check.
- 2. Extended Program Check.
- 3. Program Generation.

Security checks

- 1. Use of statements deemed critical.
- 2. Use of statements the infers ominous database access.
- 3. Selected statements that do not handle system return code.

# Tools for Performance Analysis

Code Inspector - Standard Checks Performed ….continued

Performance checks

- 1. Select statements that implicitly bypass SAP table buffers.
- 2. Check statements inside of SELETCT…ENDSELECT loops.
- 3. Nested loops over internal tables and nested SELECT statements.
- 4. Analysis of WHERE clause to determine support database indices.

Search operations

- 1. Search for single tokens.
- 2. Search for complete statements.

# Demo: Code Inspector

# Tools for Performance Analysis

## ST05 Performance Trace

- The Performance Trace allows you to record database access, locking activities, and remote calls of reports and transactions in a trace file and to display the performance log as a list. It also provides extensive support for analyzing individual trace records.

## Benefits:

- Comprehensive tools set for evaluation of SQL statements.
- Integrated tools for accessing DDIC information.

## Disadvantages:

- Dynamic analysis requires equivalent production data content.
- Limited to database access analysis, no evaluation of code path execution.

# Demo: Performance Trace

# Tools for Performance Analysis

## SE30 ABAP Runtime Analysis

- The runtime analysis provides an overview of the duration and performance of your source code, from individual statements up to complete transactions.

## Benefits:

- Quickly identifies the percentage of time spent in database in contrast to ABAP code execution.
- Evaluation of modularization units, modules, performs, functions, etc.
- Analyze internal table operations

## Disadvantage:

- Dynamic analysis requires equivalent production data content.
- Lacks the integration of comprehensive SQL analysis tools.

# Demo: RunTime Analysis

# Tools for Performance Analysis

## Extended Program Check (EPC)

- The extended program check performs a complete check that includes the interfaces of external procedures called from your program, for example, checking whether the number and type of the interface parameters in an external procedure call is correct.
- The extended program check is also only a static check. It cannot eliminate all of the circumstances that could lead to exception situations or runtime errors. For example, any statements in which you specify arguments dynamically as the contents of fields, or in which you call procedures dynamically, cannot be checked statically.

# Tools for Performance Analysis

Extended Program Check (EPC)

- 1. Test environment - verify if the program is active. The Extended Check verify only active programs.
- 2. Call Function interfaces - verify if the parameters' type is correct, the using of exceptions.
- 3. Obsolete statements - Verify obsolete statements...
- 4. Character strings - Verify if the used text pool was created
- 5. Problematic Semantics - It shows to you better codes, like "WRITE TO can be replaced by more efficient MOVE TO"
- 6. Syntax check warnings - Pay attention to use conversion functions to compare fields. Or it will be listed in syntax check warnings' topic... and select statements are verified too.
- 7. Message - verify the number of with fields of message number

# Demo: Extended Program Check

# SQL Statement Evaluation Checklist

Is there any SELECT * statements in use?

- Change them to SELECT COL1,COL2,COL3 specifying columns

Are any CHECK statements embedded in SELECT..ENDSELECTs

- Incorporate the check statement logic in WHERE clause

Do SELECTs use appropriate DB Index or is table buffered?

- Change logic, create an index, or buffer table

Is nested SELECTs being used to retrieve data?

- Convert to DB join, view or SELECT FOR ALL ENTRIES IN ITAB

Are there SELECTs without WHERE clauses, on tables that grow?

- You need to redesign the solution.

# SQL Statement Evaluation Checklist

Are SELECTs to master data tables buffered?

- Store master data in itab and use READ TABLE…BINARY SEARCH to eliminate duplicate access with same key

Is SELECT…APPEND ITAB…ENDSELECT being used?

- Change processing to read data immediately into ITAB

SELECT ORDER BY statements being used?

- Read data to ITAB and then sort, unless DB Index supports order by.

Is program using calculations and summations that can be done on the database via SUM, AVG, MIN or MAX functions of SELECT?

- Use the calculations available on the SELECT statement.

Are ITABs processed using READ TABLE itab WITH KEY?

- Change table accesses to use BINARY SEARCH method.

# SQL Statement Evaluation Checklist

Tips for Running Trace

- 1. Limit the duration of the trace to 10 minutes or less.
- 2. Preload the program once to initialize database and cursor buffers.
- 3. To access trace results later.
  - Save the trace or..
  - Record stop and start time of trace
- 4. Utilize trace SQL statement summary function.
- 5. 5,000ms access per execution, general rule of thumb.
- 6. Make note of the application server of the job you are tracing.
- 7. Only ONE trace can be active on an application server.

# Summary

In this lesson, you have learnt performance tuning for -

- Selection Criteria
- Select Statements
- Internal Tables
- Typing
- Control Statements
- Field Conversion
- ABAP Objects

# Review Question

Question 1: It is better to use Linear search instead of Binary search in Internal Tables.

- True/False

Question 2: It is better to use select * instead of selecting a few fields.

- True/False