American Journal of Computer Architecture 2024, 11(4): 53-58
DOI: 10.5923/j.ajca.20241104.04

Implementation of Gather Stats in ETL Informatica Workflows for Oracle Database Performance Tuning

Malik Syed

Systems Integration Consulting, CGI, Pittsburgh, USA

Abstract This article illustrates the role, significance and benefits of gathering statistics in Oracle database and also helps us to learn how to use them effectively for database performance tuning. It also provides brief information on available alternate solutions for performance tuning. It explains how, why, when, and how often to gather statistics on database components like tables, columns, indexes etc., This article helps to understand what gathering statistics is and teaches how to implement and use the stats effectively for performance optimization with an example walk thru of a use case. The technical section of the article summarizes the required skills and lays out the detailed design steps from planning to implementation and usage of statistics collected via a stored procedure executed from ETL Informatica workflows that load data in the target Oracle tables. Finally, the article concludes summarizing the importance, potential benefits, results, and success stories of gathering statistics.

Keywords Gather Stats, Database, Oracle, SQL, ETL, Informatica, PowerCenter, Performance Tuning, Optimization, Workflow Manager, Stored Procedure

1. Introduction

1.1. Overview

Performance tuning is very essential and plays a crucial role when dealing with any application database. For any and every database, it is very essential to examine all the possible plans for a SQL statement to pick the one with lowest estimated resource usage and this is the job done by optimizer. For an optimizer to accurately determine the cost for an execution plan it needs information about all database objects like schema, tables, columns, indexes etc., accessed in the SQL statement along with the system information on which it runs. This necessary information is commonly referred as optimizer statistics.

Understanding, implementing, analysing, and managing optimizer statistics is key to improve SQL statement execution. It is also equally important to know when and how to gather statistics in a timely manner for effective performance tuning.

1.2. Problem Statement

Loading and managing huge data daily is very critical for every organization and especially when the data grows exponentially over the time. If the data is not organized and managed effectively in database, it would have negative impacts on data retrieval resulting in application performance downgrade. Gathering statistics on a timely manner is a solution to overcome the database performance issues. We will walk thru a scenario in detail on how an organization dealt to overcome the gradual database performance issues using gather stats solution.

Due to everyday data load for hundreds of tables using ETL Informatica jobs and with no effective statistics collection in place, the gradual exponential growth of data resulted in failure of application performance and due to this the technical teams were unable to proceed with deployment of any new business requirements or enhancements. This became a major blocker and the impact was huge. Technology team had to spend extra time to find the root cause and a solution to mitigate this issue.

2. Database Performance Tuning Solutions

2.1. Alternate Solutions

Besides gathering statistics, there are a few alternate solutions to improve any database performance. Below are a few options to list:

- **Caching:** Storing frequently accessed data in memory is referred as caching and this helps to improve database performance. Oracle Coherence or SQL Server In-Memory OLTP to cache data are the tools used for caching.

* Corresponding author:
syed.mallik@gmail.com (Malik Syed)
Received: Oct. 22, 2024; Accepted: Nov. 9, 2024; Published: Nov. 12, 2024
Published online at http://journal.sapub.org/ajca

- **Hardware upgrades:** Upgrading hardware components like CPU, memory, and storage helps to improve database performance.
- **Indexing:** Adding indexes to the most important and frequently used columns in data lookup from tables is called as indexing and it helps to improve database performance by allowing the database to quickly locate data. Tools like Oracle Enterprise Manager or SQL Server Management Studio are used to create indexes.
- **SQL queries optimization:** Poorly written SQL queries can lead to slow database performance and optimizing SQL queries helps to improve database performance. Tools like Oracle SQL Developer or SQL Server Management Studio are used to optimize SQL queries.
- **Partitioning:** Dividing large tables into smaller and more manageable pieces is called partitioning which helps to improve performance. Tools like Oracle Enterprise Manager or SQL Server Management Studio are used for tables partitioning.

2.2. When to Choose Gather Stats Solution

Gathering statistics can be selected as a solution under the circumstances when:

- Database gets refreshed every day making caching method ineffective.
- Frequent hardware upgrades are not very feasible due to high infrastructure maintenance costs.
- Indexing is already in place.
- Queries are already optimized.
- Database partitioning is already in place.

3. Deep Dive into Gather Stats

3.1. When to Gather Statistics

In order to select an optimal execution plan, the optimizer must have representative statistics. Representative statistics are not necessarily up to the minute statistics but a set of statistics that help the optimizer to determine the correct number of rows it should expect from each operation in the execution plan. We can either make sure of automatic statistics gathering task offered by Oracle or we can even go for manual statistics collection.

3.2. When Not to Gather Statistics

Although the optimizer needs accurate statistics to select an optimal plan, there are scenarios as explained below where gathering statistics can be difficult, too costly, or cannot be accomplished in a timely manner and an alternative strategy is required.

Volatile Tables: A volatile table is one where the volume of data changes dramatically over time. For example, an orders queue table, which at the start of the day the table is empty. As the day progresses and orders are placed the table begins to fill up. Once each order is processed it is deleted from the tables, so by the end of the day the table is empty again. If you relied on the automatic statistics gather job to maintain statistics on such tables, then the statistics would always show the table was empty because it was empty over night when the job ran. However, during the day the table may have hundreds of thousands of rows in it. In such cases it is better to gather a representative set of statistics for the table during the day when the table is populated and then lock them. Locking the statistics will prevent the automated statistics gathering task from overwriting them. Alternatively, you could rely on dynamic sampling to gather statistics on these tables. The optimizer uses dynamic sampling during the compilation of a SQL statement to gather basic statistics on the tables before optimizing the statement. Although the statistics gathered by dynamic sampling are not as high a quality or as complete as the statistics gathered using the DBMS_STATS package, they should be good enough in most cases.

Global Temporary Tables: Global temporary tables are often used to store intermediate results in an application context. A global temporary table shares its definition system-wide with all users with appropriate privileges, but the data content is always session-private. No physical storage is allocated unless data is inserted into the table. A global temporary table can be transaction specific (delete rows on commit) or session-specific (preserves rows on commit). Gathering statistics on transaction specific tables leads to the truncation of the table. In contrast, it is possible to gather statistics on a global temporary table (that persist rows) but in previous releases its wasn't always a good idea as all sessions using the GTT had to share a single set of statistics, so a lot of systems relied on dynamic statistics. However, in Oracle Database 12c, it is now possible to have a separate set of statistics for every session using a GTT. Statistics sharing on a GTT is controlled using a new DBMS_STATS preference GLOBAL_TEMP_TABLE_ STATS. By default, the preference is set to SESSION, meaning each session accessing the GTT will have its own set of statistics. The optimizer will try to use session statistics first but if session statistics do not exist, then optimizer will use shared statistics.

Intermediate Worktables: Intermediate work tables are typically seen as part of an ELT process or a complex transaction. These tables are written to only once, read once, and then truncated or deleted. In such cases the cost of gathering statistics outweighs the benefit, since the statistics will be used just once. Instead, dynamic sampling should be used in these cases. It is recommended you lock statistics on intermediate worktables that are persistent to prevent the automated statistics gathering task from attempting to gather statistics on them.

3.3. How to Gather Statistics

The preferred method for gathering statistics in Oracle is to use the automatic statistics gathering. If you already have a well-established, manual statistics gathering procedure then you might prefer to use that instead. Whatever method

you choose to use, start by considering whether the default global preferences meet your needs. In most cases they will, but if you want to change anything then you can do that with SET_GLOBAL_PREFS. Once you have done that, you can override global defaults where necessary using the DBMS_STATS "set preference" procedures. For example, use SET_TABLE_PREFS on tables that require incremental statistics or a specific set of histograms. In this way, you will have declared how statistics are to be gathered, and there will be no need to tailor parameters for individual "gather stats" operations. You will be free to use default parameters for gather table/schema/database stats and be confident that the statistics policy you have chosen will be followed. What's more, you will be able to switch freely between using auto and manual statistics gathering.

3.4. Auto Gather Stats

Oracle automatically collects statistics for all database objects, which are missing statistics or have stale statistics during a predefined maintenance window (10pm to 2am weekdays and 6am to 2am at the weekends). The maintenance window of the job that runs in via Enterprise Manager or using the DBMS_SCHEDULER and DBMS_AUTO_TASK _ADMIN packages can be altered based on the need. [1]

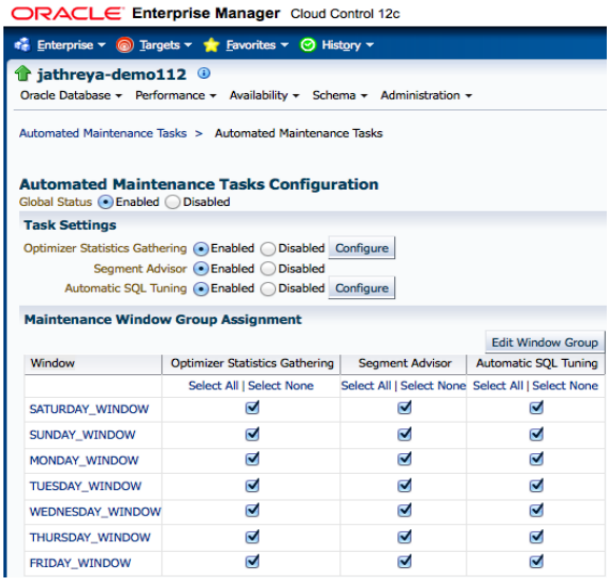

Figure 1. Automated maintenance tasks configuration

If a well-established manual statistics gathering procedure exists or if for some other reason you want to disable automatic statistics gathering you can disable the task altogether using below command:

```
begin
 dbms_auto_task_admin.disable(
 client_name=>'auto optimizer stats collection',
 operation=>null,
 window_name=>null);
end;
/
```

Figure 2. Disable auto stats gathering

3.5. Manual Gather Stats

If we intend to manually maintain optimizer statistics, when statistics should be gathered has to be determined based on staleness, as it is for the automatic job, or based on when new data is loaded in your environment. It is not recommended to continually re-gather statistics if the underlying data has not changed significantly as this will unnecessarily waste system resources. If data is only loaded into your environment during a pre-defined ETL job, then the statistics gathering operations can be scheduled as part of this process. Online statistics gathering and incremental statistics can be beneficial to include as part of statistics maintenance strategy.

3.6. Other Types of Gather Stats

Since the Cost Based Optimizer is now the only supported optimizer, all tables in the database need to have statistics, including all of the dictionary tables (tables owned by SYS, SYSTEM, etc., and residing in the system and SYSAUX tablespace) and the x$ tables used by the dynamic v$ performance views.

Dictionary Statistics: Statistics on the dictionary tables are maintained via the automated statistics gathering task run during the nightly maintenance window. It is highly recommended that you allow the automated statistics gather task to maintain dictionary statistics even if you choose to switch off the automatic statistics gathering job for your main application schema. You can do this by changing the value of AUTOSTATS_TARGET to ORACLE instead of AUTO using the procedure DBMS_STATS.SET_ GLOBAL_PREFS.

```
exec dbms_stats.set_global_prefs('autostats_target','oracle')
```

Fixed Object Statistics: Beginning with Oracle Database 12c, fixed object statistics are collected by the automated statistics gathering task if they have not been previously collected. Beyond that, the database does not gather fixed object statistics. Unlike other database tables, dynamic sampling is not automatically used for SQL statements involving X$ tables when optimizer statistics are missing so the optimizer uses predefined default values for the statistics if they are missing. These defaults may not be representative and could potentially lead to a suboptimal execution plan, which could cause severe performance problems in your system. It is for this reason that we strongly recommend you manually gather fixed objects statistics.

You can collect statistics on fixed objects using DBMS_ STATS.GATHER_FIXED_OBJECTS_STATS procedure. Because of the transient nature of the x$ tables it is import that you gather fixed object statistics when there is a representative workload on the system. This may not always be feasible on large systems due to additional resources needed to gather the statistics. If you can't do it during peak load, you should do it after the system has warmed up and the three key types of fixed object tables have been populated:

- Structural data - for example, views covering datafiles, controlfile contents, etc.
- Session based data - for example, v$session, v$access, etc.
- Workload data - for example, vsql, vsql_plan, etc.

It is recommended that you re-gather fixed object statistics if you do a major database or application upgrade, implement a new module, or make changes to the database configuration. For example, if you increase the SGA size then all of the x$ tables that contain information about the buffer cache and shared pool may change significantly, such as x$ tables used in v$buffer_pool or v$shared_pool_advice.

System Statistics: System statistics enable the optimizer to more accurately cost each operation in an execution plan by using information about the actual system hardware executing the statement, such as CPU speed and IO performance. System statistics are enabled by default, and are automatically initialized with default values; these values are representative for most systems.

4. Technical Design and Steps to Implement Gather Stats in ETL Informatica Workflows

4.1. Situation

Let's walk thru a situation of an application having approximately two hundred ETL Informatica workflows/jobs that load data from multiple sources like Hadoop, Teradata, Oracle, Mainframe files to target tables in oracle. Some jobs truncate and load data, while some appends data in target tables everyday based on different needs. Due to everyday data refresh for hundreds of tables using ETL Informatica jobs, the gradual exponential growth of data resulted in degradation of application performance. This also resulted in failure of application performance test and adversly impacted the deployments. With no preferred statistics collection in place, it became hard for technical team to find the root cause of application's poor performance. Business team often complained about the negative impact due to the poor performance. The team has to resolve this as soon as possible to avoid delaying any further deployments for application maintenance or enhancements.

4.2. Required Technical Skillsets

- Oracle Database
- SQL
- Stored Procedures
- Concepts and Usage of gather stats
- ETL Informatica PowerCenter

4.3. Implementation Tasks to Perform

The situation explained above can be handled effectively by performing below tasks:

1. Get an estimate of the loaded records count by analysing past three months of ETL Informatica workflows logs.
2. Scan ETL Informatica mappings to identify any poorly written queries and logic.
3. Scan the database scripts for tables, views, indexes, procedures etc., to identify any poorly written DDLs, DMLs.
4. Using available tools like Dynatrace, assess the data growth percentile in the database.
5. Work with DBAs to verify and identify alternate options for performance tuning.
6. Fine tune the queries wherever applicable and add indexing to improve the performance.
7. Create a stored procedure to execute gather_table_ stats with cascade option to automatically gather statistics for indexes along with the tables.
8. Call the procedure as a post SQL in each informatica workflow providing the table name as parameter.
9. Analyse and use the statistics gathered on a regular basis and implement the fixes for performance tuning.

4.4. Gather Stats Procedure

Create a stored procedure using the script similar shown in below screenshot. This is just an example procedure to collect the statistics on an oracle table, however we can collect statistics on individual items like index, column etc., based on the need.

```
--This procedure gathers statistics for the table supplied in the input parameter, it also gathers statistics for all the indexes related to the table.
--After gathering statistics, it also unlocks table statistics to avoid any locking of stats on a table.

CREATE OR REPLACE PROCEDURE SCHEMA_NAME.PROCEDURE_NAME
(
 IN_SCHEMA_NAME         VARCHAR2
,IN_TABLE_NAME          VARCHAR2
,IN_TABLE_PARTITION     VARCHAR2
) IS


BEGIN

DBMS_STATS.GATHER_TABLE_STATS
(
 OwnName                => IN_SCHEMA_NAME
,TabName                => IN_TABLE_NAME
,Estimate_Percent       => 10
,Method_Opt             => 'FOR ALL COLUMNS SIZE 1'
,Degree                 => 4
,Cascade                => TRUE
,No_Validate            => FALSE
);


DBMS_STATS.UNLOCK_TABLE_STATS
(
 OwnName                => IN_SCHEMA_NAME
,TabName                => IN_TABLE_NAME
);

END;
```

Figure 3. Gather Table Stats script as a stored procedure

4.5. ETL Informatica Parameter File

Most of the times ETL Informatica workflows uses a parameter file for dynamic lookup of workflow parameters like schema name, table name and partition key and in that case update that parameter file to include the schema, table and partition key values.

```
[Global]
$PARAM_SCHEMA=ABC
$PARAM_TABLE_NAME=SAMPLE_TABLE_1
$PARAM_TRUN_PART_KEY=P1
$PARAM_GATHER_STATS_SP=ABC.PROCEDURE_NAME
```

Figure 4. Schema, Table and Partition supplied dynamically from ETL Informatica Parameter File

4.6. Post SQL in ETL Informatica Workflow

Add below script as a post SQL in ETL Informatica workflow in order to execute the gather stats procedure.

```
CALL $PARAM_GATHER_STATS_SP('$PARAM_SCHEMA','$PARAM_TABLE_NAME','$PARAM_TRUNC_PART_KEY');
```

Figure 5. Post SQL in ETL Informatica Workflow

4.7. Results

4.7.1. Before Gathering Statistics

```
SELECT    OWNER
         ,TABLE_NAME
         ,LAST_ANALYZED
         ,STALE_STATS
         ,STATTYPE_LOCKED

FROM     DBA_TAB_STATISTICS

WHERE    OWNER = 'ABC'
;
```

Figure 6. SQL to fetch last analysed statistics

OWNER	TABLE_NAME	LAST_ANALYZED	STALE_STATS	STATTYPE_LOCKED
ABC	SAMPLE_TABLE_1	11/28/2023 12:39	NO	
ABC	SAMPLE_TABLE_2	11/28/2023 12:34	NO	
ABC	SAMPLE_TABLE_3	11/28/2023 12:35	NO	
ABC	SAMPLE_TABLE_4	11/28/2023 12:28	NO	
ABC	SAMPLE_TABLE_5	11/28/2023 12:29	NO	
ABC	SAMPLE_TABLE_6	11/28/2023 12:36	NO	
ABC	SAMPLE_TABLE_7	11/28/2023 12:44	NO	
ABC	SAMPLE_TABLE_8	11/28/2023 12:46	NO	
ABC	SAMPLE_TABLE_9	11/28/2023 12:51	NO	
ABC	SAMPLE_TABLE_10	11/28/2023 12:53	NO	

Figure 7. Sample results before gathering table statistics

4.7.2. After Gathering Statistics

OWNER	TABLE_NAME	LAST_ANALYZED	STALE_STATS	STATTYPE_LOCKED
ABC	SAMPLE_TABLE_1	11/29/2023 10:11	YES	
ABC	SAMPLE_TABLE_2	11/29/2023 10:12	YES	
ABC	SAMPLE_TABLE_3	11/29/2023 10:09	YES	
ABC	SAMPLE_TABLE_4	11/29/2023 10:14	YES	
ABC	SAMPLE_TABLE_5	11/29/2023 10:31	YES	
ABC	SAMPLE_TABLE_6	11/29/2023 10:22	YES	
ABC	SAMPLE_TABLE_7	11/29/2023 11:20	YES	
ABC	SAMPLE_TABLE_8	11/29/2023 10:44	YES	
ABC	SAMPLE_TABLE_9	11/29/2023 9:55	YES	
ABC	SAMPLE_TABLE_10	11/29/2023 9:42	YES	

Figure 8. Sample results after gathering table statistics

4.8. Common Pitfalls and Precautionary Measures

Below mentioned are some common possible pitfalls someone could face during and post implementation and these can be avoided with a little due diligence and proactive measures.

Syntax of gather stats procedure: The syntax issues in gather stats stored procedure due to human error could result in failure of its execution. These can easily be avoided by reviewing the coded procedure and by performing a test execution of it.

Syntax of post SQL in ETL informatica workflow: The syntax issues in post SQL of ETL Informatica workflow could result in failure of its execution. These can easily be avoided by reviewing it and by performing a test execution of it.

Duplicate stats gathering on same table: As an example, if same table stats are being collected from two or more ETL Informatica workflows, it would result gathering duplicate stats for the same table. This can be avoided by making sure the stats are collected for a table only once wherever it makes more sense.

Contention of table with concurrent execution: Hitting the same table to gather stats from more than one workflow could sometimes result in the table contention causing the deadlock situation. This can be avoided by making sure the job that hits the table first takes exclusive lock of the table.

Gathering stats during Oracle maintenance window: To avoid any issues or failures in gather stats execution, ensure not to schedule this during Oracle maintenance window.

5. How to Use Gathered Statistics

Good quality statistics are essential to be able to generate optimal SQL execution plans, but sometimes statistics can be of poor quality and this fact could remain unnoticed. For example, older "inherited" systems might use scripts that are no longer understood by the database administrators and, understandably, there is a reluctance to change them. However, because Oracle continuously enhances statistics gathering features it is possible that best practice recommendations will be neglected.

For these reasons, Oracle Database 12c Release 2 includes a new advisor called the Optimizer Statistics Advisor to help you to improve the quality of statistics in the database. This diagnostic software analyses information in the data dictionary, assesses the quality of statistics and discovers how statistics are being gathered. It will report on poor and missing statistics and generate recommendations to resolve these problems.

The principle behind its operation is to apply best-practice Rules to uncover potential problems. These problems are reported as a series of Findings, which in turn can lead to specific Recommendations. Recommendations can be implemented automatically using Actions (either immediately or via an auto-generated script to be executed by the database administrator). [1]

Figure 9. Optimizer Statistics Advisor

The advisor task runs automatically in the maintenance window, but it can also be run on demand. The HTML or text report generated by the advisor can be viewed at any time and the actions can be implemented at any time.

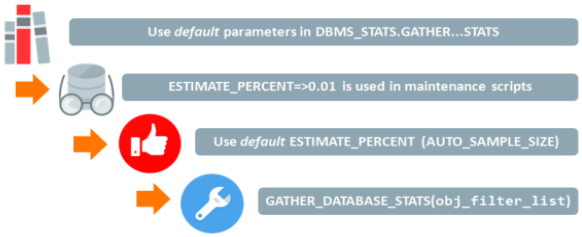

Figure 10. Rule, Finding, Recommendation and Action

The advisor task gathers and stores data in the data dictionary. It is a low performance overhead operation because it performs an analysis of optimizer statistics and statistics gathering information (that's already held in the data dictionary). It does not perform a secondary analysis of data stored in application schema objects.

Figure 11. Reading the data dictionary, executing the task via a filter and storing the results

Once the task is complete, the report can be generated in HTML or text format and an action (SQL) script can be created too.

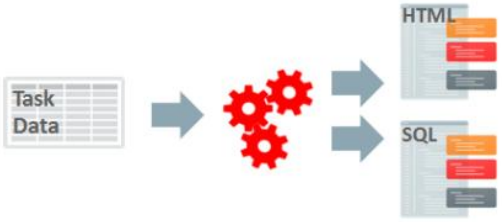

Figure 12. Reporting on the advisor task and generating the action SQL script

It is a simple matter to view the report generated by the automated task:

```
select dbms_stats.report_advisor_task('auto_stats_advisor_task') as report from dual;
```

Figure 13. Reporting on the advisor task and generating the action SQL script

Alternatively, users with the ADVISOR privilege can execute the task manually and report on the results using the following three-step process:

```
DECLARE
 tname VARCHAR2(32767) := 'sample'; -- task name
BEGIN
 tname := dbms_stats.create_advisor_task(tname);
END;
/
DECLARE
 tname VARCHAR2(32767) := 'sample'; -- task name
 ename VARCHAR2(32767) := NULL;     -- execute name
BEGIN
 ename := dbms_stats.execute_advisor_task(tname);
END;
/
SELECT dbms_stats.report_advisor_task('sample') AS report
FROM dual;
```

Figure 14. Reporting on the advisor task and generating the action SQL script

The actions generated by the advisor can be implemented immediately:

```
DECLARE
 tname VARCHAR2 (32767) := 'demo'; -- task name
 impl_result CLOB; -- report of
 implementation
BEGIN
 impl_result := dbms_stats.implement_advisor_task(tname);
END;
/
```

Figure 15. Reporting on the advisor task and generating the action SQL script

6. Conclusions

This paper concludes on a note that gathering statistics on a regular basis is very essential when there is a database with huge data is involved as it helps to assess and fine tune the performance of an application. The example provided the evidence of a successful implementation of gather stats in ETL Informatica jobs and the design provided would guide anyone to implement the gather stats to enhance the performance by optimizing database queries. Using statistics gathering task and the other techniques described in this paper a database administrator can maintain accurate set of statistics and ensure the optimizer always has the necessary information in order to select the most optimal plan. Once a statistics gathering strategy has been put in place, if needed it should be updated in a controlled manner by taking the advantage of key features such as pending statistics to ensure they do not have an adverse effect on application performance.

REFERENCES

[1] https://www.oracle.com/docs/tech/database/technical-brief-bp-for-stats-gather-19c.pdf

[2] https://docs.oracle.com/en/database/oracle/oracle-database/23/tgsql/gathering-optimizer-statistics.html

[3] Malik Syed, Mainframe Modernization Strategies, American Journal of Computer Architecture, Vol. 11 No. 1, 2024, pp. 1-9. doi: 10.5923/j.ajca.20241101.01.

[4] Pradeep Jain, Implementation of ACH Notification of Change, *American Journal of Computer Architecture*, Vol. 11 No. 4, 2024, pp. 48-52. doi: 10.5923/j.ajca.20241104.03.