Oracle Streams Performance Tuning Best Practices: Oracle Database 10*g* Release 10.2

An Oracle White Paper
November 2008

Maximum Availability Architecture

Oracle Best Practices for High Availability

ORACLE

Oracle Streams Performance Tuning Best Practices: Oracle Database 10*g* Release 2

Oracle Streams Performance Tuning Best Practices: Oracle Database 10*g* Release 2

INTRODUCTION

Performance tuning is an iterative process. Accurately diagnosing the performance problem and detecting the top bottlenecks is the first step.

The Maximum Availability Architecture (MAA) [1] is a best practices blueprint for achieving high availability and performance using Oracle technologies. This MAA white paper provides practical monitoring, performance, and tuning techniques for Oracle Streams configurations.

Once you have configured Oracle Streams, database changes are captured on the source database, and then propagated and applied on the target database. This paper describes best practices for improving Streams performance in a functional Stream environment.

By using these best practices and tuning and troubleshooting methodology, Streams throughput can increase significantly in some cases.

The paper contains the following sections:

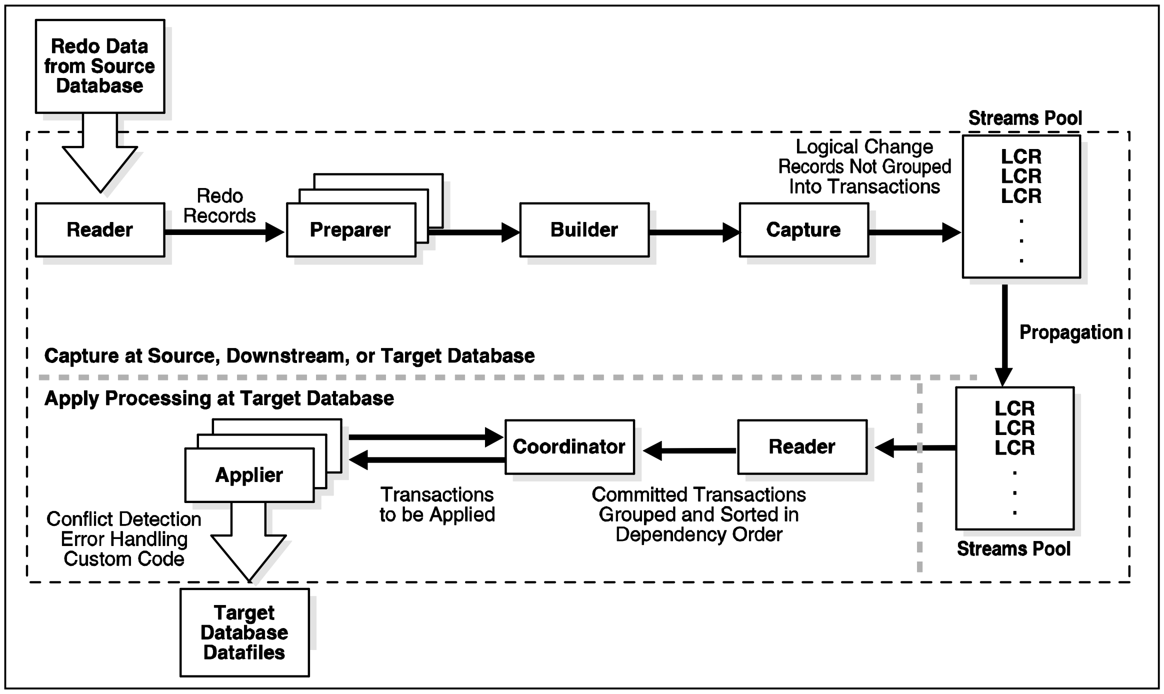
- Streams Process Flow
- Streams Configuration Prerequisites
- Gathering Performance Data
- Tuning Streams Performance
- Analyzing Streams Workflow Using the STRMMON Utility

The MAA testing described in this white paper used Oracle Streams with Oracle Database 10*g* Release 2 (10.2.0.4). This white paper is a companion to the "*Oracle Streams Configuration, Best Practices: Oracle Database 10g Release 2*" [2] MAA white paper that describes Streams Configuration best practices and troubleshooting recommendations.

STREAMS PROCESS FLOW

Oracle Streams enables the propagation and management of data, transactions, and events in a data stream either within a database, or from one database to another. Oracle Streams consists of three components: capture, propagation, and apply. Each of these components is made up of several Operating System processes, as shown in Figure 1

Figure 1: Oracle Streams Processing Architecture


Redo Data from Source Database
Streams Pool
Logical Change Records Not Grouped Into Transactions
LCR LCR LCR
Reader
Redo Records
Preparer
Builder
Capture
Propagation
Capture at Source, Downstream, or Target Database
Apply Processing at Target Database
LCR LCR LCR
Coordinator
Reader
Applier
Transactions to be Applied
Committed Transactions Grouped and Sorted in Dependency Order
Conflict Detection Error Handling Custom Code
Streams Pool
Target Database Datafiles

The capture process mines redo logs from a source database and captures changes. It consists of multiple operating-system processes:

- A **reader** process reads the redo log and divides the redo log into regions.
- The **preparer** processes scan the regions defined by the reader in parallel and pre-filter changes found in the redo log.
- A **builder** process merges redo records from the preparers and passes the merged redo records to the capture process.
- The **capture** process then formats each change into a logical change record (LCR) and enqueues to a single queue if it satisfies the defined capture rules.

You can configure one or more capture processes locally at a source database or remotely at a downstream database. A single capture process can send changes to multiple propagation and apply processes. You can also configure multiple capture processes each associated with their own separate queue.

The Oracle Streams propagation component sends LCRs to the destination database once the capture process has enqueued them into the local capture queue. Note that in downstream capture where the capture and apply process share the same queue, Data Guard redo transport mechanism is used to propagate the redo from source to target..

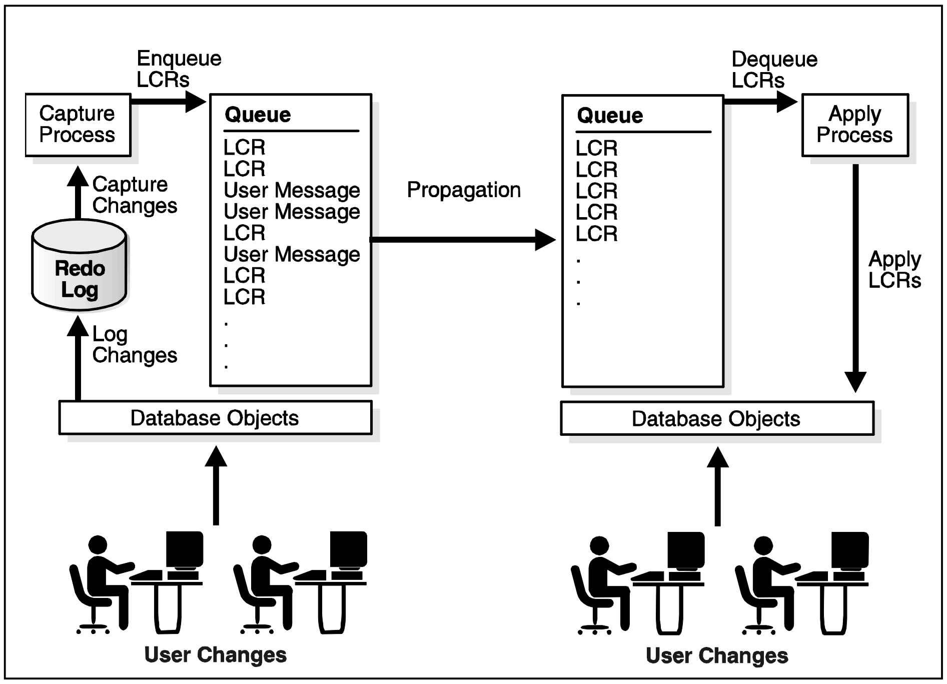
The apply component applies transactions on the target database. It consists of multiple operating system processes:

- A reader process (**apply reader**) browses LCRs in the apply queue, determines transaction dependencies, and schedules transactions to be applied.
- A coordinator process (**apply coordinator**) assigns transactions to specific apply server processes.
- One or more server processes (**apply servers**) that apply transactions assigned to them by the apply coordinator process.

Local Capture Processing

A local capture process runs at the source database and captures changes from the local source database redo log. Oracle Streams local capture is shown in Figure 2.

Figure 2: Oracle Streams Local Capture Processing



Downstream Capture Processing

You can configure a downstream capture process for real-time capture in which redo data from source database is transmitted to the downstream database and written to the *standby redo log files*, from which the capture process captures changes. Streams downstream capture is shown in Figure 3.

Figure 3: Oracle Streams Downstream Capture

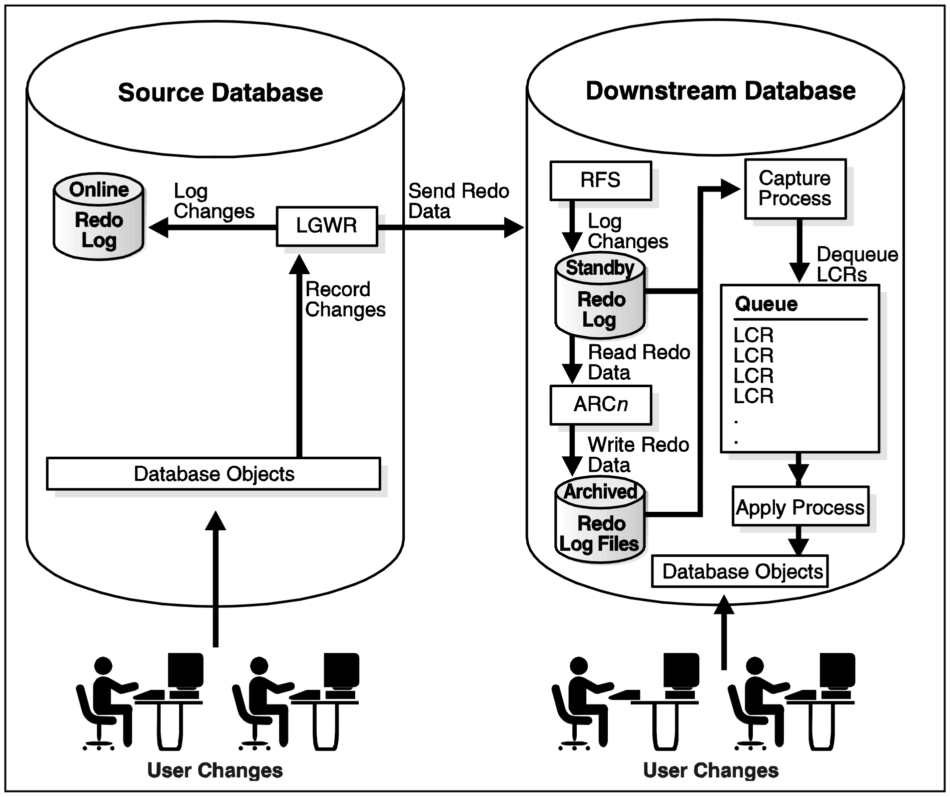
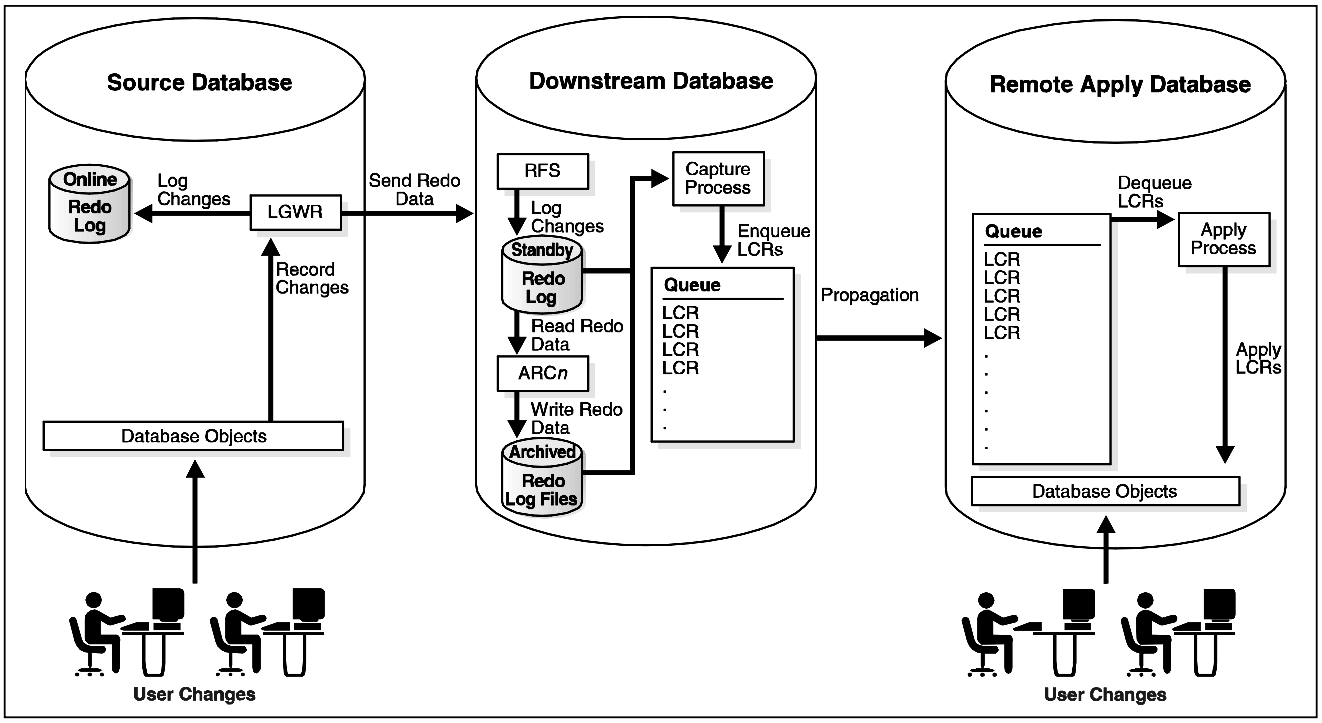


Figure 4 shows another form of downstream capture with a source database, a downstream capture database and a separate database where changes are applied.

Figure 4: Downstream Capture with a Remote Apply Database



STREAMS CONFIGURATION PREREQUISITES

This white paper assumes that you have already configured Oracle Streams and it is fully functional. To verify your Oracle Streams configuration, perform the following tasks:

- Configure Oracle Streams following the recommendations provided in the "*Oracle Streams Configuration Best Practices: Oracle Database 10g Release 2*" [2] white paper.
- Ensure the database parameters shown in Table 1 are set on each database instance running Oracle Streams:

Table 1: Recommended Initialization Parameter Settings for Oracle Streams

Database Parameter	Recommended Value(s)
`AQ_TM_PROCESSES`	Do not explicitly set this parameter to 0 or 10. Doing so could disable the queue monitoring processing and impact the streams pool memory utilization. • If this parameter has not been set, the parameter will be autotuned. • If this parameter has been explicitly set to 0 or 10, either in the initialization parameter file or by the `ALTER SYSTEM` statement, then reset the parameter to 1.
`JOB_QUEUE_PROCESSES`	>= 4
`_JOB_QUEUE_INTERVAL`	= 1
`STREAMS_POOL_SIZE`	>= 256 MB
`TIMED_STATISTICS`	`TRUE`
`STATISTICS_LEVEL`	`TYPICAL`

- Verify your Oracle Streams environment by performing the following steps:
  1. Run the Oracle Streams Health Check script on all databases that contain a capture or apply process. See Appendix D for more information.
  2. Review the Oracle Streams Health Check report for any errors in any of the Oracle Streams processes: capture, propagation and apply.
  3. Begin the Oracle Streams tuning and performance tasks described in this white paper only after you have implemented the recommended MAA best practices for configuring Oracle Streams.

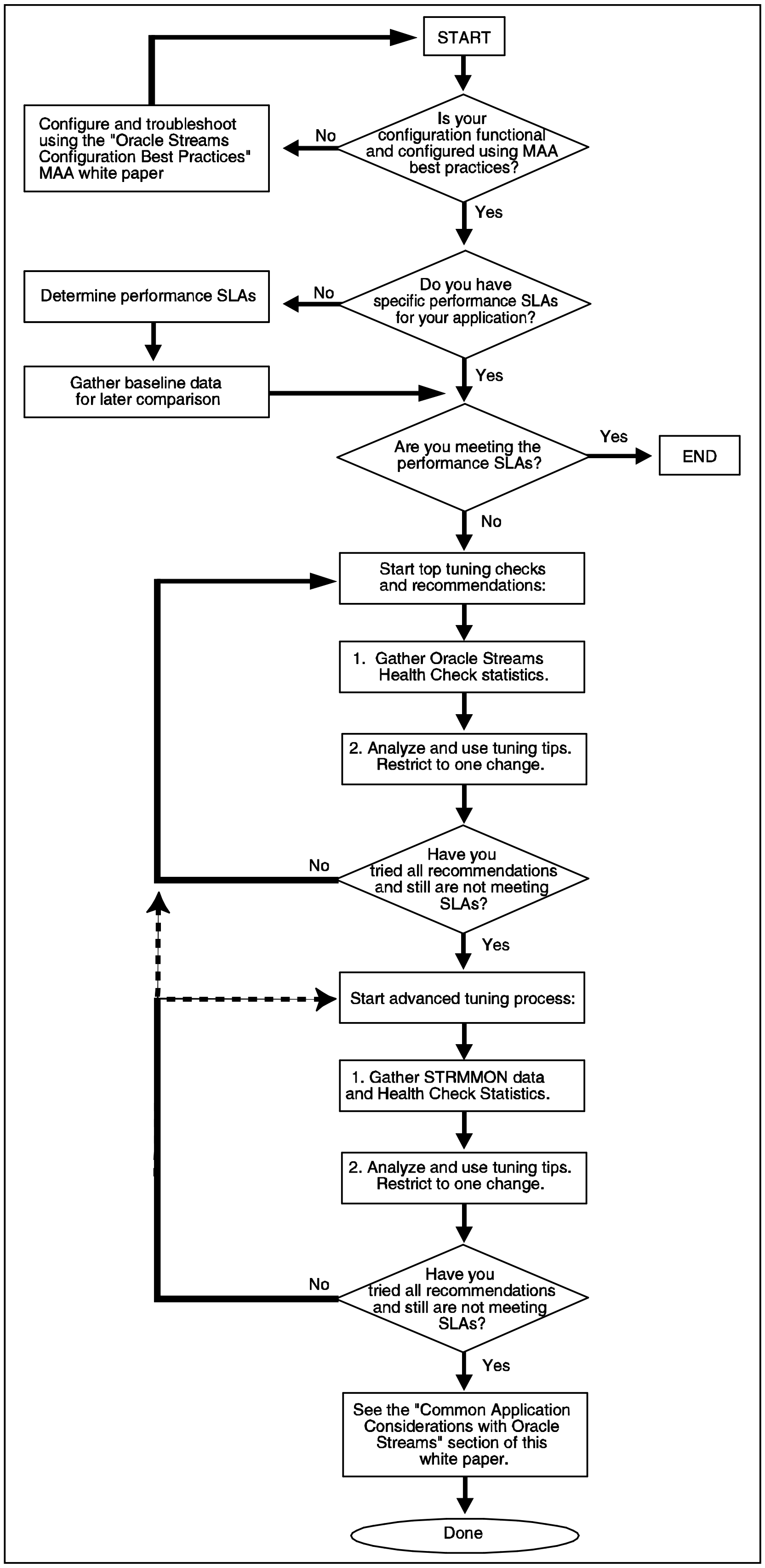
     Note: Repair errors, if any, using the guidelines in the "Troubleshooting Streams Configurations" section in the "*Oracle Streams Configuration Best Practices: Oracle Database 10g Release 2*" [2] white paper.

ROADMAP TO ORACLE STREAMS PERFORMANCE TUNING

This section describes the high-level flow recommended for tuning Streams.

1. Is your configuration functional and configured using the MAA best practices?
   a. YES: Continue with step 2.
   b. NO: Use the recommendations in the "Repairing Common Run-Time Errors" section of the "*Oracle Streams Configuration Best Practices: Oracle Database 10g Release 2*" [2] white paper.
2. Do you have specific performance service level agreements (SLAs) for your Oracle Streams environment?
   a. YES: Continue with question 3.
   b. NO: Define specific performance SLAs. See the "Establish Target and Performance Goals" section.
3. Are you meeting your performance SLAs?
   a. YES: STOP—No further performance tuning is necessary.
   b. NO: Continue with question 4.
4. Perform the Oracle Streams top tuning checks and recommendations:
   a. Run Oracle Streams Health Check script on all Streams databases, and evaluate the following guidelines in the "Initial Checks and Tuning Recommendations" section.
   b. Make only one change at a time in step a, then rerun Oracle Streams Health Check scripts, and reevaluate the effect of each change.
   c. If SLAs are met, STOP. Else, repeat steps a and b in step 4 and reevaluate until all checks and tuning recommendation have been exhausted.
5. Perform Advanced Oracle Streams Performance Tuning:
   a. Ensure that AWR is configured and taking snapshots once every hour, and configure and run the STRMMON utility using the analysis and recommendations described in the "Analyzing Oracle Streams Workflow Using STRMMON Data" section.
   b. Make only one change at a time and reevaluate the effect on the Oracle Streams workflow performance.
   c. If SLAs are being met, STOP. Else, repeat until SLAs are met or until all recommendations have been exhausted.
6. Perform application tuning, as necessary, to clear remaining performance issues. See "Common Application Considerations with Oracle Streams."

Figure 5: Flow Chart for Oracle Streams Performance and Tuning


START
Is your configuration functional and configured using MAA best practices?
No
Configure and troubleshoot using the "Oracle Streams Configuration Best Practices" MAA white paper
Yes
Do you have specific performance SLAs for your application?
No
Determine performance SLAs
Gather baseline data for later comparison
Yes
Are you meeting the performance SLAs?
Yes
END
No
Start top tuning checks and recommendations:
1. Gather Oracle Streams Health Check statistics.
2. Analyze and use tuning tips. Restrict to one change.
Have you tried all recommendations and still are not meeting SLAs?
No
Yes
Start advanced tuning process:
1. Gather STRMMON data and Health Check Statistics.
2. Analyze and use tuning tips. Restrict to one change.
Have you tried all recommendations and still are not meeting SLAs?
No
Yes
See the "Common Application Considerations with Oracle Streams" section of this white paper.
Done

GATHERING PERFORMANCE DATA

Use the following checklist to collect the required performance data:

- ❑ **Run the Oracle Streams Health Check script once a day on each Oracle Streams database.**

  Run the Oracle Streams Health Check script everyday, simultaneously on each database, and collect the output. You will use the resulting Oracle Streams Health Check reports to perform the tasks described in the "Initial Tuning Checks and Recommendations" section of this white paper. Download the Streams Health Check script from Oracle*MetaLink* Note 273674.1.

- ❑ **Install Automatic Workload Repository (AWR) and collect snapshots once every hour.**

  Install AWR on all Oracle Streams databases and collect performance data once every hour on all instances. If you see erratic performance, take frequent manual snapshots on all Oracle Streams database instances. For example, the following PL/SQL procedure creates a manual snapshot:

```
BEGIN
  DBMS_WORKLOAD_REPOSITORY.CREATE_SNAPSHOT(FLUSH_LEVEL => 'ALL');
END;
```

  Note: Automatic Database Diagnostic Monitor (ADDM) analyzes the data contained in the AWR snapshots. See Chapters 5 and 6 in the *Oracle Performance and Tuning Guide 10g Release 2 (10.2)* [6] documentation for details about AWR.

- ❑ **Install the STRMMON Utility and collect snapshots every hour.**

  You can use the STRMMON Utility to monitor the performance of an Oracle Streams environment and obtain a quick overview of the Oracle Streams activity in a database.

  Perform the following steps to install and use the Oracle Streams STRMMON utility (version 2.6 or higher) to assess the overall Oracle Streams performance:

  1. Download the latest version of the STRMMON utility from Oracle*MetaLink* Note: 296065.1.
  2. Unzip the `strmmon.zip` file and extract the `strmmon.c` C program file and the `strmmon.html` file into the `rdbms/demo` directory in your Oracle home.
  3. Follow the installation instructions to install and configure the STRMMON utility. Include the `–sysdba` parameter to obtain the streams pool memory usage.

4. Take STRMMON 30-second interval segments for ten minutes every hour, aligning the time to coincide with the AWR snapshot intervals. In the example command syntax in step 5 below, set the `–count` parameter to 20 instead of setting it to 60. After pinpointing a specific time window that requires tuning, collect samples as frequently as every 30 seconds over a 30-minute period. See Appendix A for a complete example of the output returned from the STRMMON utility.

5. To monitor more than one database, issue a STRMMON command similar to the following example. Include the `SYSDBA` parameter for each database to collect the streams pool utilization.

```
strmmon -interval 30 -count 60 -user sys -passw <sys_password>  \
  -dbname <TNS alias to source database> -sysdba -user sys \
  -passw <sys_password>  -dbname  <TNS alias to target database>\
  -sysdba >  strmmon.log
```

   This STRMMON command example specifies a collection interval of every 30 seconds (`-interval 30`), with a total iteration count of 60 (`-count 60`), which is equivalent to a duration of 30 minutes of output. (That is, 30 seconds multiplied by 60 iterations is equal to 1800 seconds, or 30 minutes.)

❑ **Gather additional performance statistics using Oracle Streams performance views.**

❑ **To gather a quick status or a more in-depth analysis, query the Oracle Streams dynamic performance views to determine the current state and processing flow of Oracle Streams:**

- `GV$STREAMS_APPLY_COORDINATOR`
- `GV$STREAMS_APPLY_READER`
- `GV$STREAMS_APPLY_SERVER`
- `GV$STREAMS_CAPTURE`
- `GV$STREAMS_POOL_ADVICE`
- `GV$STREAMS_TRANSACTION`

See Appendix C for information about downloading documents that contain sample SQL statements of queries that you can cut and paste into scripts for monitoring various Oracle Streams dynamic performance views. See the *Oracle Database Reference 10g Release 2 (10.2)* [7] for information about these views.

❑ **Monitor the `ALERT.LOG` file for critical Oracle Streams messages.**

Examine the Alert log for early detection of Oracle Streams performance issues. See Appendix B " Oracle Streams ALERT.LOG Messages for a list of critical messages that you should monitor in the Alert log.

- ❑ **Data Collection Retention**

  Purge performance data that is no longer needed to help reduce the amount of space and data set size. It is recommended that you label (timestamp) the output from the Oracle Streams Health Check report, from the STRMMON utility, and from the queries for the various `GV$STREAMS` views for organizational purposes and to help you find the associated performance data for specific time windows.

TUNING ORACLE STREAMS PERFORMANCE

This section provides the top checks and recommendations for Oracle Streams. See the "Roadmap to Streams Performance Tuning" section and follow the methodology provided there to guide you through the process.

The following sections provide a two-step tuning methodology for examining the various bottlenecks in the Oracle Streams process flow

1. The "Establish Target and Performance Goals" section defines goals for the Oracle Streams replication environment.
2. The "Initial Tuning Checks and Recommendations" section describes how to detect common problem areas using the Oracle Streams Health Check script.
3. The "Analyze the Oracle Streams Workflow Using STRMMON Data" section analyzes the Oracle Streams workflow using output from the STRMMON utility.

Establish Target and Performance Goals

Define goals for the Oracle Streams replication environment based on the application workload profile and lag time requirements on the target database. The following targets should be established:

- Throughput rate
- Data lag time (latency)

If these targets are being met by Oracle Streams, no further tuning is necessary.

The easiest way to determine the current throughput and lag time (latency) for Oracle Streams is to run the STRMMON utility as described in the previous checklist item that described how to "Install the STRMMON utility and collect data every hour, including:

- Capture LCRs captured/sec, LCRs enqueued/sec, and capture latency
- Apply LCRs applied/sec, transactions applied/sec and apply latency

This will provide rates and latency for both capture and apply.

It is strongly recommended that you complete the initial tuning check tasks in the "Initial Tuning Checks and Recommendations" section before going on to the "Analyze the Oracle Streams Workflow Using STRMMON Data" section.

Initial Tuning Checks and Recommendations

The initial tuning checks are designed to tackle a large percentage of performance issues for Oracle Streams. This section describes the following database and application tuning tasks:

Databases Checks

1. Ensure there are adequate system resources
2. Ensure there is adequate network bandwidth
3. Set the `COMMIT_SERIALIZATION` parameter for the apply process
4. Adjust the `STREAMS_POOL_SIZE` parameter
5. Adjust the degree of apply parallelism
6. Detect and reduce LogMiner spills

Application Checks

7. Common Application Considerations with Oracle Streams

Tuning is an iterative process. When implementing any of the following recommendations, run the Oracle Streams Health Check script again, and monitor the effect of the change. Revisit and evaluate each of the following checks to ensure other components are not negatively impacted.

Ensure There Are Adequate System Resources

Perform the following steps to ensure that each server that participates in a Oracle Streams replication topology is not limited by system resources (CPU, memory, I/O):

1. Check the CPU utilization and run queue size of each server to determine if there is sufficient headroom throughout a production day. For Unix/Linux platforms, use the SAR utility.
2. Check the I/O subsystem to ensure sufficient I/O throughput throughout a production day. On Unix/Linux systems, the `IOSTAT` utility can be used.
3. If the servers on which Oracle Streams runs are performing close to full capacity(CPU, I/O, memory), add additional headroom.

Ensure There Is Adequate Network Bandwidth

For an Oracle Streams configuration running over a Wide Area Network (WAN), you must:

1. Ensure there is sufficient network bandwidth to support maximum propagation rates. Consider reviewing network statistics (such as `NETSTAT`) with network administrators to understand the current network bandwidth utilization and the expected increase in usage.
2. Optimize network transfer by following the MAA best practices guidelines in Appendix A of the *Oracle Streams Configuration Best Practices: Oracle Database 10g Release 2* [2] white paper.

Set the `COMMIT_SERIALIZATION` Apply Parameter

In some cases, you can reduce the wait time and increase the throughput of the apply processes by allowing the apply servers to commit transactions independently of the order in which the transactions were committed on the source database[1]. When this parameter is set to `NONE`, this optimization can increase the apply rates by more than 50% for OLTP applications.

To determine if you should set the `COMMIT_SERIALIZATION` parameter to `NONE`, perform the following steps on the database where the apply process is running:

1. Determine the current setting of the `COMMIT_SERIALIZATION` parameter.

   At the top of the Oracle Streams Health Check report, click **Apply** on the "`Configuration`" line. Then, find the `COMMIT_SERIALIZATION` parameter in the `++ APPLY PROCESS PARAMETERS ++` table.
2. Determine the percentage of all transactions for which the apply process has waited to commit.

   On the Statistics line at the top of the Oracle Streams Health Check report, click **Apply** and look for `++ APPLY Coordinator Statistics ++` table.
3. Compute the overall percentage that the apply process has waited:

   % Wait Commit = (Total Txns Wait Commit / Total Txns Assigned) * 100
4. Set the `COMMIT_SERIALIZATION` parameter to `NONE` if either of the following conditions are true:
   - If the computation in Step 3 shows a 15% to 20% (or higher) wait time.
   - If the application running on the target database will not be adversely impacted when transactions are applied out of order compared to that of the source database.

[1] By default, the `COMMIT_SERIALIZATION` parameter is set to `FULL` so apply servers wait to commit a transaction until all other transactions with a lower commit SCN are applied.

Use the `DBMS_APPLY_ADM.SET_PARAMETER` PL/SQL procedure. For example:

```
BEGIN
        DBMS_APPLY_ADM.SET_PARAMETER('APPLY$_STREAMSS_36,
        'COMMIT_SERIALIZATION','NONE');
END;
/
```

Note: It is possible to see a high percentage of wait events for `COMMIT` transactions even when the `COMMIT_SERIALIZATION` apply parameter is set to `NONE`. See the "Common Application Considerations with Oracle Streams" section later in this white paper.

Adjust the STREAMS_POOL_SIZE Parameter

Oracle Streams uses the streams pool[2] to stage LCRs on both the source and target databases. Performance problems may result if the streams pool memory is insufficiently sized, buffered queue spills can occur, significantly affecting the overall Oracle Streams throughput.

The minimum recommended size for the `STREAMS_POOL_SIZE` parameter is 256 MB (see Table 1). However, depending on the application workload you might need to adjust the size of the `STREAMS_POOL_SIZE` parameter.

To reduce queue spills in the streams pool, use the Oracle Streams Health Check report to:

1. Review the Streams Pool Advice in Table 2 for the estimated spill count over a 24-hour time period.

   At the top of the Oracle Streams Health Check report, click **`History`** and scroll to the `++ Streams Pool Advice History for last day ++` table.

   The Streams Pool Advice table shows the streams pool history for the previous 24 hours, beginning at the time you ran the Oracle Streams Health Check report.

2. Examine the columns described in Table 2:

[2] The **Streams pool** is a portion of memory in the System Global Area (SGA) that is used by Oracle Streams to store buffered queue messages in memory and to provide memory for capture and apply processes. The streams pool always stores LCRs captured by a capture process, and it stores LCRs and messages that are enqueued into a buffered queue by applications or users.

Table 2: Columns in the Streams Pool Advice Table

Columns in the Streams Pool Advice Tables	Description
`BEGIN_TIME`, `END_TIME`, and `SNAP_ID`	Use these columns to relate the time interval and snapshot ID to the AWR for each sample and size estimate statistic
`SIZE_FOR_ESTIMATE`	Use this column to determine the size of the streams pool in MB.
`SIZE_FACTOR`	Use the size factor to determine your current streams pool size. The `SIZE_FACTOR` that is equal to 1.0 is your current streams pool size.)
`ESTD_SPILL_COUNT`	Use this column to determine the quantity of spills and the time of spills for the estimated spills that might occur at the estimated size. (`SIZE_FOR_ESTIMATE`)

3. Look for trends in the spill count.

   If you observe a trend where the estimated spill count is nonzero throughout most of the 24-hour period, set the `STREAMS_POOL_SIZE` parameter to a size where the estimated spill count is zero or close to zero.

   In the following example, the estimated size of the streams pool where estimated spill count is zero is 352 MB:

```
Size for    Size      Est Spill

 Est (MB)   Factor      Count

 ----------  ---------  -----------
    256      1.0      140,062  Current size of Streams pool and estimated spill count
    288      1.1       32,603
    320      1.3       11,955
    352      1.4            0   Estimated size of Streams Pool to reduce queue spills
```

   Thus, you should increase the `STREAMS_POOL_SIZE` from 256 MB to 352 MB (or higher). Increasing it to a value of 512 MB would provide additional memory space.

4. If you have increased the streams pool size several times without improvement, see the "Handling Large Transactions" section to determine if large transactions or other situations may be causing spills.

Adjust the Degree of Apply Parallelsim

Apply parallelism specifies the number of transactions that can be applied by Oracle Streams simultaneously. MAA testing recommends that you initially set the `PARALLELISM` parameter to 4. However, you may need to adjust the degree of parallelism to improve the throughput of the apply process.

To determine the appropriate value to specify for apply parallelism, follow these steps on the database where the apply process is running.

1. Monitor the apply servers over a 24-hour production day using the `APPLY_PCT_IDLE` query found in the `wp_apply.txt` file (see Appendix C).
2. Identify a trend in the percentage of time apply servers are idle.

   Each row that is returned shows information about each apply server process including `PERCENT_IDLE`. While the percent idle of each apply server may vary over a 24-hour time window, you should identify a trend across all apply server processes as to how busy (low percent idle) by averaging across all apply servers.
3. Increase the `PARALLELISM` parameter, if necessary.

   If the trend shows that on average, the apply servers are less than 10% idle during peak periods, then increase the `PARALLELISM` parameter by 2 from its current setting.

   For example, if the current value of `PARALLELISM` is 4, then increase it to 6 using the `DBMS_APPLY_ADM.SET_PARAMETER` PL/SQL procedure:

   ```
   BEGIN
       DBMS_APPLY_ADM.SET_PARAMETER('APPLY$_STREAMSS_36,
       'PARALLELISM','6');
   END;
   /
   ```

Detect and Reduce LogMiner Spills

To detect and reduce LogMiner spills, follow these steps on the database where the Capture process is running.

1. Look at the LogMiner statistics in the Oracle Streams Health Check report on the database where the capture process is running:

   At the top of the Oracle Streams Health Check report, click **Capture** on the Statistics line and scroll down to the `++ LOGMINER STATISTICS ++` line.

   If the statistics "`Bytes Paged Out`" is nonzero, then the LogMiner process has undergone a LogMiner spill.
2. If necessary, reduce LogMiner spills by performing the following tasks:

a. Increase the `STREAMS_POOL_SIZE` by 100 MB from its current setting.

b. Increase the `_SGA_SIZE` capture parameter to 100 MB.

For example:

```
BEGIN
  DBMS_CAPTURE_ADM.SET_PARAMETER(
    CAPTURE_NAME => 'STREAMSE$CAP',
    PARAMETER    => '_SGA_SIZE',
    VALUE        => '100');  -- Default is 10 MB
END;
/
```

Setting the`_SGA_SIZE` capture parameter may not eliminate LogMiner spills entirely, but by reducing the number of the spills it can reduce the overall time that LogMiner takes to mine and process transactions that may cause LogMiner spills.

Note: Check the databas e parameters `SGA_TARGET` and `SGA_MAX_SIZE` to ensure there is adequate room to increase the streams pool size.

Common Application Considerations with Oracle Streams

Several application scenarios can affect Oracle Streams performance. This section provides recommendations for reducing the performance effect of the following application profiles:

- Data definition language (DDL) statements
- Long-running transactions
- Large Transactions
- LOB data types

Handling Data Definition Language (DDL) Statements

When the Oracle Streams apply process encounters a DDL statement, any transaction after that DDL transaction waits until the DDL is applied. The STRMMON output may show the "`Streams: apply reader waiting for DDL to apply`" wait event of the apply reader process.

One of the following recommendations may address this situation:

- Do not replicate DDL or defer DDL statements on the source database until noncritical time periods if possible.
- Exclude tables that do not need to be replicated which have DDL operations. Use the `DBMS_STREAMS_ADM.ADD_TABLE_RULES()` procedure and set the parameter `INCLUSION_RULE=FALSE` for

these tables. This places the specific table into the *negative* rule set so that it will not be replicated.

Handling Long-Running Transactions

A message is written to the `ALERT.LOG` file when capture detects a long-running transaction that has been running for a minimum of 20 minutes. The apply process performs an apply spill of LCRs for a transaction after 15 minutes. Apply spilled transactions are read from disk rather than memory. Transactions involving aggregations rollups, report generators such as financial statements, audit reporting, etc, very often have very few changes but are long-running.

Detecting long-running transactions

1. In the Oracle Streams Health Check report for both source and target database, click **Capture** or **Apply** on the Statistics line. Scroll down until you see the "`OPEN TRANSACTIONS`" table for either Capture or Apply. The table shows all open transactions for either the Capture or Apply processes.

   Transactions remain open on the source database until the capture process processes a `COMMIT` statement. Transactions remain opened on the target database until the capture process applies and commits transaction.

2. If a transaction remains opened for some period of time, look for large cumulative message counts, large time differences between the first and last message times, or both.

Recommendation for long-running transactions—With the above understanding, consider modifying applications to reduce long-running transactions.

Note: See Appendix A for a complete example of the output returned from the STRMMON utility.

Handling Large Transactions

On the capture database, a message is written to the `ALERT.LOG` file after 10,000 row changes for a given transaction. On the apply database, if the apply parameter `TXN_LCR_SPILL_THRESHOLD` is exceeded (the default is 10,000 changes), apply spills occur.

Detecting large transactions—To detect a large transaction:, use the Oracle Streams Health Check script, as described in the "Long-Running Transactions" section.

Recommendation for Large Transactions

1. If possible, modify the application to include more `COMMIT` points as a way to reduce the transaction size. Note that this recommendation may not be appropriate for all applications.

2. If possible, determine the large transaction size and then assess if you can increase streams pool size to accommodate the entire transaction. If so, then increase the apply parameters TXN_LCR_SPILL_THRESHOLD and STREAMS_POOL_SIZE.

   If large transactions represent a higher percentage of your overall production workload profile and it is not feasible to adjust the STREAMS_POOL_SIZE or TXN_LCR_SPILL_THRESHOLD parameters, then see Oracle *MetaLink* Note 315666.1. Otherwise, leave the TXN_LCR_SPILL_THRESHOLD parameter at its default value because only a small percentage of the transaction profile is affected and you should expect some amount of spilling.

   The following example shows how to set the TXN_LCR_SPILL_THRESHOLD parameter:

```
BEGIN
        DBMS_APPLY_ADM.SET_PARAMETER('APPLY$_STREAMSS_36,
        'TXN_LCR_SPILL_THRESHOLD','15000');
END;
/
```

   Note: Also, see Appendix A for a complete example of the output returned from the STRMMON utility.

Handling Applications That Use LOB Data Types

By default, when Oracle Streams replicates tables that have LOB column data types, the LOBs are handled as multiple LCRs containing chunks of the LOB data. A LOB chunk is a piece of the LOB data contained in a data block. The apply process applies each LOB LCR of a row individually compared to assembling and applying an entire row.

To improve the performance for applying LOBs, you should register a DML error handler for the Oracle Streams apply process by setting the ASSEMBLE_LOBS parameter to TRUE[3] on the DBMS_APPLY_ADM.SET_DML_HANDLER PL/SQL procedure. The DML error handler enables the Oracle Streams apply process to assemble all of the LOB chunks in the least number of LCRs necessary when it is applying LOBs on the target objects.

Note: See Appendix C for information about downloading an example of a DML error handler for the apply process. See the *Oracle Database PL/SQL Packages and Types Reference 10g Release 2 (10.2)* [5] for information about the DBMS_APPLY_ADM.SET_DML_HANDLER PL/SQL procedure

[3] If the ASSEMBLE_LOBS parameter set to TRUE, then LOB assembly is used for LOB columns in LCRs processed by the DML handler. LOB assembly combines multiple LCRs for a LOB column resulting from a single row change into one row LCR before passing the LCR to the DML handler. Oracle Database compatibility must be set to 10.2.0 or higher to use LOB assembly.

Table 3 shows the capture and apply rates for LOBs before and after registering the DML error handler.

Table 3: Capture and Apply Rates for LOBs for a Registered Versus Non-Registered DML Error Handler

	LOB Handling When `ASSEMBLE_LOBS=FALSE` (or if the parameter is not set)	**LOB Handling When `ASSEMBLE_LOBS=TRUE`**	**Percent Change**
Capture Rate (LCRs/Sec)	1,263	1,750	28%
Capture Enqueue Rate (LCRs/Sec)	864	1196	28%
Apply Rate (LCRs/Sec)	864	1196	28%
Apply Rate (Transactions/Sec)	2.96	3.96	25%

The capture and apply rates for LOBs increased over 25% with the parameter setting: `ASSEMBLE_LOBS=TRUE`. See Appendix C for information about downloading a working example of an Oracle Streams DML error handler that improves LOB Performance.

ANALYZING THE ORACLE STREAMS WORKFLOW USING STRMMON DATA

By following the recommendations in the "Roadmap to Streams Performance Tuning" and the "Initial Tuning Check and Recommendations" section, most of the performance issues that otherwise might be encountered should be addressed. Follow these sections first before proceeding onto this section.

There are situations where further performance tuning may be necessary. Using the STRMMON utility will aid in identifying and addressing some of these performance issues.

The STRMMON utility is used to monitor Oracle Streams replication flow from the capture to the apply processes. See the "Gathering Performance Data" section above for installing and configuring STRMMON.

Based on your observations of STRMMON output, you should use one of the following cases as a guideline for your tuning methodology:

- Case 1: Apply latency is increasing
- Case 2: Capture rates are near zero and latency is increasing

Case 1: Apply Latency Is Increasing

In this scenario, the apply process latency shows an increasing time lag. Changes from the source database are propagated and applied, but Oracle Streams is not keeping up with the workload on the source database. You may also observe that no transactions are being applied for periods of time.

The tuning methodology provided for this case examines all components in the streams pipeline. The methodology starts by looking at the apply process and works backward to determine if any of the following indicators are present in the STRMMON output:

- Flow Control <F>
- Bottleneck <B>
- Wait Events

Flow Control <F>

Oracle Streams uses flow control to reduce or prevent buffered queue spills when some part of the streams process flow cannot keep up. Flow control causes all components *upstream* (shown to the left of the last component showing the <F> indicator for flow control or denoted by ***nn*%F**) to accommodate the slow Oracle Streams component. The <F> indicator in the STRMMON output may be displayed in front of the Capture or Apply processes, indicating that either process is operating under flow control. The <F> indicator is not always present but components may still be operating under flow control, as indicated by the *nn*%F indicator (for example, "78%F") in the STRMMON output. Flow control <F> and *nn*%F are only relevant for Capture (C00n), Propagation Sender (PS), Propagation Receiver (PR) , Apply (A00*n*), and Apply Reader (AR).

Example 1 provides a snippet of STRMMON output that shows the flow control indicators (highlighted in yellow) that are in effect.

Example 1: STRMMON Output Showing Flow Control Indicators

```
2008-03-17 11:39:41 || STRM10g5-> | LOG 571 | NET 252 1M | <F> C001 582 399
24min <1%I 78%F -> | Q53088 399 0 | PS01 370 0 0 <0%I 97%F -> | MEM 66 %
256M || STRM10g6-> | LOG 2M | NET 1M 50 | PR01 368 <3%I 0%F 69%"events in
waitclass Other"> | Q53478 368 0 | - A001 369 1 24min AR: <87%I 0%F -> AS(8)
<710%I 0%F -> | MEM 67 % 256M
2008-03-17 11:40:41 || STRM10g5-> | LOG 324 | NET 252 1M | <F> C001 562 386
25min <4%I 74%F -> | Q53088 386 0 | PS01 391 0 0 <0%I 97%F -> | MEM 66 %
256M || STRM10g6-> | LOG 2M | NET 1M 50 | PR01 398 <6%I 0%F 66%"events in
waitclass Other"> | Q53478 398 0 | - A001 409 1 25min AR: <83%I 0%F -> AS(8)
<700%I 0%F -> | MEM 67 % 256M
```

Use the information in Table 4 to help you understand the STRMMON utility output in Example 1:

Table 4: Description of the STRMMON Output Showing Flow Control Indicators

STRMMON Component	Description
2008-03-17 11:39:41 | | STRM10g5	Date/Time stamp and instance SID of the source database
LOG 571 | NET 252 1M	Redo log and network/DB Link sent/received statistics of the source instance
<F> C001 582 399 24min <1%I **78%F**	Capture process (C001) statistics: capture rate, enqueue rate, latency (24min), <F> indicates capture is under flow control, percent idle (1%I) and percent flow control (78%F).
Q53088 386 0	Source queue name, enqueue rate and buffered queue spill rate
PS01 370 0 0 <0%I **97%F**	Propagation Sender statistics: LCRs/second propagation rate, percent idle(0%I), percent flow control (97%F) – ignore the trailing two 0s.
MEM 66 % 256M	Streams pool memory utilization of the source database instance
STRM10g6-> | LOG 2M | NET 1M 50	Instance SID, Redo log and network/DB Link sent/received statistics of the target instance
PR01 368 <3%I **0%F** 69%"events in waitclass Other">	Propagation Receiver statistics: LCR/second received rate, percent idle (3%I), percent flow control (0%F), percent time in wait event (69% in event waitclass Other)
Q53478 368 0	Target queue name, enqueue rate and buffered queue spill rate
A001 369 1 24min AR: <87%I 0%F	Apply process (A001) statistics: LCR 369)and transaction (1) apply rates, latency (24min)
AR: <87%I 0%F	Apply Reader percent idle (87%I), percent flow control (0%F)
AS(8) <710%I 0%F	Apply Servers (8 servers processes) statistics: percent idle (710%I), percent flow control (0%F)
MEM 67 % 256M	Percent for streams pool memory percent utilization of the target database instance.

Previously, Example 1 showed two lines of output from STRMMON. Each line shows the `<F>` indicator before the Capture (C001) process, and both Capture and Propagation Sender (PS001) show a high percentage of time in flow control (as denoted by the *`nn`*`%F` indicator). The next component to the *right* is the Propagation Receiver (PR01), for which the example shows 0% in flow control but 69% (or 66%) is being spent on the wait event "`events in waitclass Others.`"

The first *downstream* component *that is not* showing flow control is typically the first place to start looking to identify the source of the bottleneck.

Table 5, which is shown after the following list, provides guidelines to help you determine potential bottlenecks that may be causing flow control. Use the following steps:

1. In the STRMMON output, locate the last component showing percentage of time in flow control, as denoted by *`nn`*`%F`, where *`nn`* is > 20. Then, in Table 5, find this component in the *Last Component In Flow Control* column.

2. In the STRMMON output shown in Example 1, identify the first component to the right, which should show no percentage of time in flow control.

3. Find the component described in Step 2 in the *Symptoms* column of Table 5 that best matches your current situation.

4. Evaluate the possible causes and recommendations Table 5. The recommendations are listed in priority order. Some are very quick checks while others will require more information to help determine the right actions. For all cases, make one change and then reevaluate the performance impact. For example, increasing the streams pool can eliminate buffer queue spills.

Use the information in Table 5 to identify causes that were not explained in the "Initial Tuning Checks and Recommendations" section earlier in this white paper.

Table 5: Identifying the Source or Possible Cause of Flow Control

Last Component In Flow Control	Symptoms	Causes	Recommendations
Capture (C001,C002, ... C00*n*)	Propagation Sender has high percentage in Wait event (other than flow control wait event), low percentage idle. **Note:** For downstream capture configuration for which the capture and apply processes are in the same database, use the symptoms and recommendations for the Apply Reader (AR) component in this table.	Time spent on database wait event (other than flow control wait event) causes reduction of throughput for Propagation Sender.	See the section about "Tuning Wait Events," related to propagation sender.
	Propagation Sender (PS) has low percentage idle, no wait events.	PS is CPU bound at its current workload. Confirm PS is CPU bound by looking at ASH report for "`CPU + Wait for CPU`" events	If the machine is not CPU bound, consider implementing separate streams (see the "Capture too Busy" section).
Propagation Sender (PS)	Propagation Receiver (PR) moderate to high percentage idle, no wait events	Possible insufficient network bandwidth over a WAN to support downstream or local capture propagation.	See "Appendix A: Using Oracle Streams Over a Network" in the companion paper "*Oracle Streams Configuration, Best Practices: Oracle Database 10g Release 2*" [2].
	Propagation Receiver has low percent idle, high percent in Wait Events (other than flow control wait event).	Time spent on database wait event causes reduction of throughput for Propagation Receiver.	See the "Wait Events," section related to propagation receiver.
Propagation Receiver (PR)	Apply Reader (AR) has low percent idle, high percent time in wait events (other than flow control wait event).	Time spent on database wait event causes reduction of throughput for apply reader.	See the "Wait Events," section related to apply reader.
	Apply Reader (AR) has low percent idle, no wait events.	Apply reader is performing apply spill of a transaction. Apply reader is performing transformations.	1. For apply spills, see the section about Large Transactions. 2. Check and tune

Last Component In Flow Control	Symptoms	Causes	Recommendations
			transformation PL/SQL code for performance.
Apply Reader (AR)	Apply Server (AS) near zero percent idle	All apply servers are performing work. The work may involve applying LCRs of transactions and/or performing transformations or DML handlers.	1. Check and adjust apply parallelism. See the "Adjusting the Degree of Apply Parallelism" section. 2. Check and tune PL/SQL code of DML handlers.
	Apply Servers (AS) have high time in idle (> 50%I) *Or* One apply server is busy and the remaining ones are idle	Apply servers are waiting on transactions to commit. A single apply server is applying a spilled transaction. Possibly slow performing DML handlers. A DDL transaction is being applied.	1. For apply spills, see the section about "Handling Large Transactions." 2. Check and tune DML and transformations PL/SQL code that may be significantly I/O bound. 3. For DDL, see the "Handling Data Definition Language (DDL) Statements section.
	Apply Servers (AS) have low percent idle and high percent time in wait events.	The apply servers are spending time waiting on database wait events.	For apply servers, see the Wait Events section.

Notes:

- For downstream capture, configurations in which capture and apply share the same queue, you will not see propagation sender or propagation receiver, and there will only be one buffered queue shown in the STRMMON output. Use Table 5 in the same way as for local capture, but with these components removed.
- Percent idle for the apply servers is based on the number of apply servers specified by the `PARALLELISM` apply parameter. If you set the `PARALLELISM` parameter to 2, the maximum percent idle will be 200. If you set the `PARALLELISM` parameter to 4, then the maximum percent idle is 400.
- The apply servers should always show `0%F` flow control for a normal Oracle Streams configuration.

Flow Control Example for Downstream Capture

This section provides instructions that show how to implement the methodology described in the previous section about "Flow Control" using the STRMMON output shown in Example 2:

Example 2: Sample STRMMON Output for Downstream Capture

```
2008-06-17 12:52:46 || STRM10g8-> | LOG 1M | NET 1K 0 | <F> C001 1262 1256
8min  <0%I 61%F ->  | Q51059 1256 0 |  -  A001 0 0 4min AR: <0%I 0%F ->  AS(2)
<195%I 0%F ->  | MEM 16 % 1024M
```

Step 1: Find the last component showing flow control. For example, in the STRMMON output in Example 2:

<F> C001 1262 1256 8min <0%I 61%F ->

Step 2: Locate the next process component, which should show no flow control:

A001 0 0 4min AR: <0%I 0%F ->

Step 3: Match the identified component in step 2 to the symptoms listed in the *Symptoms* column:

"Apply Reader (AR) has low percent idle, no wait events"

Step 4: From Table 5 above, the following probable causes:

Apply reader is performing apply spills

Because the LCR and transaction rates per second for the Apply process are 0, it indicates that an apply spill may have occurred. Confirm this by executing the query for detecting apply spills in the "Handling Large Transaction" section.

Step 5: From Table 5 above, the recommendation for apply spills is to see the "Handling Large Transaction" section.

Flow Control Example for Local Capture Starting at the Propagation Sender (PS) Process

In Example 3, flow control occurs at the propagation sender (PS) process. The next process component is propagation receiver (PR). Using the methodology, you should conclude that it is necessary to examine the network bandwidth between PS and PR processes.

Example 3: Flow Control Starting at the Propagation Sender (PS)

```
2008-03-17 15:38:43 || STRM10g5-> | LOG 5M | NET 3M 3M |  -  C001 1073 734
4min <61%I 0%F ->  | Q53088 734 0 | PS01 738 0 0 <31%I 58%F ->  | MEM 68 %
256M || STRM10g6-> | LOG 5M | NET 3M 50 | PR01 743 <42%I 0%F ->  |
Q53478 743 0 | Q51358 0 0 | Q51211 0 0 | Q51070 0 0 |  -  A001 740 2 4min AR:
<58%I 0%F ->  AS(8) <584%I 0%F 7%"events in waitclass Other">  | MEM 45 %
256M
```

Apply Bottlenecks <B>

The STRMMON output indicates that there are bottlenecks by using the `<B>` indicator on either the capture or apply processes.

In the STRMMON output shown in Example 4, the `<B>` indicator in front of A001 indicates the apply process may be a bottleneck.

Example 4: STRMMON Output Showing Bottlenecks

```
2008-03-17 09:56:48 || STRM10g5-> | … | <F> C001 13 9 8min <0%I 34%F
65%"Streams AQ: enqueue blocked on low memory">  | Q53088 9 8 | PS01 0 0 0<4%I
0%F ->  | MEM 87 % 256M || STRM10g6-> | … | PR01 0 <0%I 0%F 96%"jobq slave
wait">  | Q53478 0 0 <B> A001 0 0 4min AR: <97%I 0%F ->  AS(8) <409%I 0%F
390%"log file switch (archiving needed)">  | MEM 91 % 256M
```

Addressing Apply Process Bottlenecks

If the `<B>` indicator is in front of the apply process (`A00`*`n`*), it signifies that there are more than 10 transactions that the apply process has not yet assigned or scheduled indicating that the apply process is the main bottleneck of the streams flow. The bottleneck can be the result of apply spills, wait events or slow apply servers.

In Example 5, the apply process has the `<B>` indicator and the apply server processes (8 of them) are spending 390% (or almost 50% per apply server) on the wait event "`log file switch (archive needed).`" For this specific example, address the wait event by checking for I/O issues or adding more redo log groups. Also, see the *Oracle Performance and Tuning Guide 10g Release 2 (10.2)* [6] documentation for further recommendations and MAA Redo Log File configuration best practices in the *Oracle Database High Availability Best Practices 10g Release 2 (10.2)* [8].

Wait Events

Wait events serve as an indicator for when processes are waiting on a resource such as a latch, enqueue, IO or CPU. There are two broad classes of database wait events:

- idle – waiting for work and
- non-idle – waiting on some resource before completing its work.

See Section 10.2.2 of the *Oracle Performance and Tuning Guide 10g Release 2 (10.2)* [6] documentation for more information about wait events.

When a given Oracle Streams component is encountering a non-idle wait event, components *upstream* in the Oracle Streams process flow may show higher time in flow control. Components *downstream* will show higher time in idle. Overall throughput will drop depending upon how much time that component spends on a given wait event.

For some wait events, the solution can be straight forward and simple to implement. For others, more information may be required to determine the right course of actions.

The following is a list of general recommended steps for analyzing and addressing wait events:

1. Review the STRMMON output for repeated displays of wait events and note the timestamps. Also note the instance name for the component that is encountering the wait event.
2. Generate an AWR snapshot report using snapshot Ids that contain the timestamps and instance from step 1. Multiple snapshot Ids might be necessary.
3. Generate an ASH report for the same time period.
4. Review the AWR and ASH reports and correlate with wait events from the STRMMON report.. In the ASH report, review the "`Activity Over Time`" section to correlate with the STRMMON timestamps.
5. Generate an ADDM report using the same AWR snapshot IDs (time period) used in step 1. ADDM can provide analysis and recommendations (ACTIONS) and estimated improvement gains if the action is implemented. Running this report can provide good insight and recommendations for many wait events.
6. After performing steps 1 through 5, and no course of action from ADDM is provided, see the *Oracle Performance and Tuning Guide 10g Release 2 (10.2)* [6] documentation for more information about specific wait events.

In addition to the above, the Oracle Streams Health Check has a "`STREAMS PROCESS Wait Analysis`" section at the bottom of its report. Each Oracle Streams process has its own table showing what specific wait events for which each process spent time waiting. For all wait events where the "`BUSY`" column is "`YES`", then tune for that wait event to improve performance if the percentage of time is significant (greater than 5 or 10%). See the *Oracle Performance and Tuning Guide 10g Release 2 (10.2)* *[6]* for more information about wait events.

The following list summarizes some of the Oracle Streams specific wait events. Some wait events include recommended actions you can take, while others may require no action.

- **Streams capture: resolve low memory condition (Class: Configuration) or Streams AQ: enqueue blocked on low memory (Class: Configuration)**
  The following example provides STRMMON output showing the capture process (C001) spending time (in this snapshot) on a wait event:

```
C001 18 12 7min <0%I 0%F 70%"Streams AQ: enqueue blocked on low memory">
```

This is equivalent to flow control but it's useful to have additional information about the low memory condition. In this case, capture is spending 70 percent on the wait event "`Streams AQ: enqueue blocked on low memory`". This wait event causes the capture process to wait until there is sufficient memory to enqueue more LCRs. This particular event indicates that the buffered queue in the streams pool on the source database is becoming full. If this wait event is repeated throughout the STRMMON output, this should also correlate as one of the top 5 wait events in the AWR report. The ASH report should also show this event in several time slot samples at the bottom of its report. See the *Oracle Performance and Tuning Guide 10g Release 2 (10.2)* [6] for information about using AWR and ASH.

To resolve the above wait event, do the following:

1. Address components downstream of the capture process that are in flow control or are incurring high wait events, or are bottlenecked *first*.
2. If there are no components found in step 1, increase the streams pool by following the steps as described in the "Adjusting the `STREAMS_POOL_SIZE`" section.

- **Streams: apply reader waiting for DDL to apply (Class: Application)**

  When Oracle Streams processes a DDL statement, there may also be DML LCRs for the particular object to which Oracle Streams will apply the DDL. The apply reader will not assign any DML transactions for objects on which there is an outstanding DDL to be applied (called a DDL Barrier). Once the DDL change has been applied, any outstanding DML transactions for that object are processed and applied.

  Recommendations: "Handling Data Definition Language (DDL) Statements" section for a discussion and recommended steps.

- **Streams capture: waiting for archive log (Class: Other)**

  The capture process (dependent on LogMiner) is waiting for an archived redo log to arrive for downstream capture, or to be generated for local capture.

  Recommendations:

  1. For downstream capture, ensure that the redo transport service is configured optimally. See Appendix A of the companion white paper "*Oracle Streams Configuration, Best Practices: Oracle Database 10g Release 2*" [2] for configuration information.
  2. For local capture, tune the I/O subsystem for the online redo logs such that they can support I/O from the LGWR and LogMiner processes.

- **Streams AQ: enqueue blocked due to flow control (Class Other)**

  In the STRMMON output, this wait event may show up instead of the flow control percentage (`nn%F`) shows zero. This wait event indicates that capture is blocked because flow control is in effect. This wait event is often seen for the capture process, where capture shows a very low percent idle, and no time in flow control. However, the capture process is operating under flow control.

  Recommendations: Treat this wait event the same as flow control and follow the methodology in Table 5.

I/O and Other Wait Events

Oracle Streams can also encounter the following wait events.

- **log file sequential read**

  This wait event indicates that a process, typically LogMiner is waiting sequential I/Os on the redo logs (or archive logs) to complete.

  Recommendations: Tune the I/O subsystem where the redo logs and archive log files reside. See the *Oracle Performance and Tuning Guide 10g Release 2 (10.2)* [6] documentation for further recommendations and MAA Redo Log File configuration best practices in the *Oracle Database High Availability Best Practices 10g Release 2 (10.2)* [8].

- **db file sequential read**

  This wait event indicates that a process is waiting to complete read I/Os to one or more indexes. If the process is spending a considerable amount of time (> 15%) in this wait event, then tuning the I/O subsystem may be necessary.

  Recommendations: Tune the I/O subsystem where the indexes and tables reside. See the *Oracle Performance and Tuning Guide 10g Release 2 (10.2)* [6] documentation for further recommendations.

- **db file scattered read**

  This wait event indicates that a process is spending time waiting for full table scan I/Os to complete. Oracle Streams processes typically should not encounter this wait event except in cases where a DML PL/SQL handler might have SQL whose execution plans perform a full table scan.   An apply process can also perform full table scans if you used the `DBMS_APPLY_ADM.SET_KEY_COLUMNS` procedure to set key columns on the target tables and the target tables have no indexes on columns that were set.

Recommendations: Create indexes for key columns and tune the SQL statements with in DML handlers for optimized execution plans.

- **latch free**

  This wait event indicates that an in memory resource is under contention. The specific in memory resource is not specified in this wait event and further analysis is required.

  Recommendations: See the *Oracle Performance and Tuning Guide 10g Release 2 (10.2)* [6] documentation for further recommendations.

- **rdbms ipc message**

  This wait event indicates an idle event that may show up in the Oracle Streams Health Check report, AWR report, and ASH report. It indicates that processes are waiting for messages to be sent and delivered across the interprocess communication channels.

  Recommendations: None.

Case 2: Capture Rates Are Near Zero, Latency Is Increasing

This case focuses on the capture process and its interaction with the LogMiner process. Oracle Streams uses LogMiner to mine the redo for all changes for which Streams is interested. LogMiner statistics are not displayed in the STRMMON output but you can infer some of the LogMiner characteristics based on the performance characteristics of the capture process.

This section contains the following topics:

- ❑ Rate of change by the application
- ❑ Capture rates near zero
- ❑ Capture too busy

In the Case 2 scenario, the percent idle for capture is typically 40% or higher, waiting for the LogMiner to deliver LCRs, as shown in the following snippet of STRMMON output in Example 5:

Example 5: STRMMON Output for Case 2

```
2008-03-17 11:17:35 || STRM10g5-> | LOG 6M | NET 3M 3M | <F> C001 1226 840
5min <42%I 2%F -> | Q53088 840 0 | PS01 822 0 0 <17%I 70%F -> | MEM 62 %
256M || STRM10g6-> | LOG 5M | NET 3M 49 | PR01 843 <29%I 0%F -> | Q53478
843 0 | Q51358 0 0 | Q51211 0 0 | Q51070 0 0 | - A001 858 2 5min AR: <48%I 0%F ->
AS(8) <523%I 0%F 6%"events in waitclass Other"> | MEM 64 % 256M
2008-03-17 11:18:35 || STRM10g5-> | LOG 5M | NET 3M 3M | - C001 1087 745 6min
<56%I 0%F-> | Q53088 746 0 | PS01 761 0 0 <35%I 56%F -> | MEM 68 % 256M ||
STRM10g6-> | LOG 4M | NET 3M 57 | PR01 751 <48%I 0%F -> | Q53478 751 0 |
Q51358 0 0 | Q51211 0 0 | Q51070 0 0 | - A001 740 2 6min AR: <63%I 0%F -> AS(8)
<538%I 0%F 46%"log file switch completion"> | MEM 65 % 256M
```

The following observations of the capture process (C001) can be made:

- The capture process has very low percent flow control (0%F in the second line).
- The capture latency is increasing from 5 minutes to 6 minutes.
- The capture process has moderately high percent idle, 42% and 56%.

The low percentage of flow control and higher percentage of idle for the capture process implies that capture is waiting on LogMiner. Consider the following situations and recommendations:

- Application Rate of Change
- Capture Rates Near Zero

Application Rate of Change

Check the rate of change of the application. As long as the capture (and apply) latency is small (a couple of minutes or less), then the capture process is keeping up and it is not necessary to perform further tuning of the capture process.

Capture Rates Near Zero

Aside from the rate of change made by the application, you can observe that the LCRs capture and enqueued per second may be near or at zero and capture latency is increasing within STRMMON. The following checklist describes reasons for this condition:

❑ **Capture/LogMiner Waiting on Archived Redo Logs**

This situation typically occurs in a downstream capture database configuration where the source database is configured to ship redo (using the Oracle Data Guard redo transport services) to the downstream capture database. This service can ship redo either using the `LGWR` or `ARCH` transport methods. LGWR sends redo in either `SYNC` or `ASYNC` mode, depending on how the source database initialization parameter `LOG_ARCHIVE_DEST_`*`n`* is configured. See the "*Oracle Streams Configuration, Best Practices: Oracle Database 10g Release 2*" [2] MAA white paper for recommendations about configuring downstream capture.

For downstream capture, if the capture process is not configured for real-time mining, then it is normal to see the rates for capture fall to zero once capture has finished capturing changes from the last archive log that arrived. In addition, the capture `STATE` column in the `GV$STREAMS_CAPTURE` view contains "`WAITING FOR REDO, LAST SCN MINED <`*`scn#`*`>`".

If you configured real-time mining for the downstream capture process and you observe that capture rates fall to zero often, then you may need to tune the redo transport services. See the *Oracle Data Guard Redo Transport & Network Best Practices Oracle Database 10g Release 2* [4] MAA white paper for details.

❑ **Large DDLs Transactions and LogMiner Spills**

Large DDL and Parallel DDL can lead to LogMiner spills and reduced Oracle Streams capture rates. See the "Detect and Reduce LogMiner Spills" section.

Capture Too Busy

If the latency of capture continues to grow, is not in flow control, is not spending any time in wait events, and its percentage of time idle is near or at zero, then the capture process is most likely the bottleneck. This can occur if the capture process is doing "full" rule evaluations. Query the `TOTAL_FULL_EVALUATION` column of the `GV$STREAMS_CAPTURE` view. Full evaluations require the capture process to evaluate the LCR beyond the schema name and object name to determine if the LCR should be captured.

The snippet of output from STRMMON in Example 6 shows the capture process near zero percent idle in a downstream capture configuration.

Example 6: STRMMON Output Shows the Capture Process Near 0% Idle

```
2008-06-17 13:09:19 || STRM10g8-> | LOG 1M | NET 1K 0 | <B> C001 2000 1971
23min <2%I 0%F ->  | Q51059 1971 0 | <B> A001 2396 23 23min AR: <37%I 11%F ->
AS(2) <14%I 0%F ->  | MEM 17 % 1024M
2008-06-17 13:09:49 || STRM10g8-> | LOG 1M | NET 1K 0 | <B> C001 2220 2157
12min <2%I 0%F ->  | Q51059 2157 0 | <B> A001 2260 22 12min AR: <48%I 0%F ->
AS(2) <18%I 0%F ->  | MEM 17 % 1024M
```

In the STRMMON output, notice that the capture process is 2% idle and not in flow control in both lines of output (highlighted in yellow in the STRMMON output). Also, notice that the capture process has a `<B>` indicator indicating that it is a potential bottleneck.

Consider implementing multiple streams by configuring multiple Oracle Streams capture-propagation-apply paths and partition the application tables or schemas across each streams path:

- This recommendation only applies to source and target systems with sufficient CPUs for each server.
- This recommendation is most applicable to Oracle Streams local capture configurations because a downstream capture configuration can have only one capture process configured for real-time mining.

> **Note:** You must increase the streams pool on both source and target databases when implementing multiple streams paths. See the "Adjusting the `STREAMS_POOL_SIZE` Parameter" section to determine the proper sizing.

In one particular case study, configuring multiple streams where each streams path replicated specific groups of tables significantly improved throughput and reduced the overall latency of Oracle Streams.

Consider the following LDAP case study:

- The OID/LDAP schema has 182 tables being replicated.
- There are 3,000 simulated clients accessing the LDAP service running on the database.
- The DML workload consists of 63% `INSERTs`, 26.5% `UPDATEs`, 10.5% `DELETEs`.
- LDAP operations include `SEARCH`, `BIND`, `COMPARE`, `MODIFY`,`ADD` and `DELETE`.
- There are 100,000 LDAP entries that the clients can query or modify.

Two different Oracle Streams configurations were used in this case study. The first configuration used a single Oracle Streams capture, propagation, and apply path. The second configuration used three separate streams. For the multiple streams configuration, the tables are partitioned by table name and assigned to each of the specific streams, and the `DBMS_STREAMS_ADM.MAINTAIN_TABLE` PL/SQL procedure was used. There were no queue or apply spills in either the single or multiple Oracle Streams configurations.

Table 6 compares the single and multiple Oracle Streams configuration scenarios.

Table 6: Comparison of Single and Multiple Oracle Streams Configuration Scenarios

Capture				**Apply**			
Process Name	LCR Captured /Second	LCR Enqueued /Second	Latency (Seconds)	Process Name	LCRs Applied/ Second	Transactions Applied/ Second	Latency
Single-Path Oracle Streams							
C001	2,351	2,151	2,080	A001	2,144	154	2,082
Multiple-Path Oracle Streams (with increased `STREAMS_POOL_SIZE`)							
C001	2,460	2,107	801	A001	2,100	262	804
C002	983	281	0	A002	280	93	-37
C003	2,582	2,123	1	A003	2,120	336	-34

The table points out several differences between Oracle Stream configurations using a single path versus multiple paths for the same set of tables and workload:

- The single stream sustains a capture and enqueue rate of 2,351 and 2,151 LCRs per second, respectively.
- The single stream sustains an apply rate for LCRs and transactions applied per second of 2,144 and 154, respectively.

- The capture latency (or time lag) is 2,080 seconds (34 minutes) and the apply latency is 2,082 seconds. Nearly all of the latency is due to capture because the apply process is only 2 seconds behind.
- The aggregate rate of capture across all three streams (C001 – C003) is 6025 LCRs captured per second and 4511 LCRs enqueued per second.
- The aggregate rate for the apply rate is 4500 LCRs per second and 691 transactions per second.
- The capture latency across all capture (C001 – C003) is 802 seconds (13 minutes) with apply latency only 2 seconds behind. The A002 and A003 latencies of –37 and –34 seconds is effectively 0. That is, the apply processes have no lag time.

The distribution of LCRs captures, enqueued and applied across each of the three streams is not evenly distributed. This has much to do with the application. Not all applications may lend themselves to this type of Oracle Streams configuration but it serves to show the flexibility of Oracle Streams to achieve optimum performance.

CONCLUSION

By using the recommendations in the *Oracle Streams Configuration Best Practices: Oracle Database 10g Release 2* [2] white paper to set up Oracle Streams and then following the performance methodology prescribed in this white paper, you can achieve high throughput with Oracle Streams.

APPENDIX A: SAMPLE OUTPUT OF STRMMON

The latest version of the STRMMON utility can be downloaded from Oracle*MetaLink* Note: 296065.1. There are two examples of STRMMON output shown in this appendix: Local Capture and Downstream capture.

Local Capture Example

The following Local Capture output shows the flow rates for capture, propagation and apply. Both databases instances (`STRM10g5` and `STRM10g6`) are displayed.

```
STREAMS Monitor, v 2.5  Copyright Oracle Corp. 2002, 2005.
Interval = 60, Count=60
Logon=sys@streamssrc10g_halinux02.us.oracle.com ORACLE 10.2.0.4.0
Streams Pool Size = 256M
LOG : <redo generated per second>
NET: <client bytes per second> <dblink bytes per second>
Cnnn: < LCRs captured per second> < LCRs enqueued per second> <capture latency>
MEM : <percent of memory used> % <Streams pool size>
PRnn: <messages received per second>
Qn  : <messages enqueued per second> <messages spilled per second>
PSnn: <LCRs propagated per second> <bytes propagated per second>
Annn: < LCRs applied per second> <transactions applied per second> <dequeue latency>
<F>: flow control in effect
<B>: potential bottleneck
AR: apply reader
AS(n): n number of apply servers
<n%I n%F n%nn>: <idle wait events percentage> <flow control wait events percentage> <other
wait event percentage and name>
nn->: database instance name

STREAMS Monitor, v 2.5  Copyright Oracle Corp. 2002, 2005.
Interval = 60, Count=60
Logon=sys@streamsdest10g_halinux06.us.oracle.com ORACLE 10.2.0.4.0
Streams Pool Size = 256M
LOG : <redo generated per second>
NET: <client bytes per second> <dblink bytes per second>
Cnnn: < LCRs captured per second> < LCRs enqueued per second> <capture latency>
MEM : <percent of memory used> % <Streams pool size>
PRnn: <messages received per second>
Qn  : <messages enqueued per second> <messages spilled per second>
PSnn: < LCRs propagated per second> <bytes propaged per second>
Annn: < LCRs applied per second> <transactions applied per second> <dequeue latency>
<F>: flow control in effect
<B>: potential bottleneck
AR: apply reader
```

```
AS(n):  n number of apply servers
<n%I n%F n%nn>: <idle wait events percentage> <flow control wait events percentage> <other
wait event percentage and name>
nn->: database instance name

2008-03-17 15:32:40 || STRM10g5-> | LOG 7M | NET 4M 1M |  -  C001 443 290 34sec <30%I
0%F ->  | Q53088 290 0 | PS01 290 0 0 <53%I 28%F ->  | MEM 40 % 256M || STRM10g6-> |
LOG 1M | NET 1M 50 | PR01 295 <0%I 0%F 99%"db file sequential read"> | Q53478 295 0 |
Q51358 0 0 | Q51211 0 0 | Q51070 0 0 |  -  A001 260 0 31sec AR: <92%I 0%F ->  AS(8)
<722%I 0%F ->  | MEM 21 % 256M

2008-03-17 15:33:41 || STRM10g5-> | LOG 9M | NET 5M 2M |  -  C001 941 637 1min <1%I
0%F ->  | Q53088 637 0 | PS01 637 0 0 <20%I 56%F ->  | MEM 62 % 256M || STRM10g6-> |
LOG 4M | NET 2M 49 | PR01 638 <52%I 0%F ->  | Q53478 638 0 | Q51358 0 0 | Q51211 0 0 |
Q51070 0 0 |  -  A001 622 1 1min AR: <67%I 0%F ->  AS(8) <620%I0%F ->  | MEM 31 % 256M

2008-03-17 15:34:41 || STRM10g5-> | LOG 9M | NET 6M 2M |  -  C001 948 644 1min <2%I
0%F ->  | Q53088 644 0 | PS01 609 0 0 <20%I 57%F ->  | MEM 68 % 256M || STRM10g6-> |
LOG 4M | NET 2M 50 | PR01 634 <54%I 0%F ->  | Q53478 634 0 | Q51358 0 0  | Q51211 0 0 |
Q51070 0 0 |  -  A001 641 1 1min AR: <68%I 0%F ->  AS(8) <616%I 0%F ->  | MEM 33 %
256M
```

Downstream Capture Example

In the following Downstream Capture example, the capture and apply processes run on the same database instance (`STRM10g8`). The capture and apply processes share the same queue. Therefore, there is no propagation. In this example, the source database sends redo data to the downstream capture database.

```
STREAMS Monitor, v 2.6  Copyright Oracle Corp. 2002, 2005.
Interval = 30, Count=240
Logon= @ ORACLE 10.2.0.4.0
Streams Pool Size = 1024M

LOG : <redo generated per sec>
NET: <client bytes per second> <dblink bytes per second>
Cnnn: <LCRs captured per second> < LCRs enqueued per second> <capture latency>
MEM : <percent of memory used> % <Streams pool size>
PRnn: <messages received per second>
Qn  : <messages enqueued per second> <messages spilled per second>
PSnn:  < LCRs propagated per second> <bytes propagated per second>
Annn:  < LCRs applied per second> <transactions applied per second> <dequeue latency>
<F>: flow control in effect
<B>: potential bottleneck
AR: apply reader
```

AS(*n*): *n* number of apply server

<*n*%I *n*%F *n*%*nn*>: *<idle wait events percentage> <flow control wait events percentage> <other wait event percentage and name>*

nn->: database instance name

```
2008-05-16 18:46:37 || STRM10g8-> | LOG 764K | NET 3K 0 |  -  C001 2841 1509 8min <16%I 0%F ->  | Q51059 1510 0 | <B> A001 1301 12 8min AR: <68%I 0%F ->  AS(2) <103%I 0%F -> | MEM 12 % 1024M

2008-05-16 18:47:07 || STRM10g8-> | LOG 1M | NET 3K 0 | <B> C001 2236 2203 8min <0%I 0%F ->  | Q51059 2203 0 | <B> A001 2026 20 8min AR: <58%I 0%F ->  AS(2) <32%I 0%F ->  | MEM 13 % 1024M

2008-05-16 18:47:37 || STRM10g8-> | LOG 1M | NET 2K 0 | <B> C001 2112 2082 8min <0%I 0%F ->  | Q51059 2082 0 | <B> A001 2248 22 8min AR: <58%I 0%F ->  AS(2) <4% I 0%F ->  | MEM 14 % 1024M

2008-05-16 18:48:07 || STRM10g8-> | LOG 1M | NET 699 0 | <B> C001 1851 1825 9min  <0%I 0%F ->  | Q51059 1825 0 |  -  A001 1864 18 9min AR: <58%I 0%F ->  AS(2) <13%I 0%F ->  | MEM 14 % 1024M
```

APPENDIX B: ORACLE STREAMS ALERT.LOG MESSAGES

The messages described in this appendix are written to the `ALERT.LOG` file if these events occur. You can use these messages for parsing and alerting purposes.

Message: One of the following messages is logged when a large transaction or a long-running transaction is detected by the capture processes.

```
Wed Oct 29 06:23:35 2008
C001: large txn detected (45753 LCRs), xid: 0x0037.005.0002bc6b
Wed Oct 29 06:25:50 2008
C001: large txn committed, xid: 0x0037.005.0002bc6b
```

```
Thu Oct 30 22:48:47 2008
C001: long running txn detected, xid: 0x0036.045.000038e5
Thu Oct 30 22:56:19 2008
C001: long txn committed, xid: 0x0036.045.3000038e
```

Recommendation: Follow the recommendations in the "Handling Large Transactions" and "Handling Long-Running Transactions" sections.

Message: The following message indicates that the Oracle Streams Data Dictionary is missing.

```
Missing MVDD for object ID nnn
```

Check the `ALERT.LOG` on the target database (where the apply process is running) for the '`Missing MVDD for object ID` *`nnn`*" message. This message indicates that maintenance is being performed on specific tables or schemas on the source database. Until the Oracle Streams Data Dictionary is updated, the apply process cannot apply the pending changes. The source database information is found in the accompanying detail messages for the Missing MVDD.

Recommendation: Obtain the object ID from the message in the `ALERT.LOG,` and issue the following command on the source database:

```
EXECUTE
DBMS_CAPTURE_ADM.SET_PARAMETER(' <capture_name>' ,' _SEND_STREAMS_DIC
TIONARY' ,' <object_id>' );
```

Immediately after you run this command, the capture processes transmit the LCRs for the Oracle Streams Data Dictionary for that object ID.

Message: The following message occurs if the capture process stops. The capture name and error number are included in the error message.

```
STREAMS capture process \"%s\" aborted with ORA-%s
```

Recommendation: See the Troubleshooting section in the "*Oracle Streams Configuration, Best Practices: Oracle Database 10g Release 2*" [2] MAA white paper.

Message: The following message occurs if the apply aborts. The capture name and error number will be filled in at the time of the error.

```
STREAMS apply process \"%s\" aborted with ORA-%s
```

Recommendation: See the Troubleshooting section in the "*Oracle Streams Configuration, Best Practices: Oracle Database 10g Release 2*" [2] MAA white paper.

Message: The following message occurs if the propagation job fails after 16 attempts.

```
STREAMS propagation process \"%s\" aborted after 16 failures
```

Recommendation: See the Troubleshooting section in the "*Oracle Streams Configuration, Best Practices: Oracle Database 10g Release 2*" [2] MAA white paper.

APPENDIX C: ORACLE*METALINK* DOWNLOADS FOR ORACLE STREAMS

Oracle*MetaLink* note 749079.1 provides the following documents that you can download and cut and paste into scripts:

- **`wp_apply.txt`**— Provides sample SQL statements of queries that you can cut and paste into scripts for monitoring various Oracle Streams views such as `GV$PROPAGATION_RECEIVER`, `GV$STREAMS_APPLY_READER`, `GV$STREAMS_APPLY_COORDINATOR`; `GV$STREAMS_APPLY_SERVER`, `GV$STREAMS_TRANSACTION`, and `DBA_APPLY_PROGRESS`.
- **`wp_capture.txt`**—Provides sample SQL statements of queries that you can cut and paste into scripts for monitoring various Oracle Streams views such as `GV$LOGMNR_STATS`, `GV$STREAMS_CAPTURE`, `GV$STREAMS_TRANSACTION`, `GV$PROPAGATION_SENDER`, and `GV$BUFFERED_QUEUES` to view the Streams buffered queues. This file also includes propagation examples.
- **`wp_apply_lob_err_handler.txt`**—Provides a working example for implementing the `ASSEMBLE_LOBS` parameter to improve LOB apply performance. See the "Using LOB Data Type with Streams" section for more details.

APPENDIX D: RUNNING THE ORACLE STREAMS HEALTH CHECK SCRIPT

The Oracle Streams Health Check script provides information about the current configuration of Oracle Streams, and must be run on all databases where Oracle Streams is configured. The Oracle Streams Health Check script and instructions are available from Oracle*MetaLink* Note 273674.1.

Also, see the *Oracle Streams Configuration Best Practices: Oracle Database 10g Release 2* [2] white paper for more help with configuration and troubleshooting.

REFERENCES

1. Oracle Maximum Availability Architecture
   http://www.oracle.com/technology/deploy/availability/htdocs/maa.htm

2. *Oracle Streams Configuration Best Practices: Oracle Database 10g Release 2*
   http://www.oracle.com/technology/deploy/availability/pdf/maa_10gr2_streams_configuration.pdf

3. *Oracle Streams Concepts and Administration 10g Release 2 (10.2)*
   http://otn.oracle.com/pls/db102/db102.to_toc?partno=b14229

4. *Oracle Database 10g Release 2 Best Practices: Data Guard Redo Transport and Network Configuration* white paper
   http://www.oracle.com/technology/deploy/availability/pdf/MAA_WP_10gR2_DataGuardNetworkBestPractices.pdf

5. *Oracle Database PL/SQL Packages and Types Reference 10g Release 2 (10.2)*
   http://otn.oracle.com/pls/db111/db111.to_toc?partno=b28419

6. *Oracle Performance and Tuning Guide 10g Release 2 (10.2)*
   http://otn.oracle.com/pls/db102/db102.to_toc?partno=b14211

7. *Oracle Database Reference 10g Release 2 (10.2)*
   http://otn.oracle.com/pls/db102/db102.to_toc?partno=b14237

8. *Oracle Database High Availability Best Practices 10g Release 2 (10.2)*
   http://otn.oracle.com/pls/db102/db102.to_toc?partno=b25159

Oracle Streams Performance Tuning Best Practices: Oracle Database 10*g* Release 10.2
November 2008
Author: Darryl Presley and Pat McElroy
Contributors: Nimar Arora, Stephan Haisley, Viv Schupmann, Lawrence To, Byron Wang, Lik Wong

Oracle USA, Inc.
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com