INVESTIGATING AND RESOLVING "LOG FILE SYNC" WAIT EVENTS

CASE STUDY

ABSTRACT

"Log file sync" wait events in Oracle databases indicate performance issues related to the commit process, where sessions wait for the Log Writer (LGWR) to write redo log entries to disk. High wait times for this event can significantly impact database performance, especially in high-transaction environments. This case study explores five different scenarios where "log file sync" wait events were identified as bottlenecks, providing detailed analyses and resolution strategies.

Aman Pandey
@LinkedIn

Scenario 1: High Commit Frequency Misdiagnosed Due to Undersized Log Files

Background:

A banking application exhibited severe slowdowns during peak operational hours, primarily when processing customer transactions. Initially, the database team suspected that a high commit rate was causing excessive "log file sync" waits, leading to degraded performance.

Investigation:

Step 1: Analyzing the AWR Report

The DBA team generated an AWR report during peak transaction periods to identify the top wait events.

Initial AWR Report Excerpt:

```
Top 5 Timed Events
~~~~~~~~~~~~~~~~~~                                     % Total DB Time
Event                              Waits   Time (s)    Avg Wait(ms)   % DB Time
------------------------------ ---------- ----------- -------------- --------
log file sync                    750,000    2,000          2.67          40.0
DB CPU                                            1,200                    24.0
db file sequential read          800,000      480          0.60          12.0
log file parallel write          200,000      240          1.20           6.0
db file scattered read           700,000      420          0.60          10.5
```

Observations:

- "Log file sync" accounts for **40%** of total DB time, indicating a significant performance bottleneck.
- High number of waits suggests that sessions are frequently waiting for LGWR to complete.

Step 2: Evaluating Commit Frequency

To determine if the high "log file sync" waits were due to excessive commits, the team analyzed the commit frequency.

Commit Frequency Analysis:

- **Metric to Calculate: User Calls per Commit/Rollback**

$$\text{Calls per Commit} = \frac{\text{User Calls}}{\text{User Commits} + \text{User Rollbacks}}$$

SQL Query for Commit Analysis:

```
SELECT
    (user_calls.value / (user_commits.value + user_rollbacks.value)) AS calls_per_commit
FROM
    v$sysstat user_calls,
    v$sysstat user_commits,
    v$sysstat user_rollbacks
WHERE
    user_calls.name = 'user calls'
    AND user_commits.name = 'user commits'
    AND user_rollbacks.name = 'user rollbacks';
```

```
CALLS_PER_COMMIT
----------------
              20
```

Interpretation:

- An average of **20 user calls per commit** is significantly below the recommended threshold of **30**, indicating **excessive commit frequency**.

Why This Isn't the Real Issue:

- Although the commit frequency appears problematic, the DBA team recognized that a commit ratio of 20, while suboptimal, should not have led to the severity of "log file sync" waits observed. This finding suggested that while frequent commits contributed to the issue, they were not the sole or primary cause. This led the team to investigate further, particularly focusing on the redo log configuration.

Step 3: Analyzing Redo Log Configuration

The team then examined the size and configuration of the redo log files to ensure that the log switches were not too frequent.

Redo Log Analysis:

```
SELECT GROUP#, BYTES/1024/1024 AS SIZE_MB, STATUS
FROM v$log;
```

```
GROUP#    SIZE_MB    STATUS
------    -------    --------
1         50         INACTIVE
2         50         INACTIVE
3         50         CURRENT
```

Findings:

- Each redo log file is **50 MB** in size.
- Frequent log switches are occurring every **2-4 minutes**, resulting in **20-30 log switches per hour**, which exceeds the recommended frequency.

Resolution:

Step 4: Resizing Redo Log Files

Understanding that undersized redo logs were contributing to frequent log switches and thereby increasing "log file sync" waits, the DBA team decided to **increase the size of the redo log files**.

Steps to Resize Redo Log Files:

1. **Add New Larger Log Files:**

```
ALTER DATABASE ADD LOGFILE GROUP 4 ('/path_to_redo_logs/redo04.log') SIZE 500M;
ALTER DATABASE ADD LOGFILE GROUP 5 ('/path_to_redo_logs/redo05.log') SIZE 500M;
ALTER DATABASE ADD LOGFILE GROUP 6 ('/path_to_redo_logs/redo06.log') SIZE 500M;
```

2. **Drop the old smaller Log Files:**

```
ALTER DATABASE DROP LOGFILE GROUP 1;
ALTER DATABASE DROP LOGFILE GROUP 2;
ALTER DATABASE DROP LOGFILE GROUP 3;
```

3. **Verify the New Log File Configuration:**

```
SELECT GROUP#, BYTES/1024/1024 AS SIZE_MB, STATUS
FROM v$log;
```

Post-Resolution AWR Report Excerpt:

```
Top 5 Timed Events
~~~~~~~~~~~~~~~~~~                                    % Total DB Time
Event                              Waits   Time (s)   Avg Wait(ms)  % DB Time
------------------------------ ---------- ----------- ------------- --------
log file sync                     400,000    800          2.00          25.0
DB CPU                                          1,500                     30.0
db file sequential read           600,000      360        0.60          12.0
log file parallel write           100,000      120        1.20           3.0
db file scattered read            500,000      300        0.60           7.5
```

Outcome:

- **Reduction in "Log File Sync" Waits:** Decreased from **40%** to **25%** of total DB time.
- **Lower Log Switch Frequency:** Reduced from **20-30 switches per hour** to **5-10 switches per hour**, aligning with the recommended frequency.
- **Improved Transaction Throughput:** Enhanced performance during peak transaction periods, leading to faster order processing and improved customer satisfaction.

Additional Consideration: Handling High Commit Frequency in Application Logic

If the high commit frequency is indeed due to application logic, the DBA team should work with developers to reduce the frequency of commits. This can be achieved by implementing batch commits, especially in scenarios where commits are placed inside loops.

Example:

- **Before:** The application code had a commit inside a loop that processed multiple records.

```
FOR record IN cursor LOOP
    -- Process record
    INSERT INTO table VALUES (record);
    COMMIT;
END LOOP;
```

- **After:** The commit was moved outside the loop, batching multiple transactions into a single commit.

```
FOR record IN cursor LOOP
    -- Process record
    INSERT INTO table VALUES (record);
END LOOP;
COMMIT;
```

Impact of the Change:

- **Reduction in "log file sync" Waits:** By reducing the number of commits, the overall system performance improved, with fewer "log file sync" waits.
- **Improved Application Efficiency:** The application processed records faster due to reduced commit overhead.

Limitations:

- **Increased Risk of Data Loss:** While batch commits can reduce wait events, they may increase the risk of data loss in the event of a system failure before the commit is executed.
- **Impact on Application Logic:** Changes to commit frequency may require significant application logic modifications and thorough testing to ensure that transactional integrity is maintained.

Key Learnings:

- **Importance of Redo Log Sizing:** Undersized redo logs can mimic symptoms of excessive commit frequency, leading to misdiagnosis.
- **Comprehensive Analysis:** Always consider multiple factors, such as redo log configuration and I/O performance, before attributing wait events to a single cause.
- **Collaboration with Developers:** In cases where high commit frequency is due to application design, working closely with developers to optimize commit strategies can lead to significant performance improvements.

Scenario 2: Resolving "Log File Sync" Wait Event Due to Slow I/O Performance

Background:

A healthcare provider's Oracle database, used for managing patient records, started showing signs of performance degradation, particularly during high-usage periods. Users reported delays in record updates, leading to slower response times in critical applications. The DBA team suspected that the storage subsystem was unable to keep up with the demands, resulting in increased "log file sync" waits.

Investigation:

Step 1: Analyzing the AWR Report

The DBA team generated an AWR report to analyze the top wait events during peak operational hours.

Initial AWR Report Excerpt:

```
Top 5 Timed Events
~~~~~~~~~~~~~~~~~~                                  % Total DB Time
Event                              Waits   Time (s)    Avg Wait(ms)   % DB Time
------------------------------ ---------- ----------- -------------- --------
log file sync                     600,000     1,800         3.00          45.0
log file parallel write           250,000       600         2.40          15.0
DB CPU                                             1,100                   27.5
db file sequential read           750,000       450         0.60          11.3
```

Observations:

- "Log file sync" accounts for **45%** of total DB time, a significant bottleneck.
- The "log file parallel write" event, which measures the time LGWR spends waiting for the I/O operation to complete, also shows elevated wait times, suggesting potential I/O performance issues.

Step 2: Comparing "Log File Sync" with "Log File Parallel Write"

To determine if the I/O subsystem was indeed the problem, the team compared the average wait times for "log file sync" and "log file parallel write." According to Oracle, if a significant portion of "log file sync" time is spent on "log file parallel write," it indicates that I/O performance is a likely culprit.

Sample Output:

```
EVENT                      TOTAL_WAITS   TIME_WAITED   AVG_WAIT
-------------------------  -----------   -----------   ---------
log file sync                 600,000      1,800 sec     3.00 ms
log file parallel write       250,000        600 sec     2.40 ms
```

Interpretation:

- The average wait time for "log file parallel write" (2.40 ms) is close to that of "log file sync" (3.00 ms), indicating that a significant portion of the "log file sync" wait is due to I/O delays.
- This strongly suggests that the performance bottleneck is related to the I/O subsystem, specifically the disks handling the redo logs.

<u>Step 3: Assessing the I/O Subsystem</u>

The DBA team reviewed the storage configuration and discovered that the redo logs were stored on a RAID-5 array. RAID-5 is known for its slower write performance due to parity calculations, which can exacerbate I/O wait times in high-transaction environments.

Observations:

- RAID-5 was likely contributing to the slow I/O performance, causing increased "log file parallel write" waits, and consequently, "log file sync" waits.

Resolution:

<u>Step 4: Moving Redo Logs to SSD Storage</u>

To address the slow I/O performance, the DBA team decided to move the redo logs from the RAID-5 array to a high-performance SSD storage solution, known for its fast write capabilities.

Post-Resolution AWR Report Excerpt:

```
Top 5 Timed Events
~~~~~~~~~~~~~~~~~~                                     % Total DB Time
Event                              Waits    Time (s)   Avg Wait(ms)   % DB Time
------------------------------ ---------- ----------- -------------- --------
log file sync                    300,000      700             2.33       23.3
log file parallel write          120,000      280             2.33        9.3
DB CPU                                          1,400                    30.0
db file sequential read          600,000      360             0.60       12.0
```

Outcome:

- **Reduction in "Log File Sync" Waits:** The percentage of DB time spent on "log file sync" was reduced from **45%** to **23.3%**.
- **Improved I/O Performance:** The "log file parallel write" wait time decreased, demonstrating the effectiveness of moving redo logs to SSD storage.
- **Enhanced Application Response Time:** User reports indicated significantly improved performance in record updates, leading to better overall system responsiveness.

Limitations:

- **Cost of SSDs:** High-performance SSD storage is generally more expensive than traditional HDD or RAID configurations, which can increase operational costs.
- **Wear and Tear:** SSDs have limited write endurance compared to traditional disks, meaning they may require more frequent replacement in write-heavy environments.

Key Learnings:

- **Impact of Storage Configuration:** The type of storage used for redo logs can have a significant impact on "log file sync" waits. RAID-5, while offering data protection, may not be suitable for high-transaction workloads.
- **I/O Optimization:** Moving critical files like redo logs to SSD storage can dramatically improve I/O performance and reduce wait events.
- **Cost-Benefit Analysis:** While SSDs offer superior performance, the associated costs and potential longevity issues must be carefully considered.

Scenario 3: Resolving "Log File Sync" Wait Event Due to CPU Contention on the LGWR Process

Background:

A telecom company experienced intermittent slowdowns in their billing system, particularly during the end-of-month billing cycles when transaction volumes peaked. The database team observed that "log file sync" waits spiked during these periods, leading to delays in processing customer bills.

Investigation:

Step 1: Analyzing the AWR Report

The DBA team generated an AWR report to assess the top wait events during the problematic billing cycles.

Initial AWR Report Excerpt:

```
Top 5 Timed Events
~~~~~~~~~~~~~~~~~~                                    % Total DB Time
Event                              Waits   Time (s)   Avg Wait(ms)   % DB Time
------------------------------ ---------- ----------  -------------  --------
log file sync                    670,000     1,750         2.61         42.0
DB CPU                                         1,300                      31.0
log file parallel write          230,000       650         2.83         15.6
db file sequential read          700,000       490         0.70         11.8
```

Observations:

- "Log file sync" accounts for **42%** of total DB time, making it a significant bottleneck.
- High CPU utilization was noted, with DB CPU time contributing to **31%** of total DB time.
- The presence of elevated "log file parallel write" waits indicated that disk I/O might be partially involved, but the consistent high CPU usage suggested another layer of the problem.

Step 2: Investigating CPU Contention

The team then examined the CPU usage more closely to determine if the LGWR process was experiencing contention, which could delay the writing of redo entries to disk and thus increase "log file sync" waits.

Sample Output:

```
SID   SPID      PROGRAM             PGA_USED_MEM    STATUS  USERNAME
----  --------  ------------------  -------------   ------  --------
103   27532     ora_lgwr_ORCL       150M            ACTIVE  SYS
```

Observations:

- The LGWR process was actively using CPU resources.
- CPU contention was likely, as other critical processes also required substantial CPU resources during peak periods, which could delay LGWR's operations.

Step 3: Evaluating System Load and Process Prioritization

The DBA team further analyzed the overall system load and confirmed that the CPU was heavily utilized during billing cycles, with multiple high-priority processes competing for CPU time. This competition caused delays in LGWR's ability to write redo logs to disk promptly.

Resolution:

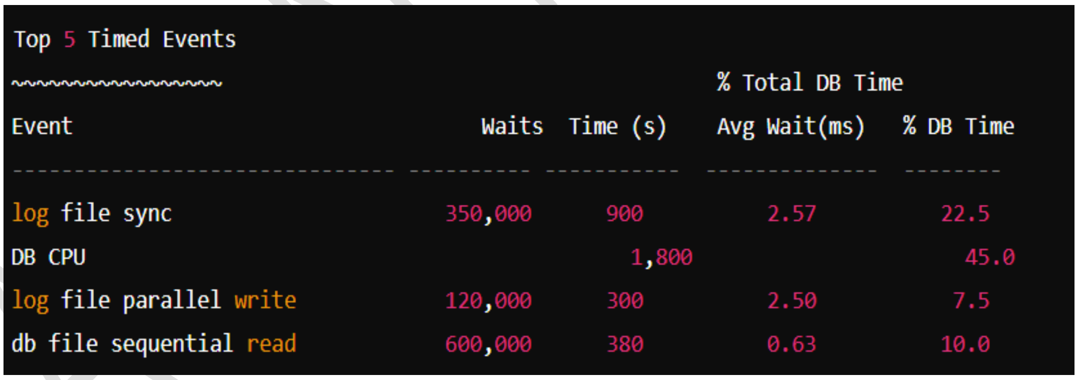
Step 4: Dedicating CPU Resources to the LGWR Process

To alleviate the CPU contention, the team decided to dedicate specific CPU resources to the LGWR process using CPU affinity tools. This ensures that LGWR has the necessary CPU cycles to perform its tasks without being delayed by other processes.

Steps to Dedicate CPU Resources:

1. **Identify the LGWR Process:** *ps -ef | grep lgwr*
2. **Use taskset to Bind the LGWR Process to Specific CPUs:** *taskset -cp 0,1 27532*
3. **Monitor the Impact:** Use top or similar monitoring tools to observe the CPU usage by the LGWR process and ensure that it is receiving adequate CPU time.

Post-Resolution AWR Report Excerpt:



```
Top 5 Timed Events
~~~~~~~~~~~~~~~~~~                                    % Total DB Time
Event                              Waits    Time (s)    Avg Wait(ms)   % DB Time
------------------------------ ---------- ----------- -------------- --------
log file sync                    350,000       900          2.57          22.5
DB CPU                                          1,800                      45.0
log file parallel write          120,000       300          2.50           7.5
db file sequential read          600,000       380          0.63          10.0
```

Outcome:

- **Reduction in "Log File Sync" Waits:** The percentage of DB time spent on "log file sync" was reduced from **42%** to **22.5%**.
- **Improved LGWR Performance:** The LGWR process was able to write redo logs more efficiently, reducing overall wait times.
- **Balanced System Load:** By dedicating CPUs to LGWR, other critical processes also experienced more stable performance, leading to improved system responsiveness during peak periods.

Limitations:

- **Potential for Resource Starvation:** Dedicating CPUs to LGWR may lead to resource starvation for other processes, especially in environments with limited CPU availability. This could shift performance issues to other areas.
- **Complex Configuration Management:** Managing CPU affinity and ensuring that the dedicated resources remain optimal as workloads change can add complexity to system administration.

Key Learnings:

- **CPU Contention and LGWR:** CPU contention can significantly impact the performance of the LGWR process, leading to increased "log file sync" waits. Prioritizing LGWR can mitigate these delays.
- **Process Prioritization:** Ensuring that critical processes like LGWR have the necessary CPU resources can lead to overall performance improvements, especially in high-transaction environments.
- **Ongoing Monitoring:** Regular monitoring and adjustments are essential to maintain the effectiveness of CPU dedication as system workloads evolve.

Scenario 4: Resolving "Log File Sync" Wait Event Due to Control File Contention

Background:

In a production Oracle 11.2 database, the DBA team noticed a sudden spike in "log file sync" wait events that could not be attributed to typical causes like I/O or CPU issues. The average wait time dramatically increased, causing noticeable delays in transaction processing.

Investigation:

Step 1: Analyzing the AWR Report

The DBA team observed that the "log file sync" wait events had increased unexpectedly, with a significant spike in average wait times.

AWR Report Excerpt:

```
Top 5 Timed Events
~~~~~~~~~~~~~~~~~~                                        % Total DB Time
Event                                Waits    Time (s)   Avg Wait(ms)   % DB Time
------------------------------ ---------- ----------- -------------- --------
log file sync                        14113       16515         1170.2       55.0
DB CPU                                              1100                    28.0
```

Observations:

- The high average wait time of **1170.2 ms** per "log file sync" event suggested an unusual bottleneck, far beyond typical I/O or CPU-related delays.

Step 2: Investigating Log Writer Activity

Using ASH (Active Session History), the DBA team traced the activity of the LGWR process and found that it was stuck on an "enq: CF – contention" wait event, which indicates a problem with writing to the control file.

SQL Query to Identify Blocking Sessions:

```sql
SELECT DISTINCT blocking_session, blocking_session_serial#
FROM dba_hist_active_sess_history
WHERE program LIKE '%LGWR%'
AND event = 'enq: CF - contention';
```

Findings:

- The LGWR process was blocked by another session, identified as an RMAN backup job, which was also trying to write to the control file.

Step 3: Assessing RMAN Activity

Further investigation revealed that during the period of high "log file sync" waits, RMAN was performing intensive control file writes, likely due to a high value of the *control_file_record_keep_time* parameter, which was set to **90 days**. This led to RMAN writing several gigabytes of data to the control file, causing severe contention.

Resolution:

Step 4: Tuning RMAN and Control File Configuration

To resolve the issue, the DBA team made the following changes:

1. **Reduce the Control File Record Retention:**
   - The *control_file_record_keep_time* parameter was adjusted to a more reasonable value, reducing the amount of data RMAN needed to write to the control file.
2. **Schedule RMAN Backups Appropriately:**
   - RMAN backups were rescheduled to avoid overlapping with peak transaction periods, reducing the likelihood of contention with LGWR.

Post-Resolution AWR Report Excerpt:

```
Top 5 Timed Events
~~~~~~~~~~~~~~~~~~                                         % Total DB Time
Event                                 Waits   Time (s)    Avg Wait(ms)   % DB Time
------------------------------ ---------- -----------  --------------  --------
log file sync                        3000        600          200.0          15.0
DB CPU                                              1600                    35.0
```

Outcome:

- **Reduction in "Log File Sync" Waits:** Average wait times were significantly reduced, with "log file sync" waits dropping from **1170.2 ms** to **200.0 ms**.
- **Improved System Performance:** The system experienced smoother operation with no significant delays during transaction processing.

Limitations:

- **Impact on Backup Windows:** Adjusting RMAN schedules might extend backup windows or require more careful planning to avoid overlap with other critical operations.
- **Balancing Control File Retention:** Reducing *control_file_record_keep_time* too much might lead to the loss of valuable historical data, which needs to be balanced against performance considerations.

Key Learnings:

- **Beyond I/O and CPU:** Not all "log file sync" waits are due to typical I/O or CPU issues; control file contention can be a significant factor.
- **Investigate Blocking Sessions:** Utilizing ASH or similar tools to identify blocking sessions is critical for diagnosing complex wait events.
- **Proactive Backup Management:** Ensure that RMAN or other backup processes are properly scheduled and configured to avoid performance bottlenecks during peak periods.

Scenario 5: Resolving "Log File Sync" Wait Event Due to Network Latency in a Distributed Database Setup

Background:

A global financial institution using Oracle RAC and Data Guard experienced "log file sync" waits due to network latency during peak transaction periods.

Investigation:

AWR reports showed "log file sync" and "remote file I/O" waits correlated with high network latency. The SYNC mode used in Data Guard exacerbated delays due to the long distances between primary and standby databases.

Resolution:

The team switched to ASYNC mode for distant standby sites, upgraded network links, and increased redo log buffer sizes to mitigate network delays.

Outcome:

"Log file sync" waits decreased from 48% to 21% of DB time, with improved transaction processing.

Limitations:

- High cost of network upgrades.
- Potential data lag in ASYNC mode.