**Cloud Support Engineer - Log Analysis Report**

**Information Extracted from the Log Files**

The log files (task1.txt, task2.txt, task3.txt) provide a detailed, chronological record of application events, revealing insights into the system's structure, component interactions, and operational health.

**System Overview**

* **Dblogger (Source: task1.txt):** Manages client connections and disconnections (Client connected, Client D disconnected). Also manages internal data sources (Starting new data source, Compacting data sources, archiving data source)
* **Connlogger (Source: task2.txt):** Responsible for connection and resource management, particularly in handling concurrency. It allocates and deallocates threads (Using threads) and manages remote connections (Waiting on remote, Finished remote connections, Clearing connections pool).
* **StoreLogger (Source: task3.txt):** Data persistence and processing layer. It handles significant volumes of data reads (Reading N MBi) and writes (Writing N kB). It also performs storage-related maintenance (TRIMing SSD) and manages connections (Dealing with connection, Client connected, Client disconnected)

**Issues Identified**

Based on the warning level logs and unusual patterns, here are the identified issues:

1. **Warning Data Source (DbLogger):**

* **Issue:** “Compacting data sources" and “Archiving data source" are the most frequent warnings in task1.txt
* **Impact:** These operations are resource-intensive (CPU, memory, and especially disk I/O). Their constant execution can significantly impact overall performance and availability.

1. **Waiting for disk write (DbLogger):**

* **Issue:** “Still waiting for disk write to be committed” This warning indicates that the Dblogger is experiencing severe delays when writing data to disk.
* **Impact:** Write operations are queued, degrading the entire system and increasing the risk of data loss.

1. **Connection to host failed (DbLogger):**

* **Issue: “**Connection to hostA.local failed: host unreachable” and “Connection to hostB.local failed: Could not resolve host”
* **Impact:** These failures indicate potential cluster instability, replication failures or an inability to maintain high availability.

1. **Duplicate key violation for primary key (DbLogger):**

* **Issue: “**Failed to write record, duplicate key violation for primary key”
* **Impact:** Potential error in application data handling, this issue indicates data corruption or skipped writes.

1. **Connections pool (Connlogger):**

* **Issue: “**Clearing connections pool” Most frequent warning in task2
* **Impact:** Frequent deletion suggests instability or resource exhaustion in connection management, creating possible service interruptions and reduced performance.

1. **DNS timeout (Connlogger):**

* **Issue: “**DNS timeout” Failed to resolve a domain name to an IP address
* **Impact:** This directly affects the ability to communicate with other services or external resources, resulting in service interruptions or functional issues.
* **Correlation:** This is a possible cause of the "Could not resolve host" warning in task 1.

1. **Dealing with remote (Connlogger):**

* **Issue: “**Dealing with remote” Frequent warning
* **Impact:** It could be a symptom of network latency or the remote system responding slowly.

1. **Bandwidth usage (Connlogger):**

* **Issue: “**Using XMb of bandwidth” up to 996Mb
* **Impact:** This high usage, with DNS timeouts and connection pool flushing as a significant factor contributing to system instability. Network saturation can cause packet loss, increased latency, and ultimately connection failures.

1. **Frequent Client Disconnections (Storelogger):**

* **Issue: “**Client disconnected” This is the most frequent warning in task3 and the second most frequent across all logs.
* **Impact:** It can cause data integrity issues as writes or reads may be incomplete if connections are interrupted during the operation.

1. **Disk queue (StoreLogger):**

* **Evidence: “**Long disk queue on /dev/sd[a-z]” 300+ occurrences across many devices
* **Impact:** This causes massive performance degradation, operations on these disks will be extremely slow and can cause applications to time out, retry excessively or crash.
* **Correlation:** This directly explains the "Still waiting for disk write to be confirmed" error in Task 1 and likely contributes to the client disconnected warnings in both task 1 and task 3.

1. **TRIMing SSD (StoreLogger):**

* **Evidence: “**TRIMing SSD” Very high frequency.
* **Impact:** Constantly running TRIM indicates an extremely high rate of data deletion or modification. This increases disk I/O load and can contribute to long disk queue issues if SSDs become overloaded, impacting their performance.

**Information for On-Call Engineers**

An on-call engineer would need a blend of real-time data, historical context, and system configuration details to effectively debug these issues.

* **System Metrics (Real-time and Historical):** Disk I/O Metrics, Network Metrics, CPU and Memory Metrics.
* **System Configuration Details:** Hardware Specifications, Operating System Configuration, Network Topology, Database Configuration, Application Configuration and Storage Configuration.
* **Recent Changes/Deployment History:** Any recent code deployments, configuration changes, infrastructure changes (hardware, network, OS patches), or increases in traffic/load. This is crucial for identifying potential regression or new bottlenecks.
* **Baseline Performance Data:** What constitutes "normal" behavior for bandwidth, disk queues, connection pool activity, etc.

**Support on-call engineers to produce a fix**

* **Prioritized Action Plan:** Provide a clear, prioritized list of issues based on severity and impact (e.g., Disk I/O first, then network, then client disconnects).
* **Relevant Documentation/Runbooks:** Point to existing runbooks or internal documentation for common troubleshooting steps related to database performance, network issues, or Go application diagnostics.
* **Escalation Paths:** Clearly define when to escalate to specific teams (e.g., database administrators, network engineers for DNS/bandwidth, storage engineers for disk problems).
* **Communication Hub:** Establish a dedicated incident communication channel, where all findings, actions, and observations are recorded.
* **Knowledge Sharing:** After resolution, help document the fix, root cause, and preventative measures in a post-mortem or knowledge base article.

**Additional Information**

* **Application Architecture Diagram:** A high-level diagram showing how Dblogger, Connlogger, and StoreLogger interact with each other and external dependencies (clients, DNS, other hosts) would clarify communication paths.
* **Expected Workload Profile:** Understanding the typical and peak load for this system (e.g., number of concurrent clients, data ingestion rate, query patterns) would help determine if the current issues are due to misconfiguration, resource exhaustion, or unexpected load spikes.
* **Severity Level of "Using XMb of Bandwidth":** Confirm if these messages are explicitly logged as WARNING by the application or if they were included because they seemed like abnormal patterns.

**Recommended Improvements**

* **Centralized Logging and Monitoring System:** To improve system monitoring and detect problems early, you can implement a centralized logging system and collect comprehensive metrics with monitoring agents. Create unified dashboards to gain a single view of system status and proactive alerts at key thresholds to notify engineers before problems escalate.
* **Runbooks and Automated Remediation:** To improve incident handling, detailed workbooks should be created for common issues and self-healing or basic diagnostic tasks, such as restarting services or collecting data when alerts are triggered, should be automated.

**Assumptions Made**

* **Log Level Interpretation:** I assumed that all entries provided are indeed classified as "warnings" by the logging configuration of the respective applications.
* **Default Network/Disk Setup:** I assumed a typical server environment with standard network interfaces and disk performance characteristics unless otherwise specified.
* **Logical Grouping of Logs:** I assumed that task1.txt corresponds to Dblogger, task2.txt to Connlogger, and task3.txt to StoreLogger based on the previously provided context and the nature of the messages.
* **TRIMing SSD is a warning for frequent execution:** TRIM is a healthy operation, it’s very high frequency (3559 times) suggests a potential performance impact due to constant garbage collection, warranting a warning.
* **The system components (Dblogger, Connlogger, StoreLogger) are running on the same server or closely interacting servers:** This allows for the correlation of disk I/O, network, and application issues.
* **Production Load:** System experiencing higher than normal operational load

**Analysis Methodology**

**Initial Overview and Component Identification:**

* **Read through log samples:** Began by reading the provided snippets for each task\*.txt to understand the general types of messages and identify the main components (Dblogger, Connlogger, StoreLogger) and their apparent roles.
* **Identify Message Patterns:** Looked for recurring phrases or keywords that hinted at specific functionalities (e.g., "connections," "disk write," "bandwidth").

**Warning Extraction:**

* **Focus on "Warning" Level:** Specifically requested and then analyzed only the warning level messages, as these are indicators of potential problems that the application itself deems significant.
* **Quantitative Analysis:** Used the provided counts for each warning message to identify the most frequent issues, giving an immediate sense of their prevalence and potential impact.

**Using Command Line Tools:** To efficiently filter and process the logs, I used VS Code with a WSL terminal and the following command:

*Bash*

*grep -i "level=warning" task\*.txt | awk -F'msg="' '{print $2}' | awk -F'"' '{print $1}' | sort | uniq -c >> warnings.txt*

* **grep -i "level=warning" task1.txt:** Filter lines containing "level=warning" (ignoring case).
* **awk -F'msg="' '{print $2}':** Extract the part of the message following msg=".
* **awk -F'"' '{print $1}':** Get only the message content before the next quote.
* **sort | uniq -c:** Count the unique occurrences of each warning message.
* **>> warnings.txt:** Redirect output to a file for further analysis.

**Detailed Message Interpretation and Impact Assessment:**

* **Keyword Analysis:** For each unique warning message, I analyzed keywords and phrases to infer its direct meaning (e.g., "host unreachable" means a network problem, "Long disk queue" means I/O latency).
* **System Impact:** For each warning, I considered its potential consequences on system performance, stability, data integrity, and user experience. I drew on general knowledge of system administration, networking, and database operations to assess impact.
* **Contextual Understanding:** Related messages to the likely function of the component logging them. For example, "DNS timeout" in Connlogger it as a core network issue.

**Cross-Log Correlation:**

* **I looked for connections between warnings across different logs. For example:** Long disk queue (StoreLogger) directly correlates with Still waiting for disk write to be committed (Dblogger). DNS timeout (Connlogger) correlates with Could not resolve host (Dblogger). High Bandwidth usage (Connlogger) combined with DNS timeout and Clearing connections pool strongly points to network congestion. Severe underlying infrastructure issues (disk I/O, network) are likely root causes for client-facing issues like Client disconnected.

**Recommendations:**

* **Troubleshooting Steps:** Proposed concrete steps for on-call engineers to verify the issues and gather more data.
* **Remediation Strategies:** Suggested potential fixes, categorized by resource type (e.g., disk, network, application).
* **Long-Term Improvements:** Identified systemic changes (monitoring, logging, architecture) that would prevent recurrence and improve future supportability.