🔥 Common Keywords to Search for Java Errors/Exceptions in GC Logs

When analyzing **Java GC logs** for potential issues, certain keywords can indicate **memory pressure**, **promotion failures**, or **runtime errors**. Searching for these terms helps quickly pinpoint **critical GC events** or **JVM errors**.

---

1. **"OutOfMemoryError"**
    - Indicates the JVM ran out of heap or metaspace.
    - Example log snippet:

java.lang.OutOfMemoryError: Java heap space

2. **"GC overhead limit exceeded"**
    - Means the JVM is spending too much time doing GC and **not reclaiming enough memory**.
    - Often leads to **OutOfMemoryError** if the situation persists.
3. **"Promotion failed"** / **"to-space exhausted"**
    - Occurs when surviving objects **cannot be promoted** from the young generation to the old generation.
    - Can trigger **Full GC** or **concurrent mode failures**.
4. **"Allocation Failure"**
    - Typically seen in G1 GC logs, indicating the **heap is too fragmented** or **full**, prompting a GC cycle.
    - May lead to **Full GC** or **Evacuation Failure**.
5. **"Evacuation Failure"** (G1 GC)
    - Happens when the GC can't move objects to **Survivor or Old regions** due to insufficient space or fragmentation.
6. **"Concurrent mode failure"** (CMS GC)
    - Shows the **Concurrent Mark Sweep** collector **failed** to clean up memory in time, forcing a **stop-the-world Full GC**.
7. **"CMS: concurrent promotion failed"** (CMS GC)
    - Specifically indicates **promotion** of young generation objects **failed** under CMS, leading to a **Full GC**.

Santhosh Kumar J

8. **"Full GC"**
    - A **stop-the-world** collection of **both young and old generations**.
    - Frequent Full GCs → **severe performance bottleneck** or memory leak.
9. **"GCLocker Initiated GC"**
    - Indicates a **GC forced** by the **JVM safepoint** mechanism (e.g., **JNI critical regions**).
10. **"Unloading Class"** / **"Class Unloading"**
    - May appear if **class metadata** is running out of space, or in **Metaspace** scenarios.
    - Look for **OutOfMemoryError: Metaspace** references.

📌 **Sample Grep Commands for GC Logs**

Search for key GC errors or exceptions in GC logs

grep -E "OutOfMemoryError|GC overhead limit exceeded|Promotion failed|to-space exhausted|Allocation Failure|Evacuation Failure|Concurrent mode failure|Full GC|CMS: concurrent promotion failed|GCLocker Initiated GC" gc.log

---

🚀 **Pro Tips**

- **Combine keywords** with **time-based searches** (e.g., grep -A 10 -B 10 to see context).
- **Parse logs** with dedicated tools (e.g., ELK, Splunk) for better **visualization**.
- **Correlate** GC events with **application logs** to see **impact on performance**.

🔥 Advanced Grep Commands for Java GC Log Analysis (Time-Based & Contextual Searches)

When analyzing **GC logs** and **JVM errors**, it's critical to **search for specific keywords** while also retrieving surrounding context (**before & after relevant lines**).

📌 **1. Basic Grep for Critical JVM Errors**

grep -E "OutOfMemoryError|GC overhead limit exceeded|Promotion failed|to-space exhausted|Allocation Failure|Evacuation Failure|Concurrent mode failure|Full GC|CMS: concurrent promotion failed|GCLocker Initiated GC" gc.log

✅ Searches for **major GC-related failures** in gc.log.

Santhosh Kumar J

📌 **2. Grep with Context (View Surrounding Logs)**

✅ **Show 10 lines before (-B 10) and after (-A 10) a match**:

grep -A 10 -B 10 -E "OutOfMemoryError|GC overhead limit exceeded|Promotion failed|to-space exhausted|Allocation Failure|Evacuation Failure|Concurrent mode failure|Full GC|CMS: concurrent promotion failed|GCLocker Initiated GC" gc.log

✅ Helps **see events leading up to the failure** and **what happened after**.

📌 **3. Search for Errors Along with Timestamp**

✅ **Find GC failures with timestamps to correlate with application logs**:

grep -A 10 -B 10 -E "OutOfMemoryError|GC overhead limit exceeded|Promotion failed|to-space exhausted|Allocation Failure|Evacuation Failure|Concurrent mode failure|Full GC" gc.log | grep -E "^\[|ERROR|WARN"

✅ This ensures **timestamps ([TimeStamp]) and error lines** (ERROR, WARN) are captured together.

📌 **4. Filter for Specific Time Range**

✅ **Extract logs only between 08:00 and 09:30**:

awk '$1 >= "[2025-02-11T08:00:00" && $1 <= "[2025-02-11T09:30:00"' gc.log | grep -E "OutOfMemoryError|GC overhead limit exceeded|Full GC"

✅ Useful for **analyzing failures within a specific test execution window**.

📌 **5. Extract Top 5 Most Frequent Errors in GC Logs**

✅ **Find the most common JVM errors using awk and sort**:

grep -E "OutOfMemoryError|GC overhead limit exceeded|Full GC|CMS: concurrent promotion failed" gc.log | awk '{print $NF}' | sort | uniq -c | sort -nr | head -5

✅ Helps **prioritize which errors occur most frequently**.

Santhosh Kumar J

📌 6. Monitor GC Log Errors in Real-Time

✅ **Live tail logs and highlight critical errors (--color=auto):**

tail -f gc.log | grep --color=auto -E "OutOfMemoryError|Full GC|Allocation Failure|Evacuation Failure"

✅ Useful for **real-time monitoring** during performance testing.

🔥 Summary: Key Grep & Time-Based Search Commands

Command	Purpose
`grep -E "OutOfMemoryError	Full GC" gc.log`
`grep -A 10 -B 10 -E "OutOfMemoryError	Full GC" gc.log`
`awk '$1 >= "[Time1]" && $1 <= "[Time2]"' gc.log	grep "Full GC"`
`grep -E "Full GC	OOM" gc.log
`tail -f gc.log	grep --color=auto -E "OutOfMemoryError

Santhosh Kumar J