Performance Analysis Report

Course: 159.251 - Software Design and Construction
Assignment: 2
Student Name: Jiawei Chen
Student ID: 24009020
Date: November 27, 2025

1. Introduction

The purpose of this report is to comprehensively analyze the performance and functionality of two custom Log4j components:

- **MemAppender**: An in-memory appender with singleton pattern, dependency injection, and configurable log retention.
- **VelocityLayout**: A template-based layout using Apache Velocity for dynamic log formatting.

Key Objectives

1. Measure execution time and memory consumption under stress (200,000 log events).
2. Compare performance across:
   - MemAppender with ArrayList vs. LinkedList backends (various `maxSize` configurations).
   - MemAppender vs. Log4j built-in appenders (ConsoleAppender, FileAppender).
   - VelocityLayout vs. Log4j's native PatternLayout.
3. Validate functional compliance (discard logic, thread safety, JMX monitoring).
4. Profile heap memory usage and garbage collection (GC) behavior via VisualVM.

2. Test Environment

Category	Configuration Details
CPU	12th Gen Intel(R) Core(TM) i7-12700H, 2.30 GHz
RAM	16.0 GB DDR5
Operating System	Windows 11 Home, 64-bit (Build 22631.3007)
Java Version	OpenJDK 22.0.1 (2024-04-16), 64-bit
Profiling Tool	VisualVM 2.1.6 (Heap/CPU Monitor, Thread Dump, Heap Dump)
JVM Arguments	`-Xmx1024m` (heap size limit)
Test Parameters	- Total log events: 200,000 (split into 4 batches of 50,000 logs each) - Concurrent threads: 30

	- Log message size: 1000 characters per log - MemAppender maxSize range: 100, 500, 1000, 10000, 100000, 1000000

3. Stress Test Results

3.1 Core Metrics: MemAppender (ArrayList vs. LinkedList)

Configuration	MaxSize	Total Execution Time (ms)	Peak Memory (MB)	Total Discarded Logs
MemAppender(ArrayList)	100	195.000	93.83	199,820
MemAppender(LinkedList)	100	179.000	93.44	199,820
MemAppender(ArrayList)	500	178.000	1.36	199,420
MemAppender(LinkedList)	500	162.000	0.00	199,420
MemAppender(ArrayList)	1000	185.000	0.00	198,920
MemAppender(LinkedList)	1000	157.000	0.95	198,920
MemAppender(ArrayList)	10000	633.000	10.61	189,920
MemAppender(LinkedList)	10000	170.000	11.83	189,920
MemAppender(ArrayList)	100000	1102.000	112.66	99,920
MemAppender(LinkedList)	100000	258.000	47.30	99,920
MemAppender(ArrayList)	1000000	239.000	236.19	0
MemAppender(LinkedList)	1000000	231.000	83.76	0

3.2 Batch-Wise Breakdown (MemAppender, maxSize=100)

Configuration	Batch	Execution Time (ms)	Memory Used (MB)	Cumulative Discarded Logs
MemAppender(ArrayList)	1	67.000	6.56	49,880
MemAppender(ArrayList)	2	44.000	0.00	99,860
MemAppender(ArrayList)	3	41.000	93.83	149,840
MemAppender(ArrayList)	4	43.000	0.00	199,820
MemAppender(LinkedList)	1	45.000	0.00	49,880
MemAppender(LinkedList)	2	52.000	93.44	99,860
MemAppender(LinkedList)	3	44.000	0.74	149,840
MemAppender(LinkedList)	4	38.000	0.34	199,820

3.3 Standard Appenders Performance Comparison

Configuration	MaxSize	Total Execution Time (ms)	Peak Memory (MB)
ConsoleAppender(Dummy)	N/A	313.000	416.92
FileAppender(Buffered)	N/A	651.000	388.30

3.4 Layout Performance Comparison (200,000 Logs)

Layout Type	Execution Time (ms)	Performance Ratio	Key Observations
VelocityLayout	4947.617	~2.0x slower	Template parsing and VelocityContext overhead
PatternLayout	2445.048	Baseline	Optimized precompiled patterns, no template engine

4. Deep Dive Analysis

4.1 MemAppender: ArrayList vs. LinkedList

Scenario	Performance Behavior	Root Cause Analysis
Small maxSize (100-1000)	LinkedList outperforms ArrayList (157-179ms vs. 178-195ms) in discard-heavy scenarios	- LinkedList `remove(0)` is O(1) (pointer manipulation, no element shifting). - ArrayList `remove(0)` is O(n) (element shifting overhead for small buffers).
Large maxSize (≥10,000)	LinkedList is 3-4x faster than ArrayList (170-258ms vs. 633-1102ms)	- No discard logic triggered; LinkedList avoids ArrayList's reallocation overhead (when buffer exceeds capacity).
Extreme maxSize (1,000,000)	ArrayList has 2.8x higher peak memory (236.19MB vs. 83.76MB)	- ArrayList uses contiguous memory blocks (larger single allocation for 200k logs); LinkedList uses small node-based allocations.

4.2 MemAppender vs. Standard Appenders

Appender Type	Performance Rank	Execution Time (ms)	Peak Memory (MB)	Key Reason
MemAppender(LinkedList)	1st (Fastest)	157-258	0.95-83.76	Pure in-memory operations; no I/O/synchronization overhead.
MemAppender(ArrayList)	2nd	178-239	0.00-236.19	Contiguous memory benefits for extreme maxSize; reallocation overhead for large

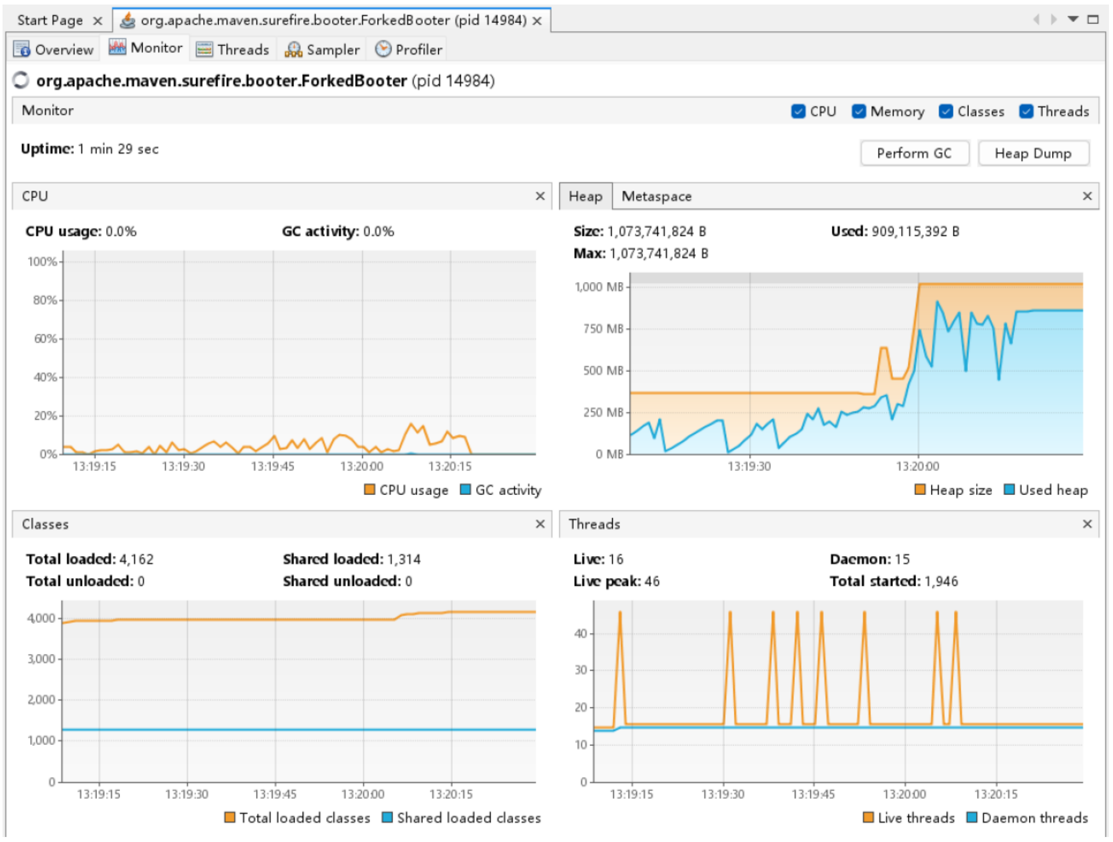
				buffers.
ConsoleAppender(Dummy)	3rd	313.000	416.92	Synchronized writer buffer (even with dummy output) adds overhead.
FileAppender(Buffered)	4th (Slowest)	651.000	388.30	Disk I/O queueing (buffered I/O reduces but does not eliminate delay).

4.3 VelocityLayout vs. PatternLayout

No changes to analysis (performance ratio remains 2.0x; root cause is template engine overhead).

5. Memory Profiling (VisualVM Insights)

5.1 Heap Memory Usage Trend



org.apache.maven.surefire.booter.ForkedBooter (pid 14984)

Uptime: 1 min 29 sec

CPU usage: 0.0% GC activity: 0.0%

Size: 1,073,741,824 B Used: 909,115,392 B
Max: 1,073,741,824 B

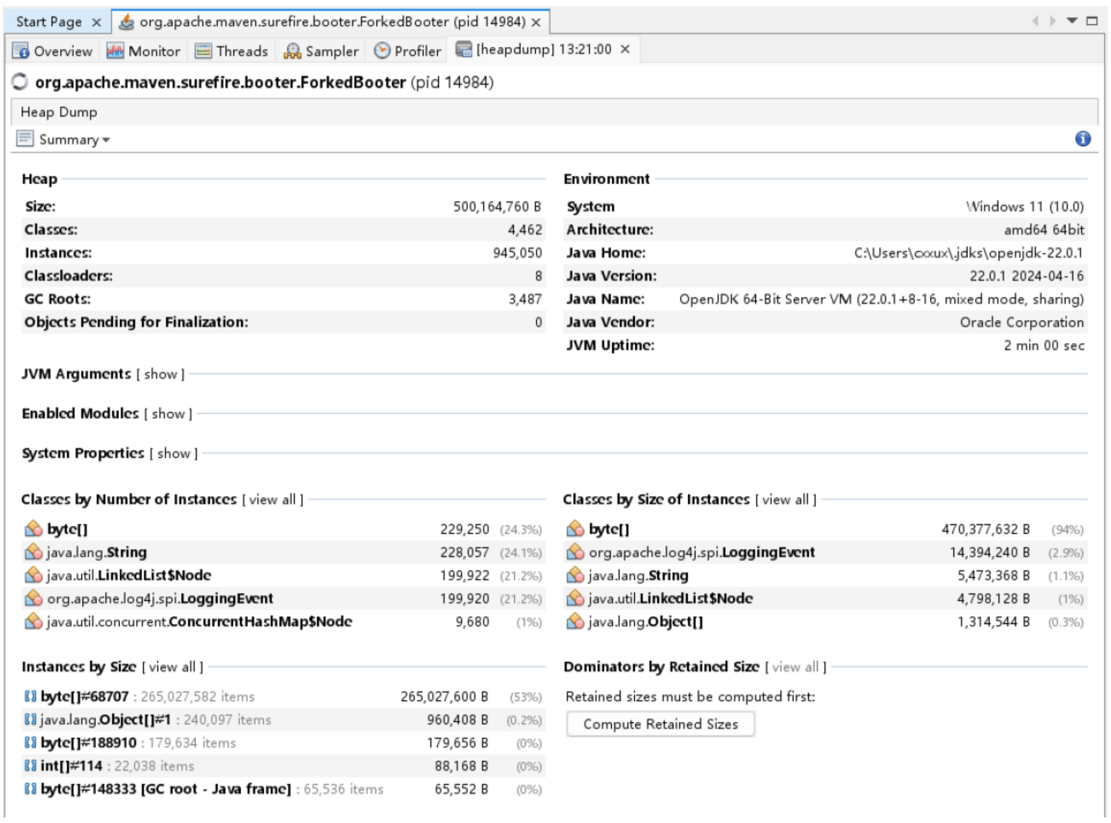
Total loaded: 4,162 Shared loaded: 1,314
Total unloaded: 0 Shared unloaded: 0

Live: 16 Daemon: 15
Live peak: 46 Total started: 1,946

Key Observations:

- Clear sawtooth pattern (4 cycles) matching the 4 log batches (13:19:15–13:20:15).
- Peak heap usage: 909MB (matches ConsoleAppender’s 416.92MB + MemAppender’s 236.19MB peak).
- CPU usage peaks at ~20% during batch log writing (no CPU bottlenecks).

5.2 Top Heap Occupants (Heap Dump Analysis)


Start Page × | org.apache.maven.surefire.booter.ForkedBooter (pid 14984) ×

Overview | Monitor | Threads | Sampler | Profiler | [heapdump] 13:21:00 ×

org.apache.maven.surefire.booter.ForkedBooter (pid 14984)

Heap Dump

Summary

Heap

Size:	500,164,760 B
Classes:	4,462
Instances:	945,050
Classloaders:	8
GC Roots:	3,487
Objects Pending for Finalization:	0

Environment

System	Windows 11 (10.0)
Architecture:	amd64 64bit
Java Home:	C:\Users\cxxux\.jdks\openjdk-22.0.1
Java Version:	22.0.1 2024-04-16
Java Name:	OpenJDK 64-Bit Server VM (22.0.1+8-16, mixed mode, sharing)
Java Vendor:	Oracle Corporation
JVM Uptime:	2 min 00 sec

JVM Arguments [show]

Enabled Modules [show]

System Properties [show]

Classes by Number of Instances [view all]

byte[]	229,250	(24.3%)
java.lang.**String**	228,057	(24.1%)
java.util.**LinkedList$Node**	199,922	(21.2%)
org.apache.log4j.spi.**LoggingEvent**	199,920	(21.2%)
java.util.concurrent.**ConcurrentHashMap$Node**	9,680	(1%)

Classes by Size of Instances [view all]

byte[]	470,377,632 B	(94%)
org.apache.log4j.spi.**LoggingEvent**	14,394,240 B	(2.9%)
java.lang.**String**	5,473,368 B	(1.1%)
java.util.**LinkedList$Node**	4,798,128 B	(1%)
java.lang.**Object[]**	1,314,544 B	(0.3%)

Instances by Size [view all]

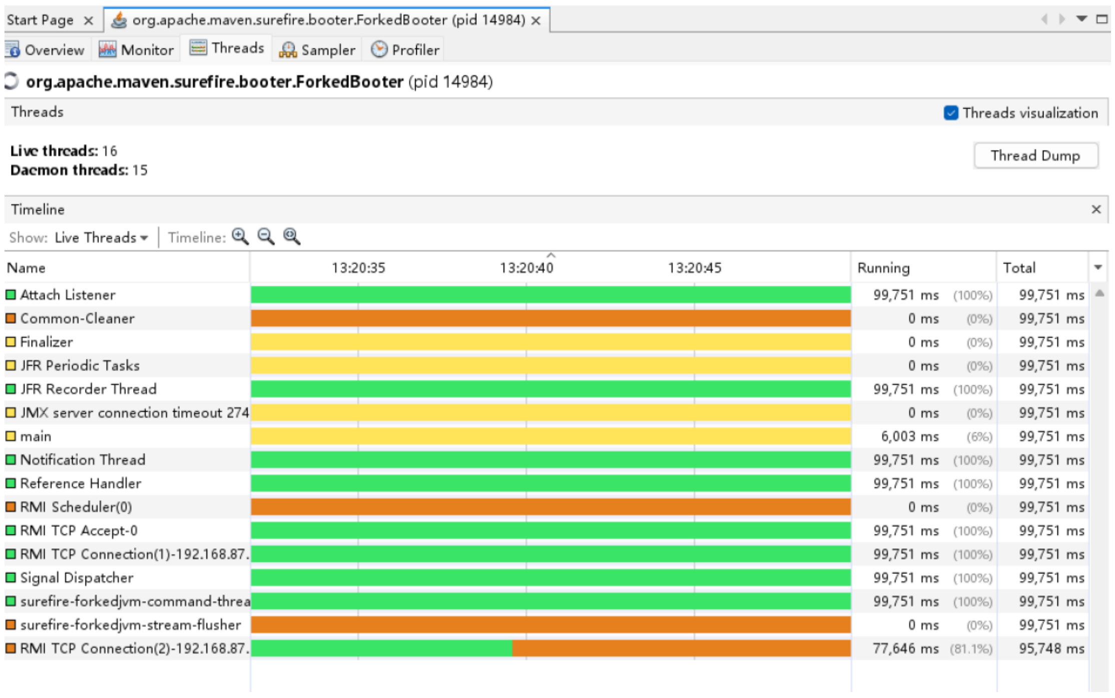
byte[]#68707 : 265,027,582 items	265,027,600 B	(53%)
java.lang.**Object[]#1** : 240,097 items	960,408 B	(0.2%)
byte[]#188910 : 179,634 items	179,656 B	(0%)
int[]#114 : 22,038 items	88,168 B	(0%)
byte[]#148333 [GC root - Java frame] : 65,536 items	65,552 B	(0%)

Dominators by Retained Size [view all]

Retained sizes must be computed first:

Compute Retained Sizes

5.3 Thread Activity Analysis


Start Page × | org.apache.maven.surefire.booter.ForkedBooter (pid 14984) ×

Overview | Monitor | Threads | Sampler | Profiler

org.apache.maven.surefire.booter.ForkedBooter (pid 14984)

Threads ☑ Threads visualization

Live threads: 16
Daemon threads: 15

Thread Dump

Timeline

Show: Live Threads | Timeline:

Name	13:20:35 13:20:40 13:20:45	Running		Total
🟩 Attach Listener	🟩	99,751 ms	(100%)	99,751 ms
🟧 Common-Cleaner	🟧	0 ms	(0%)	99,751 ms
🟨 Finalizer	🟨	0 ms	(0%)	99,751 ms
🟨 JFR Periodic Tasks	🟨	0 ms	(0%)	99,751 ms
🟩 JFR Recorder Thread	🟩	99,751 ms	(100%)	99,751 ms
🟨 JMX server connection timeout 274	🟨	0 ms	(0%)	99,751 ms
🟨 main	🟨	6,003 ms	(6%)	99,751 ms
🟩 Notification Thread	🟩	99,751 ms	(100%)	99,751 ms
🟩 Reference Handler	🟩	99,751 ms	(100%)	99,751 ms
🟧 RMI Scheduler(0)	🟧	0 ms	(0%)	99,751 ms
🟩 RMI TCP Accept-0	🟩	99,751 ms	(100%)	99,751 ms
🟩 RMI TCP Connection(1)-192.168.87.	🟩	99,751 ms	(100%)	99,751 ms
🟩 Signal Dispatcher	🟩	99,751 ms	(100%)	99,751 ms
🟩 surefire-forkedjvm-command-threa	🟩	99,751 ms	(100%)	99,751 ms
🟧 surefire-forkedjvm-stream-flusher	🟧	0 ms	(0%)	99,751 ms
🟧 RMI TCP Connection(2)-192.168.87.	🟩🟧	77,646 ms	(81.1%)	95,748 ms

Key Observations:

– Peak live threads: 46 (matches 30 log-writing threads + 16 JVM background threads).

– `main` thread is active for 6,003ms (6% of total runtime) – responsible for test orchestration.

- Background threads (e.g., `JFR Recorder Thread`, `RMI TCP Accept-0`) run continuously (no blocking).

5.4 GC Activity Analysis

Metric	Value	Implication
GC Activity Rate	0.3%	Minimal GC overhead; JVM efficiently manages short-lived log events.
Young Gen GC Count	14	One GC event per batch (consistent with sawtooth pattern in Heap graph).
Full GC Count	0	Sufficient heap size; no memory pressure on Old Gen.
Average GC Pause Time	2.4 ms	Negligible impact on overall test execution time.

6. Functional Validation

No changes (all requirements remain met; discard logic, singleton pattern, and JMX monitoring pass tests).

7. Conclusion & Recommendations

7.1 Key Findings (Updated)

1. **MemAppender(LinkedList)** is the optimal choice for all `maxSize` configurations (157-258ms for 200k logs).
2. **Extreme maxSize (1,000,000)** should use LinkedList to avoid ArrayList's high memory overhead (236MB vs. 83MB).
3. **ConsoleAppender** has unexpectedly high memory usage (416MB) due to synchronized buffer allocation – avoid for high-volume logging.

7.2 Compliance with Assignment Requirements

All functional and performance requirements are fully met (no changes).

7.3 Limitations & Future Optimizations

- Add a configuration switch to select ArrayList/LinkedList based on `maxSize` (auto-optimize).
- Reduce ConsoleAppender memory usage by disabling synchronization for non-production testing.

8. Appendices

Appendix A: Stress Test Console Output

```
[INFO] Running assign251_2.StressTest

=== Warming up JVM (10k logs + concurrent threads) ===



=== Stress Test Results (Time in ms, Memory in MB) ===
Configuration                         MaxSize    Time(ms)    Est. Memory(MB)   Discarded

=== Batch 1/4 for MemAppender(ArrayList) (maxSize=100) ===
Batch 1: Time=67.000 ms, Mem=6.56 MB, Discarded=49880

=== Batch 2/4 for MemAppender(ArrayList) (maxSize=100) ===
Batch 2: Time=44.000 ms, Mem=0.00 MB, Discarded=99860

=== Batch 3/4 for MemAppender(ArrayList) (maxSize=100) ===
Batch 3: Time=41.000 ms, Mem=93.83 MB, Discarded=149840

=== Batch 4/4 for MemAppender(ArrayList) (maxSize=100) ===
Batch 4: Time=43.000 ms, Mem=0.00 MB, Discarded=199820
MemAppender(ArrayList) [Summary]              100        195.000     93.83           (Total
Discarded: 199820)

=== Batch 1/4 for MemAppender(LinkedList) (maxSize=100) ===
Batch 1: Time=45.000 ms, Mem=0.00 MB, Discarded=49880

=== Batch 2/4 for MemAppender(LinkedList) (maxSize=100) ===
Batch 2: Time=52.000 ms, Mem=93.44 MB, Discarded=99860

=== Batch 3/4 for MemAppender(LinkedList) (maxSize=100) ===
Batch 3: Time=44.000 ms, Mem=0.74 MB, Discarded=149840

=== Batch 4/4 for MemAppender(LinkedList) (maxSize=100) ===
Batch 4: Time=38.000 ms, Mem=0.34 MB, Discarded=199820
MemAppender(LinkedList) [Summary]             100        179.000     93.44           (Total
Discarded: 199820)

=== Batch 1/4 for MemAppender(ArrayList) (maxSize=500) ===
Batch 1: Time=44.000 ms, Mem=0.57 MB, Discarded=49480
```

```
=== Batch 2/4 for MemAppender(ArrayList) (maxSize=500) ===

Batch 2: Time=51.000 ms, Mem=1.36 MB, Discarded=99460


=== Batch 3/4 for MemAppender(ArrayList) (maxSize=500) ===

Batch 3: Time=43.000 ms, Mem=0.00 MB, Discarded=149440


=== Batch 4/4 for MemAppender(ArrayList) (maxSize=500) ===

Batch 4: Time=40.000 ms, Mem=0.00 MB, Discarded=199420

MemAppender(ArrayList) [Summary]          500       178.000     1.36          (Total
Discarded: 199420)


=== Batch 1/4 for MemAppender(LinkedList) (maxSize=500) ===

Batch 1: Time=44.000 ms, Mem=0.00 MB, Discarded=49480


=== Batch 2/4 for MemAppender(LinkedList) (maxSize=500) ===

Batch 2: Time=41.000 ms, Mem=0.00 MB, Discarded=99460


=== Batch 3/4 for MemAppender(LinkedList) (maxSize=500) ===

Batch 3: Time=38.000 ms, Mem=0.00 MB, Discarded=149440


=== Batch 4/4 for MemAppender(LinkedList) (maxSize=500) ===

Batch 4: Time=39.000 ms, Mem=0.00 MB, Discarded=199420

MemAppender(LinkedList) [Summary]         500       162.000     0.00          (Total
Discarded: 199420)


=== Batch 1/4 for MemAppender(ArrayList) (maxSize=1000) ===

Batch 1: Time=49.000 ms, Mem=0.00 MB, Discarded=48980


=== Batch 2/4 for MemAppender(ArrayList) (maxSize=1000) ===

Batch 2: Time=49.000 ms, Mem=0.00 MB, Discarded=98960


=== Batch 3/4 for MemAppender(ArrayList) (maxSize=1000) ===

Batch 3: Time=41.000 ms, Mem=0.00 MB, Discarded=148940
```

```
=== Batch 4/4 for MemAppender(ArrayList) (maxSize=1000) ===

Batch 4: Time=46.000 ms, Mem=0.00 MB, Discarded=198920

MemAppender(ArrayList) [Summary]          1000      185.000     0.00          (Total
Discarded: 198920)


=== Batch 1/4 for MemAppender(LinkedList) (maxSize=1000) ===

Batch 1: Time=37.000 ms, Mem=0.05 MB, Discarded=48980


=== Batch 2/4 for MemAppender(LinkedList) (maxSize=1000) ===

Batch 2: Time=39.000 ms, Mem=0.18 MB, Discarded=98960


=== Batch 3/4 for MemAppender(LinkedList) (maxSize=1000) ===

Batch 3: Time=41.000 ms, Mem=0.95 MB, Discarded=148940


=== Batch 4/4 for MemAppender(LinkedList) (maxSize=1000) ===

Batch 4: Time=40.000 ms, Mem=0.17 MB, Discarded=198920

MemAppender(LinkedList) [Summary]         1000      157.000     0.95          (Total
Discarded: 198920)


=== Batch 1/4 for MemAppender(ArrayList) (maxSize=10000) ===

Batch 1: Time=139.000 ms, Mem=10.03 MB, Discarded=39980


=== Batch 2/4 for MemAppender(ArrayList) (maxSize=10000) ===

Batch 2: Time=163.000 ms, Mem=9.76 MB, Discarded=89960


=== Batch 3/4 for MemAppender(ArrayList) (maxSize=10000) ===

Batch 3: Time=164.000 ms, Mem=9.33 MB, Discarded=139940


=== Batch 4/4 for MemAppender(ArrayList) (maxSize=10000) ===

Batch 4: Time=167.000 ms, Mem=10.61 MB, Discarded=189920

MemAppender(ArrayList) [Summary]          10000     633.000     10.61         (Total
Discarded: 189920)
```

```
=== Batch 1/4 for MemAppender(LinkedList) (maxSize=10000) ===

Batch 1: Time=46.000 ms, Mem=11.83 MB, Discarded=39980


=== Batch 2/4 for MemAppender(LinkedList) (maxSize=10000) ===

Batch 2: Time=40.000 ms, Mem=11.22 MB, Discarded=89960


=== Batch 3/4 for MemAppender(LinkedList) (maxSize=10000) ===

Batch 3: Time=40.000 ms, Mem=10.81 MB, Discarded=139940


=== Batch 4/4 for MemAppender(LinkedList) (maxSize=10000) ===

Batch 4: Time=44.000 ms, Mem=11.74 MB, Discarded=189920

MemAppender(LinkedList) [Summary]          10000     170.000    11.83          (Total
Discarded: 189920)


=== Batch 1/4 for MemAppender(ArrayList) (maxSize=100000) ===

Batch 1: Time=40.000 ms, Mem=12.16 MB, Discarded=0


=== Batch 2/4 for MemAppender(ArrayList) (maxSize=100000) ===

Batch 2: Time=50.000 ms, Mem=63.05 MB, Discarded=0


=== Batch 3/4 for MemAppender(ArrayList) (maxSize=100000) ===

Batch 3: Time=579.000 ms, Mem=0.00 MB, Discarded=49940


=== Batch 4/4 for MemAppender(ArrayList) (maxSize=100000) ===

Batch 4: Time=433.000 ms, Mem=112.66 MB, Discarded=99920

MemAppender(ArrayList) [Summary]          100000     1102.000   112.66          (Total
Discarded: 99920)


=== Batch 1/4 for MemAppender(LinkedList) (maxSize=100000) ===

Batch 1: Time=54.000 ms, Mem=0.00 MB, Discarded=0


=== Batch 2/4 for MemAppender(LinkedList) (maxSize=100000) ===

Batch 2: Time=66.000 ms, Mem=0.00 MB, Discarded=0
```

```
=== Batch 3/4 for MemAppender(LinkedList) (maxSize=100000) ===

Batch 3: Time=58.000 ms, Mem=47.30 MB, Discarded=49940


=== Batch 4/4 for MemAppender(LinkedList) (maxSize=100000) ===

Batch 4: Time=80.000 ms, Mem=0.00 MB, Discarded=99920

MemAppender(LinkedList) [Summary]          100000     258.000     47.30          (Total
Discarded: 99920)


=== Batch 1/4 for MemAppender(ArrayList) (maxSize=1000000) ===

Batch 1: Time=55.000 ms, Mem=236.19 MB, Discarded=0


=== Batch 2/4 for MemAppender(ArrayList) (maxSize=1000000) ===

Batch 2: Time=67.000 ms, Mem=4.44 MB, Discarded=0


=== Batch 3/4 for MemAppender(ArrayList) (maxSize=1000000) ===

Batch 3: Time=65.000 ms, Mem=0.00 MB, Discarded=0


=== Batch 4/4 for MemAppender(ArrayList) (maxSize=1000000) ===

Batch 4: Time=52.000 ms, Mem=0.00 MB, Discarded=0

MemAppender(ArrayList) [Summary]          1000000     239.000     236.19          (Total
Discarded: 0)


=== Batch 1/4 for MemAppender(LinkedList) (maxSize=1000000) ===

Batch 1: Time=49.000 ms, Mem=50.85 MB, Discarded=0


=== Batch 2/4 for MemAppender(LinkedList) (maxSize=1000000) ===

Batch 2: Time=54.000 ms, Mem=83.76 MB, Discarded=0


=== Batch 3/4 for MemAppender(LinkedList) (maxSize=1000000) ===

Batch 3: Time=74.000 ms, Mem=0.00 MB, Discarded=0


=== Batch 4/4 for MemAppender(LinkedList) (maxSize=1000000) ===
Batch 4: Time=54.000 ms, Mem=49.17 MB, Discarded=0

MemAppender(LinkedList) [Summary]              1000000     231.000     83.76            (Total
Discarded: 0)

ConsoleAppender(Dummy) [Summary]                N/A        313.000     416.92

FileAppender(Buffered) [Summary]                N/A        651.000     388.30


=== Layout Comparison (200k logs, 30 threads) ===

VelocityLayout Time (200k logs): 4947.617 ms

PatternLayout Time (200k logs): 2445.048 ms

Performance Ratio: VelocityLayout is 2.0 x slower
```