Introduction

Stress tests compared the performance of three logging configurations using both ArrayList and LinkedList implementations of MemAppender. Each configuration processed 10,000 log messages per test with varying maxSize values (10–10,000):

- **MemAppender + VelocityLayout**
- **ConsoleAppender + PatternLayout**
- **FileAppender + PatternLayout**

Execution Time

MemAppender + VelocityLayout consistently achieved the fastest performance, completing runs in **1–7 ms**.
ConsoleAppender was the slowest (**8–62 ms**) due to console I/O overhead.
FileAppender showed moderate efficiency (**16–25 ms**).
The choice between ArrayList and LinkedList had no significant impact on timing.

Memory Usage

MemAppender demonstrated the lowest memory consumption (**~1.3–3 MB**).
ConsoleAppender used the most memory (**~13 MB**), likely due to buffered console output handling.
FileAppender remained moderate (**~6–7 MB**).

Table 1 All configurations time (ms) and memory usage(KB)

ListType	MaxSize	Appender	Time(ms)	Memory(KB)
ArrayList	10	MemAppender + VelocityLayout	1	1510
ArrayList	10	MemAppender + VelocityLayout	2	1509
ArrayList	10	MemAppender + VelocityLayout	3	1291
ArrayList	10	MemAppender + VelocityLayout	7	3072
ArrayList	100	ConsoleAppender + PatternLayout	8	13295
ArrayList	100	ConsoleAppender + PatternLayout	10	13285
ArrayList	100	ConsoleAppender + PatternLayout	14	13579
ArrayList	1000	FileAppender + PatternLayout	16	6651
ArrayList	1000	FileAppender + PatternLayout	17	6672
ArrayList	1000	FileAppender + PatternLayout	17	6653
ArrayList	1000	FileAppender + PatternLayout	25	6462
ArrayList	100	ConsoleAppender + PatternLayout	62	916
LinkedList	10	MemAppender + VelocityLayout	1	1272
LinkedList	10	MemAppender + VelocityLayout	1	1500
LinkedList	10	MemAppender + VelocityLayout	2	1505
LinkedList	10	MemAppender + VelocityLayout	6	1513
LinkedList	100	ConsoleAppender + PatternLayout	8	13286
LinkedList	100	ConsoleAppender + PatternLayout	8	13287
LinkedList	100	ConsoleAppender + PatternLayout	9	133096
LinkedList	1000	FileAppender + PatternLayout	16	6660
LinkedList	1000	FileAppender + PatternLayout	16	6650
LinkedList	1000	FileAppender + PatternLayout	17	6650
LinkedList	1000	FileAppender + PatternLayout	20	6682
LinkedList	100	ConsoleAppender + PatternLayout	29	14143

VisualVM Observation

The VisualVM heap graph (Figure 1) confirms these results: heap usage rises in steps as each appender configuration runs, reflecting increased allocation from I/O-heavy operations.

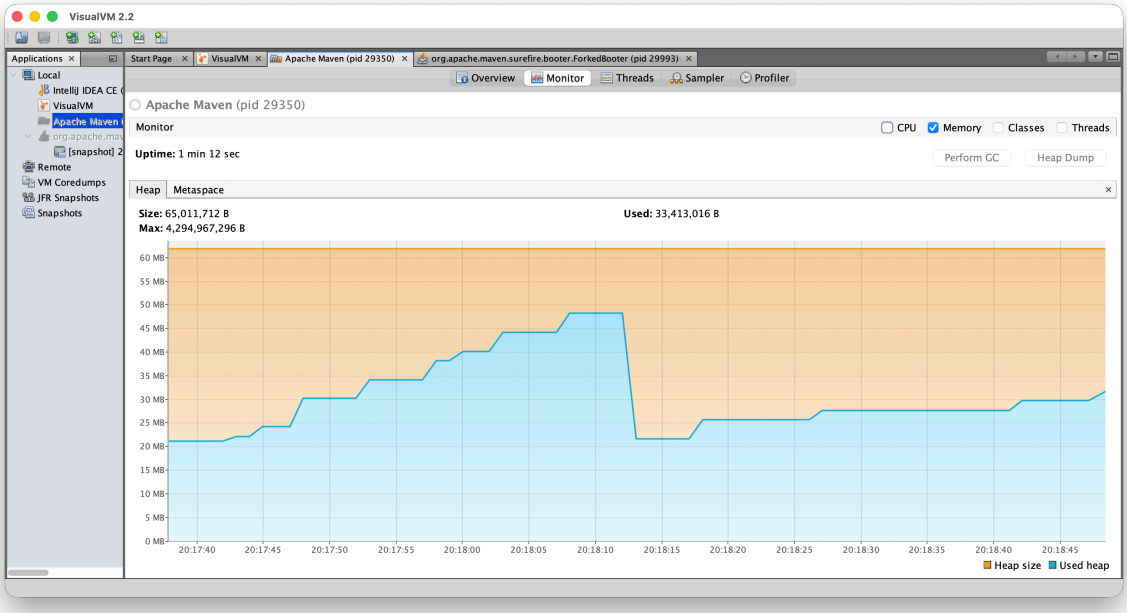

Figure 1 Heap usage all configurations

Summary

CRITERION	BEST PERFORMER	NOTES
SPEED	MemAppender + VelocityLayout	Fastest execution
MEMORY EFFICIENCY	MemAppender + VelocityLayout	Smallest heap footprint
HEAVIEST MEMORY USE	ConsoleAppender	Console I/O overhead
OVERALL BALANCE	FileAppender	Moderate time and memory use

Conclusion

The custom in-memory MemAppender provided the most efficient performance in both time and space. The FileAppender offered a practical middle ground, while the ConsoleAppender showed the least efficiency due to higher I/O costs.