Introduction

This report presents the results of benchmarking multiple string search algorithms and stress testing a string search server. The benchmarking evaluates algorithm efficiency across datasets of different sizes, while the stress test assesses server performance under increasing query loads. Together, these results provide insights into the most efficient algorithms and the server's scalability limits.

Setup

The evaluation was conducted on datasets containing 10,000; 100,000; 500,000; and 1,000,000 lines of randomly generated text. Six algorithms were benchmarked: Linear Scan, Generator Scan, Regex Match, Set Membership, Multithreaded Scan, and Binary Search. Each algorithm was tested in two modes: re-reading the file for each query and using a cached dataset. The stress test measured the server's query throughput (QPS) for different file sizes, stopping when the failure rate exceeded 10%.

Search Algorithms Overview

The benchmarked algorithms included: - Linear Scan: Sequentially scans the file for the query string. - Generator Scan: Uses a generator to yield lines for checking. - Regex Match: Uses compiled regular expressions for matching. - Set Membership: Loads lines into a set for O(1) membership checks. - Multithreaded Scan: Splits the file into chunks and scans in parallel. - Binary Search: Operates on sorted lines for logarithmic search time.

Performance Analysis

Benchmark results show that with caching (reread_on_query=False), Linear Scan and Generator Scan achieved the fastest times for large datasets, while Regex Match was slower due to pattern compilation overhead. Set Membership excelled when using cache, but required significant time when re-reading due to repeated set construction. Multithreaded Scan provided benefits for large datasets, and Binary Search was efficient for cached sorted data but slow when re-reading unsorted files. The stress test revealed that as QPS increased, total execution time grew linearly until the server's capacity limit was reached, at which point failure rates rose sharply.

Speed Test Report

Linear Scan

File Size	True (ms)	False (ms)
10000	3.25	147.00
100000	33.13	186.55
500000	168.13	134.18
1000000	424.53	150.32

Generator Scan

File Size	True (ms)	False (ms)

10000	4.15	129.87
100000	49.77	137.18
500000	248.07	116.99
1000000	569.98	128.81

Regex Match

File Size	True (ms)	False (ms)
10000	4.79	307.60
100000	51.97	295.76
500000	258.51	232.56
1000000	574.77	255.56

Set Membership

File Size	True (ms)	False (ms)
10000	2.98	0.02
100000	38.21	0.00
500000	224.18	0.00
1000000	953.88	0.00

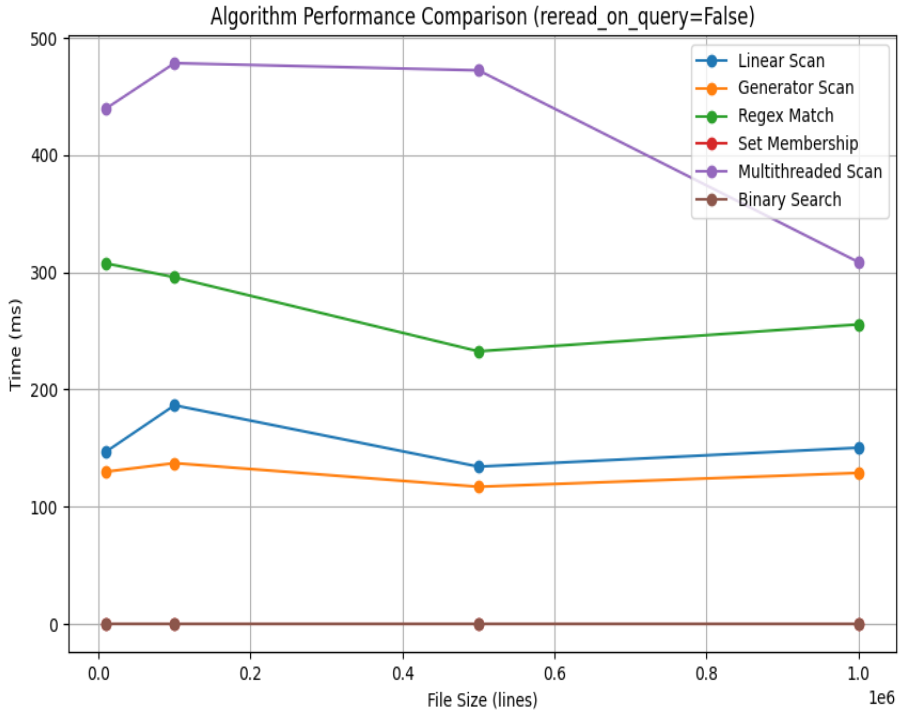
Multithreaded Scan

File Size	True (ms)	False (ms)
10000	199.86	439.85
100000	53.25	478.49
500000	710.06	472.36
1000000	744.09	308.67

Binary Search

File Size	True (ms)	False (ms)
10000	5.06	0.01
100000	65.68	0.01
500000	479.57	0.01
1000000	957.33	0.01

Overall Comparison Chart (reread_on_query=False)



String Search Server QPS Stress Test

This chart shows how execution time changes with QPS for various file sizes. A sharp drop in success rate marks the server's capacity limit.

Conclusions

The tests demonstrate that caching dramatically improves search performance for most algorithms. For large datasets, Linear Scan, Generator Scan, and Set Membership (cached) provide the best balance of speed and simplicity. Regex Match is most useful when pattern flexibility is required. The server stress test highlights the importance of optimizing for QPS handling and monitoring failure rates to avoid performance degradation under load.