Analysis of Light Actor Go Framework Benchmark Results

In-Process Actor Benchmark

Description: This benchmark test utilizes code initially employed by the Proto.Actor framework for in-process testing. However, to ensure a fair comparison, we executed the test locally. We adapted the exact codebase used by Proto.Actor to test our custom Light Actor Go Framework. The results of our framework can also be compared to Hollywood actor framework.

Light Actor Go Framework Results:

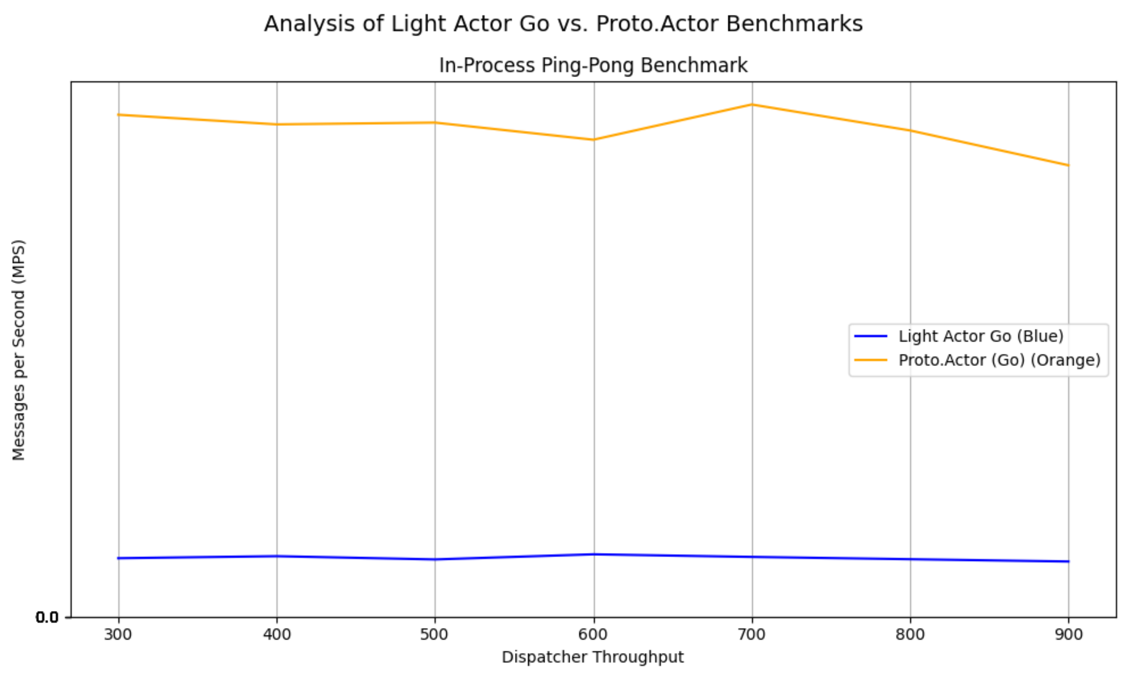
Throughput	Elapsed Time	Messages per sec
300	5.0218053s	3,186,105
400	4.8539012s	3,296,317
500	5.1289654s	3,119,537
600	4.7077577s	3,398,645
700	4.9131583s	3,256,561
800	5.1094147s	3,131,474
900	5.3216761s	3,006,571

ProtoActor Framework Results:

Throughput	Elapsed Time	Messages per sec
300	585.2226ms	27,340,024
400	596.6617ms	26,815,866
500	594.532ms	26,911,926
600	615.8854ms	25,978,860
700	573.5071ms	27,898,522
800	604.144ms	26,483,754
900	650.7483ms	24,587,078

Hollywood Framework Results:

- Messages per second: ~3,511,664



Analysis: The ProtoActor framework significantly outperforms the Light Actor Go framework in terms of messages processed per second. This performance difference is consistent across all throughput levels, with the ProtoActor framework achieving over eight times the messages per second at the highest throughput level tested. The Hollywood framework performs on average slightly better than the Light Actor Go framework in terms of messages per second in an in-process scenario.

Remote Actor Benchmark

For this benchmark test, we used custom code to evaluate the performance of remote message processing. This test measures the actor framework's efficiency in sending and processing messages between local and remote actors under controlled conditions. We used the same codebase to test both our custom framework and the Proto.Actor framework to compare their performance.

Light Actor Go Framework Results:

Processed Messages	Elapsed Time (ms)	Messages per second
100	71.3487	1401.567232
100	73.6918	1357.003086
100	62.9371	1588.887953
100	96.6997	1034.129372
100	137.7655	725.871136
100	104.9728	952.627728
100	134.5064	743.459047

100	61.0701	1637.462523
100	60.0227	1666.036350

ProtoActor Framework Results:

Processed Messages	Elapsed Time (ms)	Messages per second
100	8.0497	12422.823211
100	18.4204	5428.763762
100	8.752	11425.959781
100	20.4513	4889.664716
100	11.0052	9086.613601
100	9.8179	10185.477546

Analysis: Again, the ProtoActor framework performs significantly better than the Light Actor Go framework in the remote actor benchmark. The ProtoActor framework processes messages at a rate almost an order of magnitude higher than the custom framework, highlighting the efficiency and optimization differences.

Actor Spawn Benchmark

Description: This benchmark test measures the time taken to spawn a large number of actors and perform a computation. Although the code differs significantly from that used in Proto.Actor's benchmark, it performs the same task and produces the expected results.

Light Actor Go Framework Results:

- Time taken: ~14.5sec

ProtoActor Framework Results:

- Time taken: ~1.2 sec

Analysis: The results indicate that the ProtoActor has superior optimizations for actor spawning and computation execution, making it more efficient for applications requiring rapid actor creation and message handling.

Conclusion

While the ProtoActor framework showcases superior performance across various benchmarks, the Light Actor Go framework demonstrates commendable throughput and efficiency, particularly in in-process message handling. Its results in these areas suggest promising scalability and suitability for certain use cases, but with some performance gaps compared to ProtoActor.