Performance Impact of COMPMSG on MQ Sender/Receiver Channels

Objective

This paper explores the impact of IBM MQ's COMPMSG (compressed message) feature when enabled on server and receiver channels. The primary objective is to evaluate and compare its effect on message throughput and performance across different message sizes using various compression algorithms—specifically, ZLIBFAST and LZ4FAST. The evaluation is performed under a distributed Requester/Responder test model with multiple queues and server channel pairs.

We compare results with and without COMPMSG enabled to assess its efficiency and trade-offs. These insights will help determine which compression algorithm best suits your performance requirements in high-throughput IBM MQ environments.

Test Environment

Component	Version/Details
Platform	Linux x86-64
IBM MQ Version	9.4.3
Network Bandwidth	10 Gb/s
Queue Mode	Bindings Mode (non-transactional)
Topology	Distributed queues with multiple server/receiver channel pairs
Test Type	Requester/Responder

Test Configuration

- Message Sizes: 2 KB, 20 KB, and 200 KB, 2M(Latency)
- All performance tests were conducted using 10 Gb network links between systems, ensuring consistent and reliable measurement.
- Each round trip consists of 2 PUT operations and 2 GET operations.
- In the baseline run, COMPMSG is switched off and no channel-level compression is applied, every other channel setting, hardware spec, and network condition matches those used in the compression tests.
- The compression rates reported in this paper were obtained from the COMPRATE field in IBM MQ channel status, with channel monitoring enabled.
- Channel Configuration:
    - Sender and Receiver channels with/without COMPMSG enabled
    - Multiple queue pairs for sending and receiving messages
- Compression Algorithms Tested: ZLIBFAST, LZ4FAST
- Baseline Test:
    - Channels with COMPMSG disabled to evaluate uncompressed throughput
- Concurrency:
    - Simultaneous message transfers over multiple server channel pairs

Results

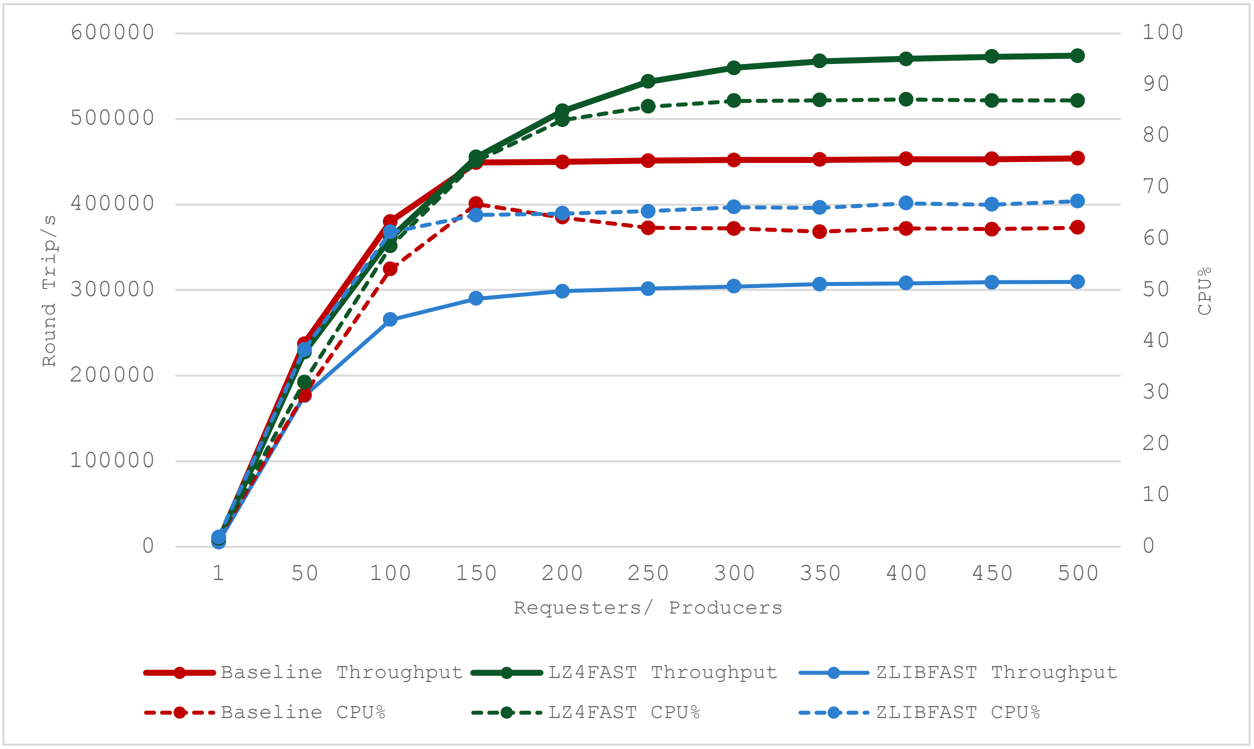

Figure 1 –2 k non-persistent messaging

Compression Efficiency & CPU Impact – 2 k Messages

- LZ4FAST achieved the highest throughput but, consumed higher CPU at ~90%, spending processor overhead to gain higher message rate.
- Baseline delivered lower throughput than LZ4FAST but maintained lower CPU usage, making it a lightweight option with reasonable performance.
- ZLIBFAST resulted in the lowest throughput and the higher CPU consumption than Baseline, indicating that its compression overhead was not well-suited for small message sizes.

Config	Throughput (k Round trip/s)	CPU % @ peak	Compression rate
Baseline	454	62.17	NA
LZ4FAST	574	86.9	79%
ZLIBFAST	311	67.33	62%

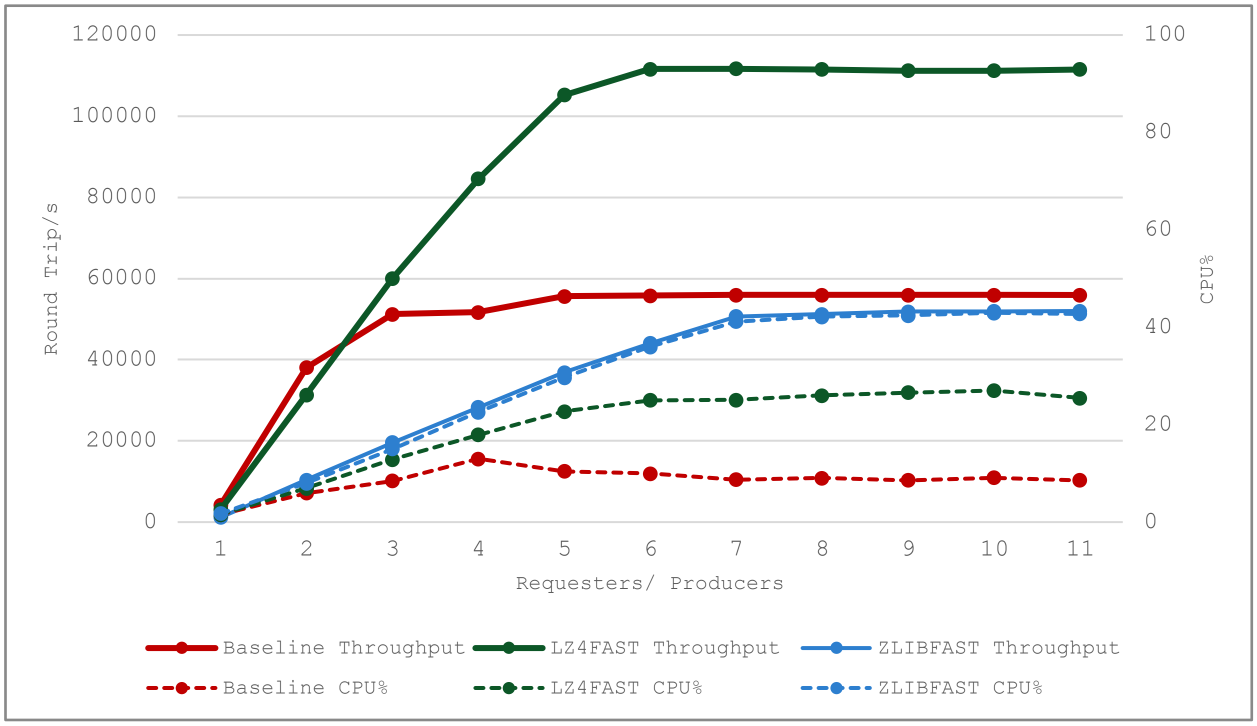

Figure 2 –20 k non-persistent messaging

Compression Efficiency & CPU Impact – 20 k Messages

- LZ4FAST achieves the highest throughput (peaking around 112 k Round trip/s) and scales efficiently with increasing clients, showcasing strong compression performance with moderate CPU usage.
- Baseline offers stable performance with minimal CPU overhead, plateauing around 56 k Round trip/s.
- ZLIBFAST shows consistent scaling but reaches lower throughput (~52 k Round trip/s) with noticeably higher CPU consumption, while LZ4FAST achieves significantly higher throughput with more efficient CPU utilization.

Config	Throughput (k Round trip/s)	CPU % @ peak	Compression rate
Baseline	56	13	NA
LZ4FAST	112	27	50%
ZLIBFAST	52	42.74	37%

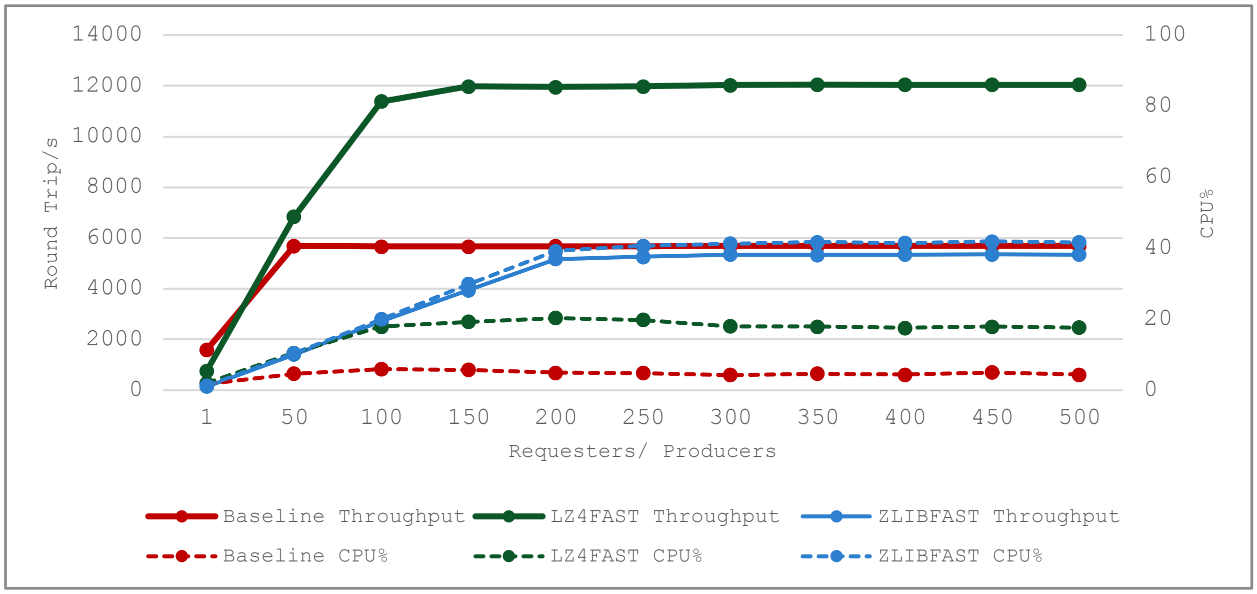

Figure 3 –200 k non-persistent messaging

Compression Efficiency & CPU Impact – 200 k Messages

- LZ4FAST delivers the highest throughput, reaching ~12 k Round trip/s, and demonstrates good scalability with larger message sizes. Despite a moderate increase in CPU usage (~18%), it clearly outperforms both Baseline and ZLIBFAST, proving that LZ4FAST's compression remains efficient even as payload sizes grow.
- Baseline provides stable but limited throughput (~5.7 k Round trip/s) with minimal CPU impact, indicating it is lightweight but constrained in its ability to scale with larger messages.
- ZLIBFAST delivers comparable throughput to Baseline (~5.3 k Round trip/s), but at the cost of significantly higher CPU usage (~42%), indicating increased processing overhead for larger messages.

Config	Throughput (k Round trip/s)	CPU % @ peak	Compression rate
Baseline	5.7	5.7	NA
LZ4FAST	12	18	48%
ZLIBFAST	5.3	41.9	36%

Compression Impact Under Network Latency

(Message Sizes:2 k, 20 k, 200 k and 2 M | Latency: 1 ms, 10 ms, 50 ms total round-trip)

To evaluate performance under network delays, symmetric latency was introduced on both links (e.g., 1 ms total = 0.5 ms each direction). In the following tests, we compare the results of Baseline and LZ4FAST, as LZ4FAST performs better than ZLIBFAST in previous scenarios. The results cover latency values of 1 ms, 10 ms, and 50 ms, focusing on both throughput and CPU impact.

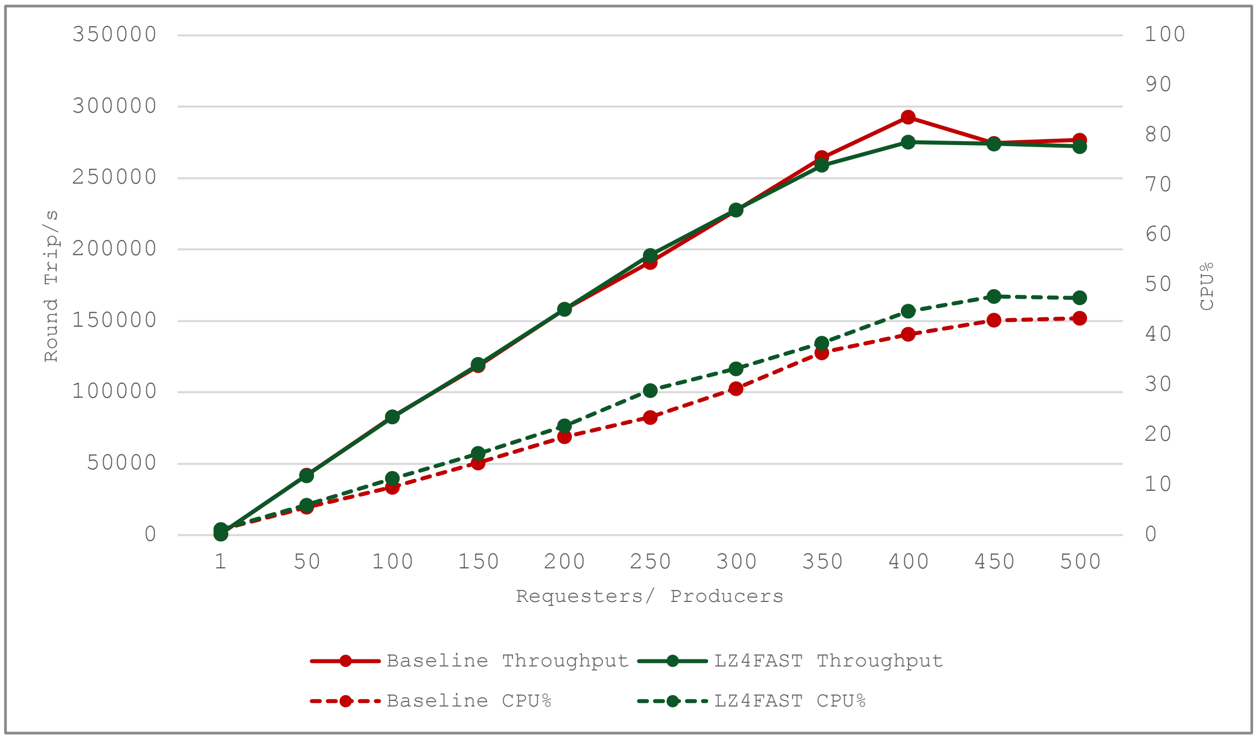

Figure 4 –2 k non-persistent messaging with 1 ms latency

Compression Efficiency & CPU Impact – 1 ms latency,2 k message size

- Throughput for both Baseline and LZ4FAST scales well, peaking around 270 k Round trip/s at 500 clients.
- LZ4FAST matches Baseline in throughput across all client loads, showing no performance drop due to compression in lower message sizes, with 1 ms network latency the workload is latency bound so compression has no improvement on the rate.
- CPU usage is slightly higher for LZ4FAST (~48%) compared to Baseline (~43%) at peak load.
- In lower message sizes under low-latency conditions, Baseline and LZ4FAST perform similarly, with minimal differences in throughput.

Config	Throughput (k Round trip/s)	CPU % @ peak	Compression rate
Baseline	276.7	43	NA
LZ4FAST	272	47.7	79%

•

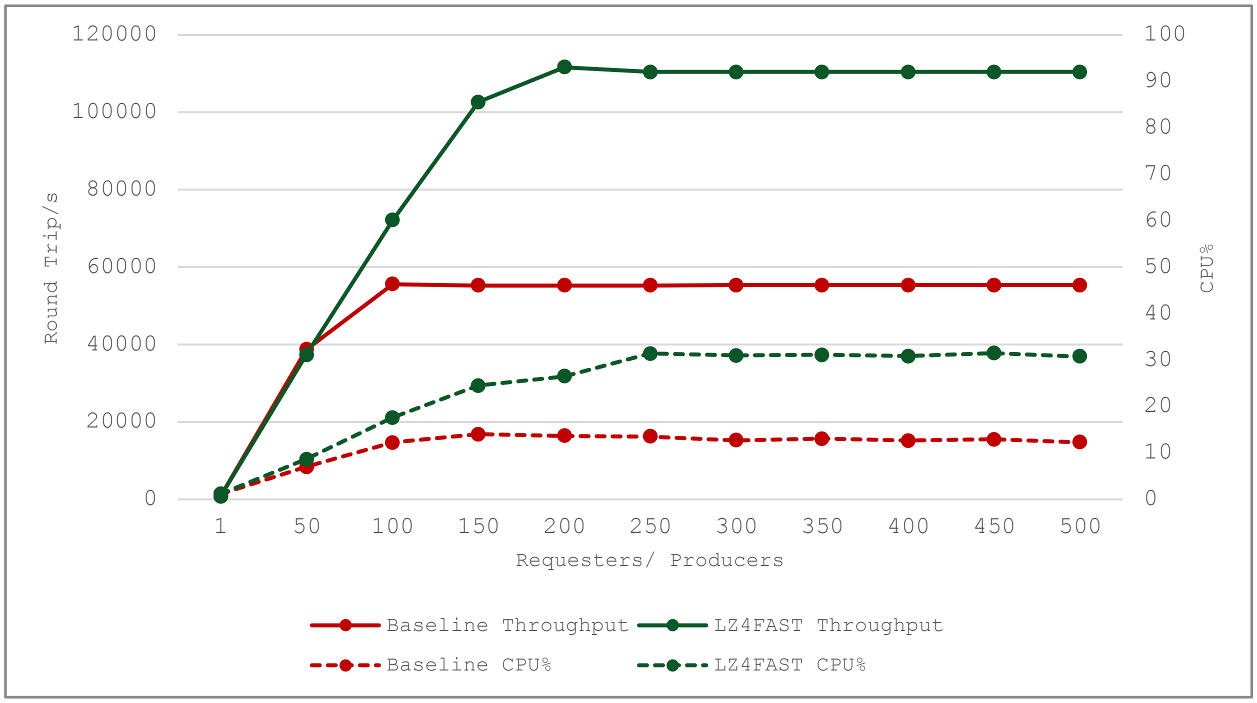

Figure 5 – 20 k non-persistent messaging with 1 ms latency

Compression Efficiency & CPU Impact – 1 ms latency,20 k message size

- LZ4FAST achieves nearly double the throughput of Baseline, peaking at ~111 k Round trip/s.
- Baseline throughput plateaus early around ~55 k Round trip/s, as no. of clients increases.
- LZ4FAST incurs higher CPU usage, peaking at ~30% compared to Baseline's ~13%.
- The increase in CPU usage reflects the additional processing required for compression.
- Best suited for environments requiring high throughput with moderate CPU availability under low-latency.

Config	Throughput (k Round trip/s)	CPU % @ peak	Compression rate
Baseline	55	13.6	NA
LZ4FAST	111	31.4	48%

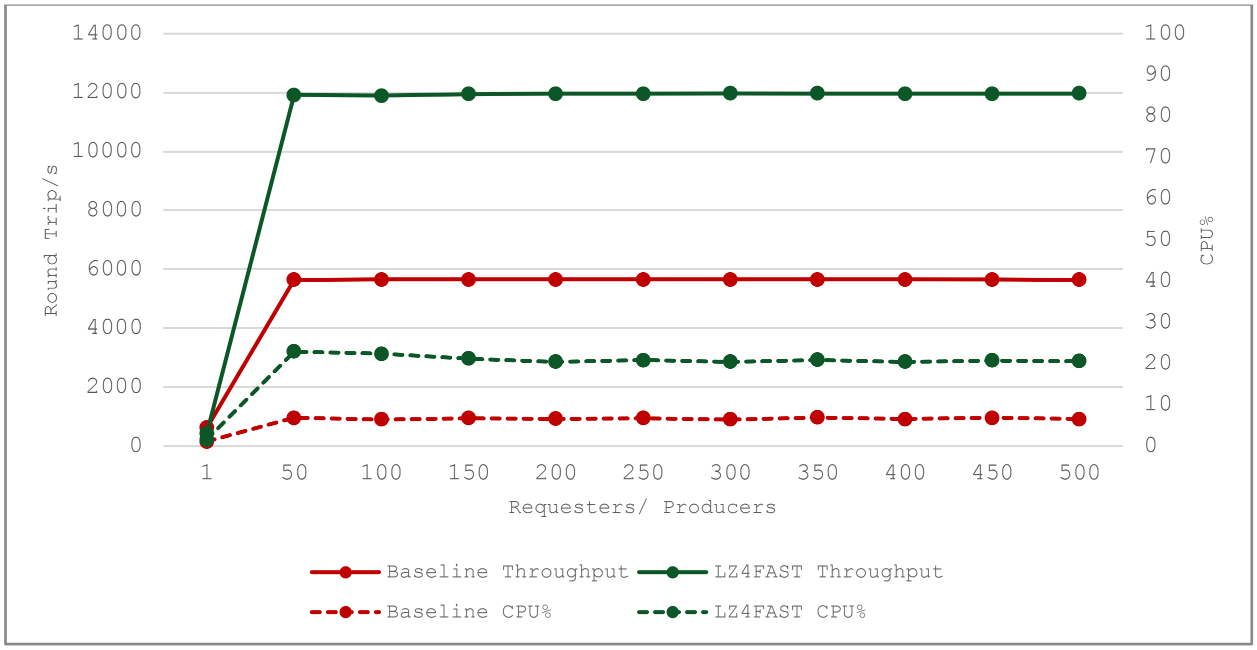

Figure 6 –200 k non-persistent messaging with 1 ms latency

Compression Efficiency & CPU Impact – 1 ms latency,200 k message size

- LZ4FAST consistently outperforms Baseline, achieving throughput in the range of ~11.9 k Round trip/s, whereas Baseline stabilizes at ~5.6 k Round trip/s, showing almost 2× improvement with LZ4FAST.
- Throughput scales well with client load for LZ4FAST, remaining stable even at high concurrency (up to 500 clients), while Baseline flattens early after 100 clients.
- LZ4FAST CPU% gradually increases and peaks at ~20%, indicating efficient compression overhead.
- Baseline CPU% stays significantly lower, around ~6%, but with lower message rates, suggesting limited utilization of system resources.
- Although LZ4FAST uses moderately more CPU, it provides nearly double the throughput compared to Baseline.
- The increase in CPU usage remains controlled and proportional to the improved performance, showing efficient compression behaviour under low latency.

Config	Throughput (k Round trip/s)	CPU % @ peak	Compression rate
Baseline	5.6	6.8	NA
LZ4FAST	11.9	21.2	48%

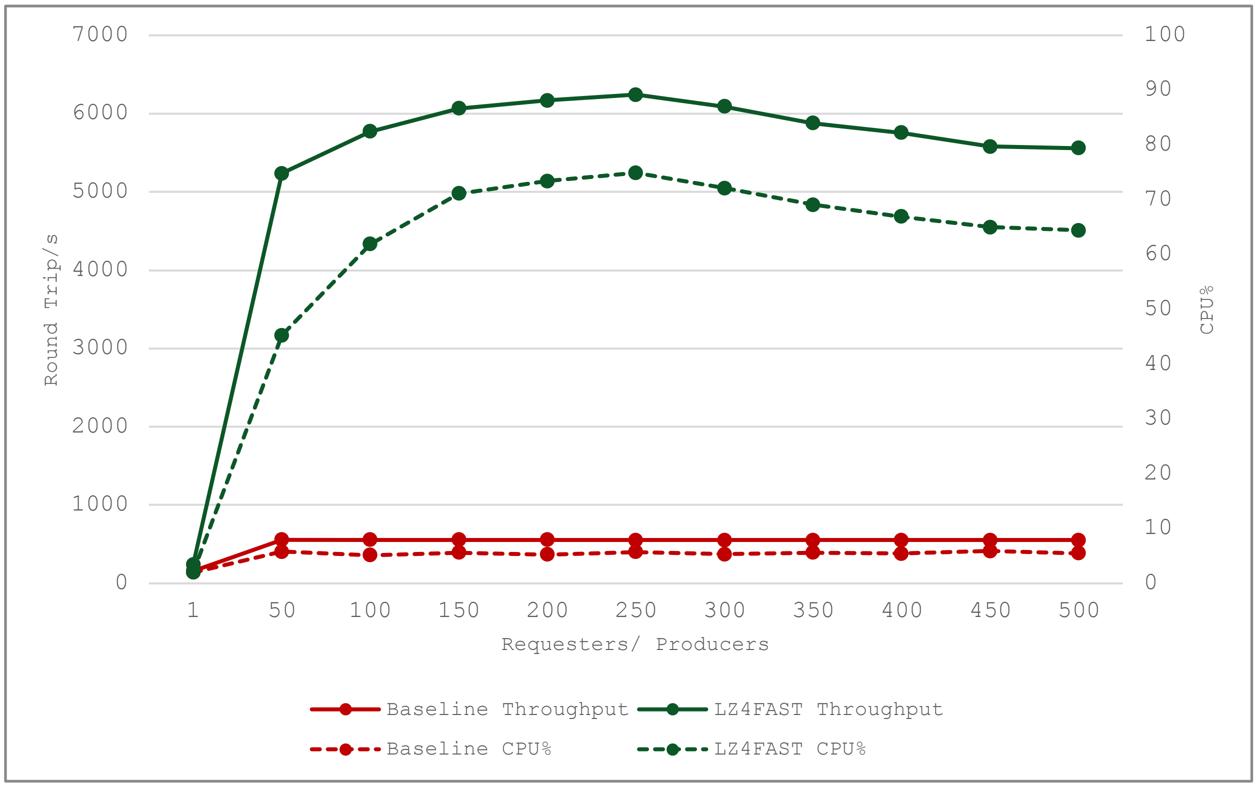

Figure 7 –2 M non-persistent messaging with 1 ms latency

Compression Efficiency & CPU Impact – 1 ms latency,2 M message size

- Throughput for LZ4FAST reaches a peak of ~6 k Round trip/s around 250 clients, then gradually declines, stabilizing near ~5 k Round trip/s.
- Baseline throughput saturates early at ~550 msgs/sec, remaining flat across all client loads.
- LZ4FAST shows significant compression benefit, delivering ~10x higher throughput than Baseline at all scales.
- CPU usage for LZ4FAST rises to ~70% as client load increases, while Baseline stays flat around ~6%.
- The higher CPU usage for LZ4FAST is expected due to compression, but is justified by the substantial gain in message rate.

Config	Throughput (k Round trip/s)	CPU % @ peak	Compression rate
Baseline	0.5	5.8	NA
LZ4FAST	6.2	73.4	3%

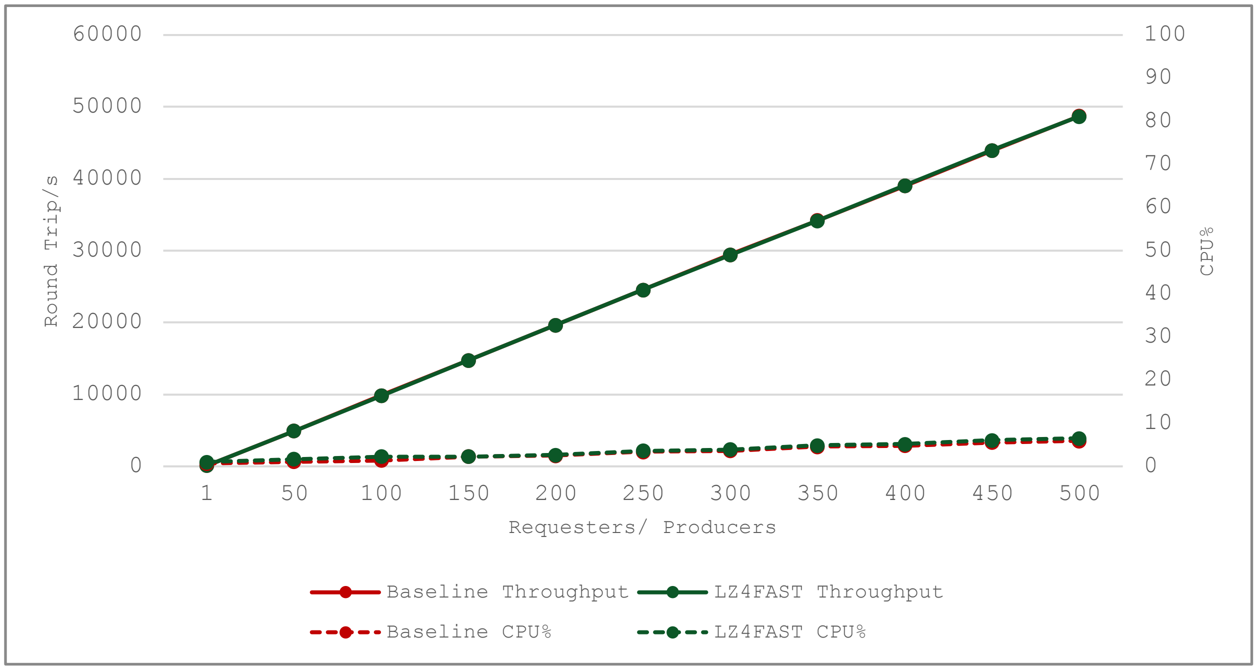

Figure 8 –2 k non-persistent messaging with 10 ms latency

Compression Efficiency & CPU Impact – 10 ms latency,2 k message size

- Throughput for both Baseline and LZ4FAST scales identically, peaking around 48.7 k Round trip/s at 500 clients.
- LZ4FAST matches Baseline at every load point, confirming no performance gain or drop due to compression for small messages under low-latency conditions.
- At 500 clients, Baseline CPU usage averages around 5.93%, while LZ4FAST averages 6.53%, showing only a marginal increase of ~0.6% with compression.
- Overall, compression provides no clear benefit or penalty in this 2 k message size test under 10 ms latency conditions.

Config	Throughput (k Round trip/s)	CPU % @ peak	Compression rate
Baseline	48.7	5.9	NA
LZ4FAST	48.6	6.5	79%

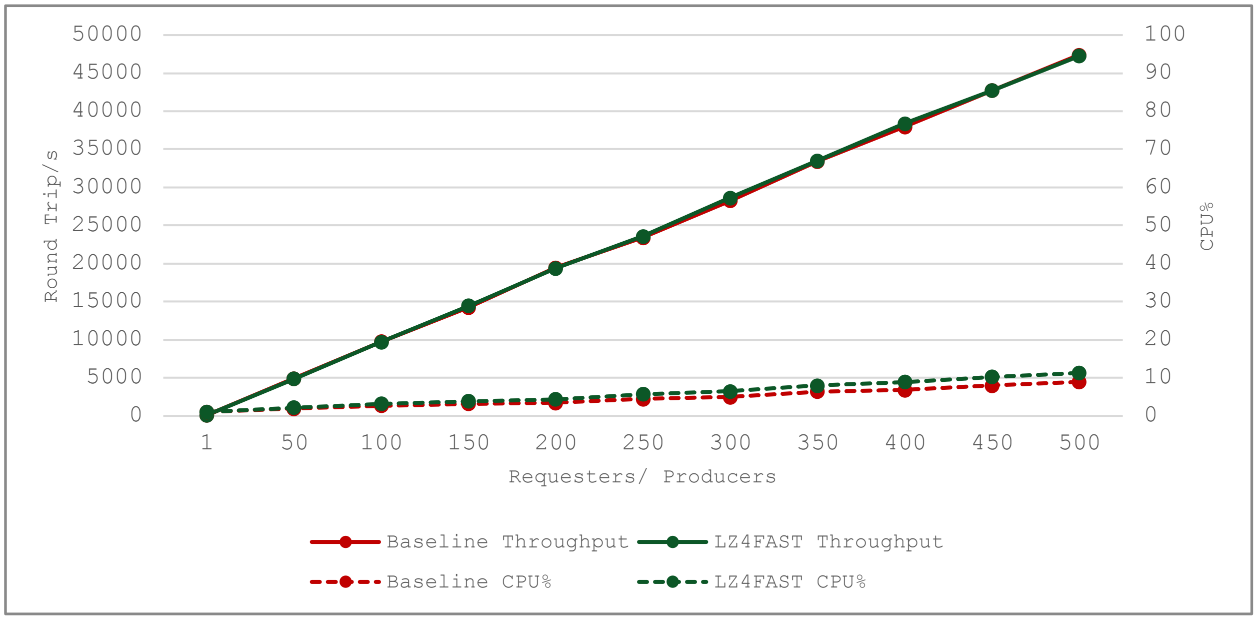

Figure 9 –20 k non-persistent messaging with 10 ms latency

Compression Efficiency & CPU Impact – 10 ms latency,20 k message size

- Throughput scales linearly for both Baseline and LZ4FAST, reaching ~47 k Round trip/s at 500 clients.
- Compression has no impact on throughput, LZ4FAST consistently matches Baseline performance across all loads.
- CPU usage is nearly identical for both configurations — averaging between 8–11%, even at peak load.
- This test shows that for 20 k messages under 10 ms latency, LZ4FAST compression achieves equivalent performance with minimal CPU overhead, making it a suitable optimization without trade-offs.

Config	Throughput (k Round trip/s)	CPU % @ peak	Compression rate
Baseline	47.3	8.9	NA
LZ4FAST	47.2	11.2	50%

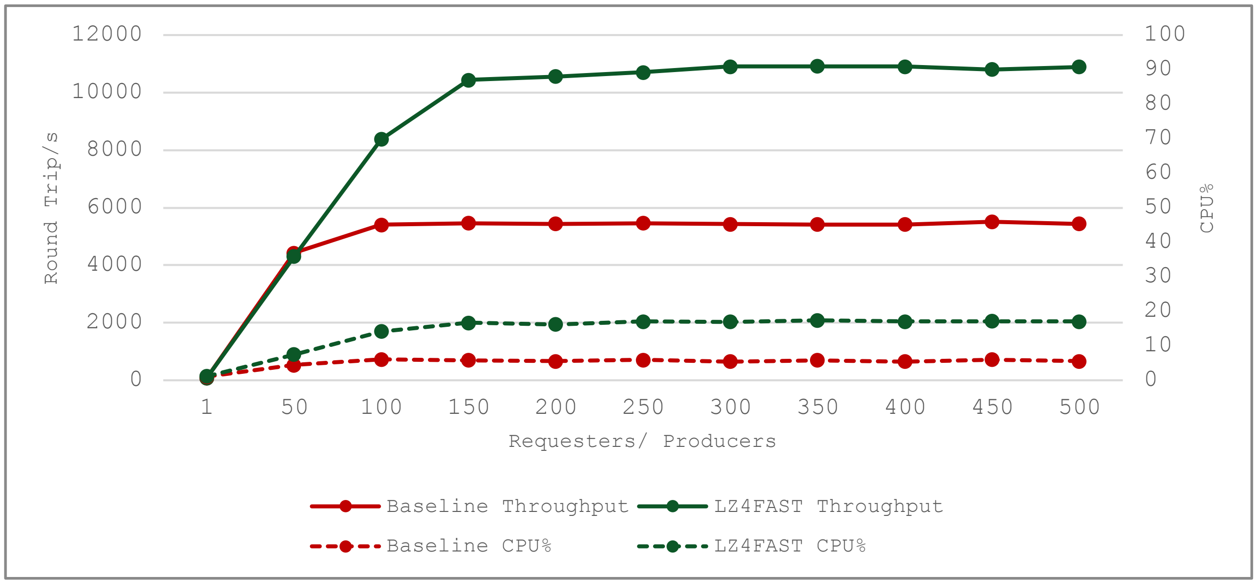

Figure 10 –200 k non-persistent messaging with 10 ms latency

Compression Efficiency & CPU Impact – 10 ms latency,200 k message size

- LZ4FAST consistently outperforms Baseline in throughput, peaking over ~11 k Round trip/s at 500 requesters, while Baseline plateaus near ~5 k Round trip/s.
- CPU usage for LZ4FAST is modest, peaking around 17%, compared to ~6% for Baseline at peak load—showing minimal CPU overhead for the added throughput.
- LZ4FAST maintains continuous throughput growth with increasing producer counts, demonstrating excellent scalability for large messages.
- Baseline throughput flattens beyond 150 clients, while LZ4FAST continues to scale effectively up to 500, making it more suitable for high-volume, large-message environments.

Config	Throughput (k Round trip/s)	CPU % @ peak	Compression rate
Baseline	5.5	5.8	NA
LZ4FAST	10.9	17.3	48%

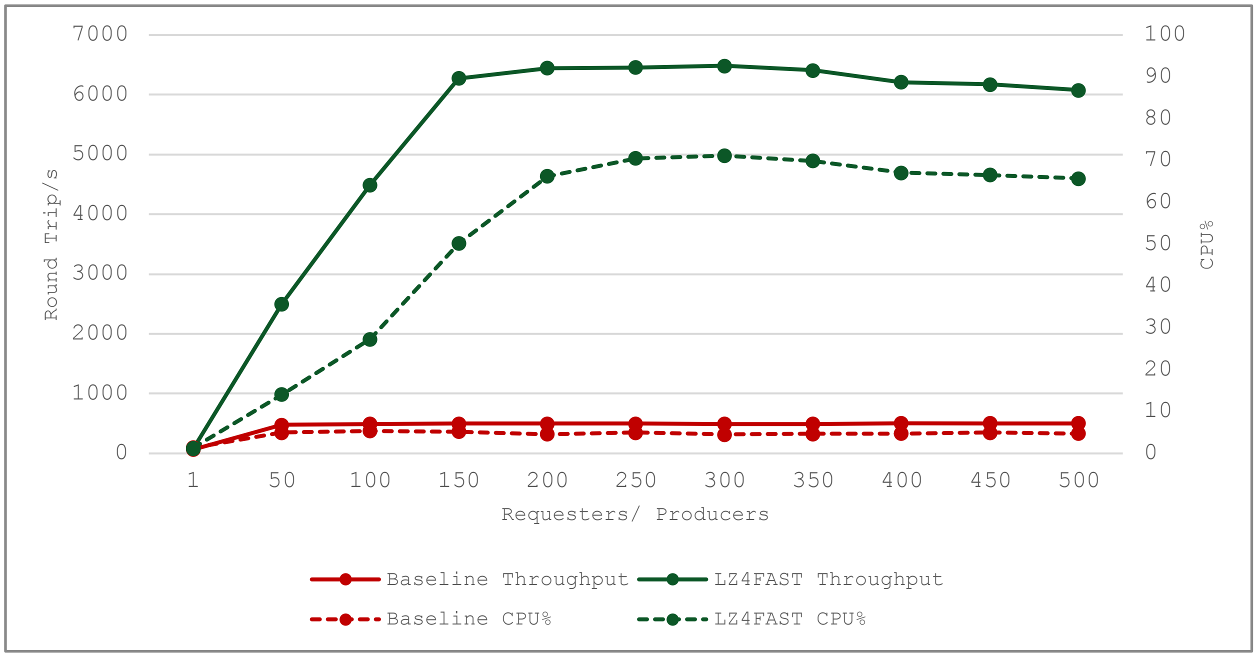

Figure 11 –2 M non-persistent messaging with 10 ms latency

Compression Efficiency & CPU Impact – 10 ms latency,2 M message size

- LZ4FAST shows significantly higher throughput than Baseline across all producer counts, peaking around ~6 k Round trip/s at 300 clients.
- Baseline throughput saturates early, plateauing around 500 msgs/sec, showing limited scalability with increasing load.
- CPU usage for LZ4FAST increases sharply, reaching 71%, while Baseline CPU% remains below 10%, indicating low resource utilization.
- High message sizes under moderate latency amplify the benefits of compression—LZ4FAST effectively trades off CPU for throughput, making it ideal for throughput-bound systems.

Config	Throughput (k Round trip/s)	CPU % @ peak	Compression rate
Baseline	0.5	5.3	NA
LZ4FAST	6.4	71.1	8%

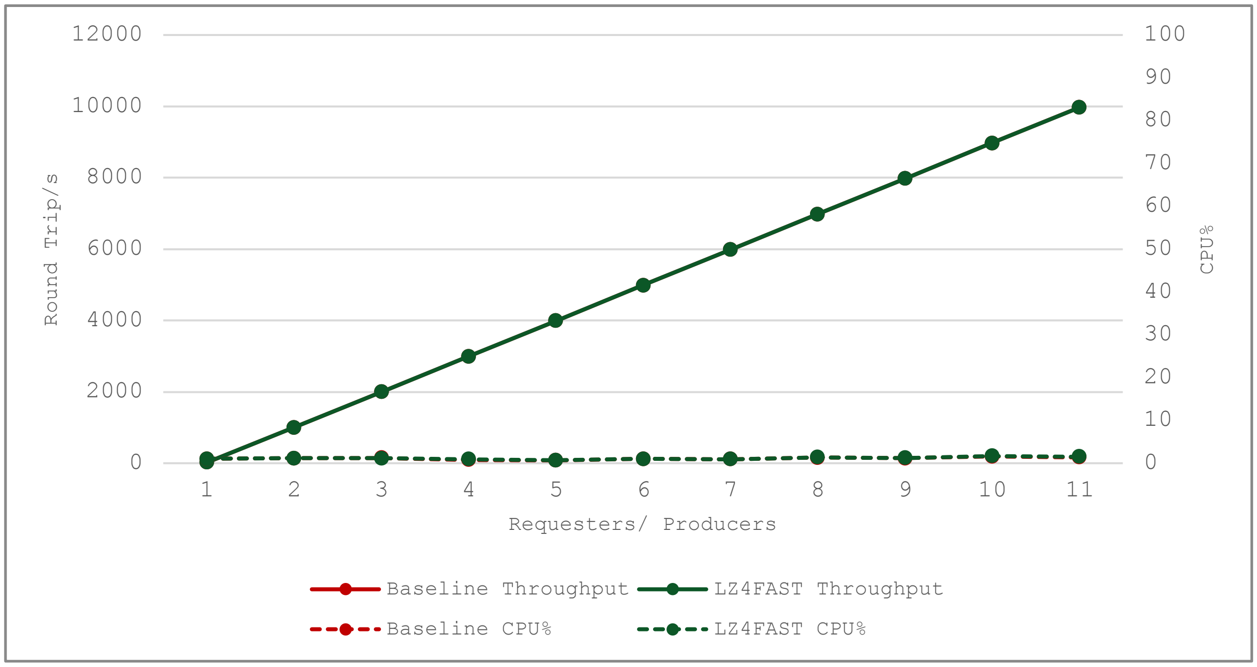

Figure 12 –2 k non-persistent messaging with 50 ms latency

Compression Efficiency & CPU Impact – 50 ms latency,2 k message size

- LZ4FAST and Baseline are practically identical across all load points, topping out just under 10 k Round trip/s.
- CPU Usage for both stay very low (~ 1–2 %), showing no appreciable CPU overhead from compression.
- For small messages under high latency, compression has limited impact on throughput. Latency becomes the dominant bottleneck, keeping CPU usage low for both Baseline and LZ4FAST.

Config	Throughput (k Round trip/s)	CPU % @ peak	Compression rate
Baseline	9.9	1.5	NA
LZ4FAST	9.9	1.6	79%

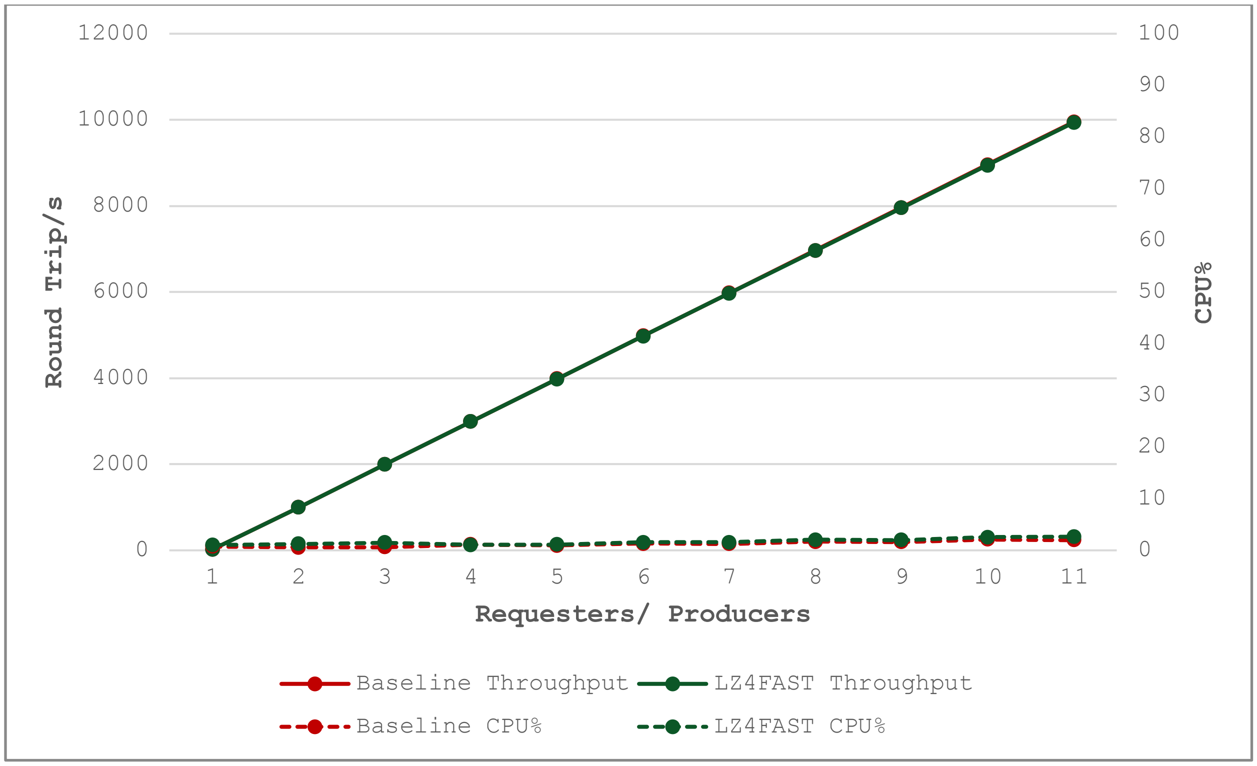

Figure 13 –20 k non-persistent messaging with 50 ms latency

Compression Efficiency & CPU Impact – 50 ms latency,20 k message size

- Baseline and LZ4FAST deliver nearly identical throughput across all requester loads, scaling linearly from ~20 round trips/sec at 1 requester to ~10 k round trips/sec at 500 requesters.
- CPU usage remains low for both configurations, with Baseline peaking at ~2.1% and LZ4FAST peaking at ~2.6%, indicating that compression adds only a small CPU cost at this message size and latency.
- Both maintain a strong linear scaling trend in throughput and CPU up to 500 requesters, with no signs of early saturation.
- The throughput trend is consistent with previous tests at 1 ms and 10 ms latency, indicating that for small message sizes over high-latency links, compression provides no measurable throughput advantage over uncompressed transfers, while keeping CPU cost only marginally higher.

Config	Throughput (k Round trip/s)	CPU % @ peak	Compression rate
Baseline	9.9	2.1	NA
LZ4FAST	9.9	2.6	57%

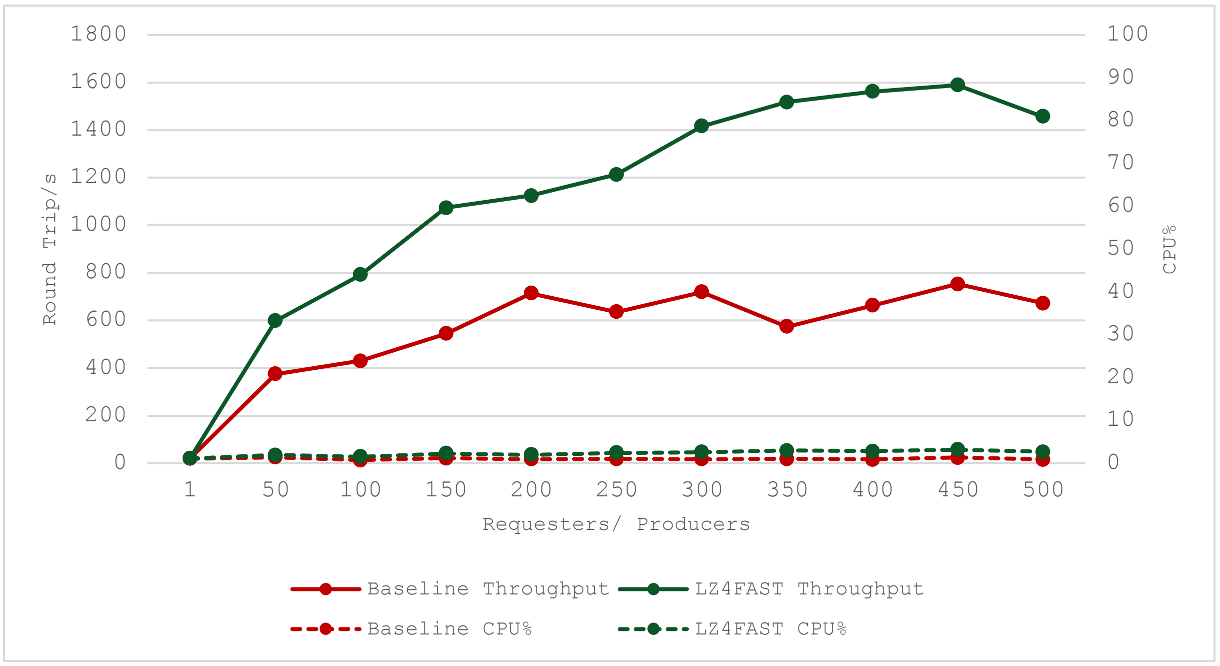

Figure 14 –200 k non-persistent messaging with 50 ms latency

Compression Efficiency & CPU Impact – 50 ms latency,200 k message size

- Both Baseline and LZ4FAST reach peak throughput at 300-350 requesters, with LZ4FAST achieving ~1600 msgs/sec and Baseline ~700 msgs/sec, showing that LZ4FAST significantly outperforms Baseline under peak conditions .
- In this test, CPU usage is consistently low (~1–3%) for both Baseline and LZ4FAST. This is because large message sizes reduce the frequency of processing, and the 50 ms network latency causes the system to wait on I/O rather than compute. The workload becomes network-bound, not CPU-bound. As a result, both configurations utilize minimal CPU resources, and the additional compression overhead from LZ4FAST remains negligible in this scenario.
- Overall, LZ4FAST provides better throughput and maintains efficiency under heavy load and high-latency conditions, while Baseline shows more variability.

Config	Throughput (k Round trip/s)	CPU % @ peak	Compression rate
Baseline	0.7	1.3	NA
LZ4FAST	1.6	3.1	54%

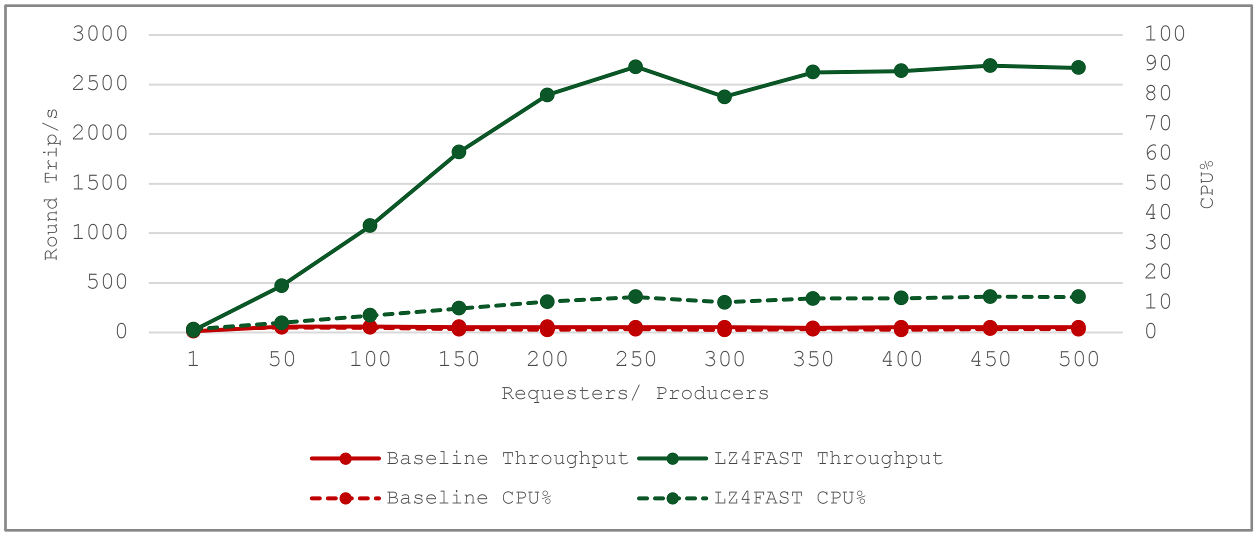

Figure 15 –2 M non-persistent messaging with 50 ms latency

Compression Efficiency & CPU Impact – 50 ms latency,2 M message size

- LZ4FAST reaches a peak throughput of ~2 k Round trip/s at 450 producers, clearly outperforming Baseline, which peaks at ~58 messages/sec at 50 producers.
- Baseline throughput remains consistently low and shows no scalability beyond initial client counts, while LZ4FAST demonstrates significant scaling with increasing clients.
- The CPU usage for LZ4FAST steadily rises with load (peaking over 12%), indicating compression overhead, whereas Baseline maintains low CPU usage throughout.
- The performance gap between LZ4FAST and Baseline remains substantial across all client loads, showcasing the effectiveness of compression for large message sizes under high latency conditions.

Config	Throughput (k Round trip/s)	CPU % @ peak	Compression rate
Baseline	0.05	1.6	NA
LZ4FAST	2.7	12	8%

Conclusion

Enabling COMPMSG significantly changes the performance of IBM MQ sender/receiver channels. In every scenario we tested, channel compression introduced no significant overhead for small-payload, low-latency traffic, and yielded clear benefits once either the payload size grew or round-trip latency increased.

Of the two algorithms assessed, LZ4FAST offered the strongest combination of higher throughput and modest CPU demand. It consistently out-performed the uncompressed baseline for medium- and large-payload workloads and maintained this advantage even with network delays up to 50 ms. ZLIBFAST achieved the highest raw compression ratio, but its extra CPU usage offset that gain in most rate-bound situations.

If primary goal is higher throughput, LZ4FAST offers the most consistent gains—especially when messages are medium-to-large (≈ 20 kb or bigger) or the network constrained environment. For small-sized messages on very low-latency links, the uncompressed baseline delivers equivalent performance, so enabling compression there adds little value.