Evaluation Report

Sahithi Banda (sqb6440) **Arya Amit Mohite (aam7044)**

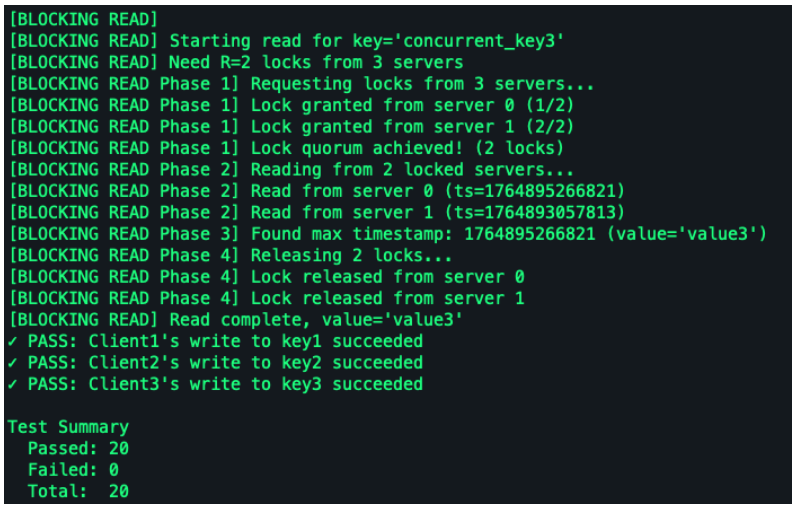
This report presents a comprehensive evaluation of the distributed key-value store implementation, comparing the ABD and Blocking protocols across correctness, performance, and fault tolerance metrics. We used W135 machines for testing. For more details of implementation and design, please check readme and design files in the github repository or the code provided.

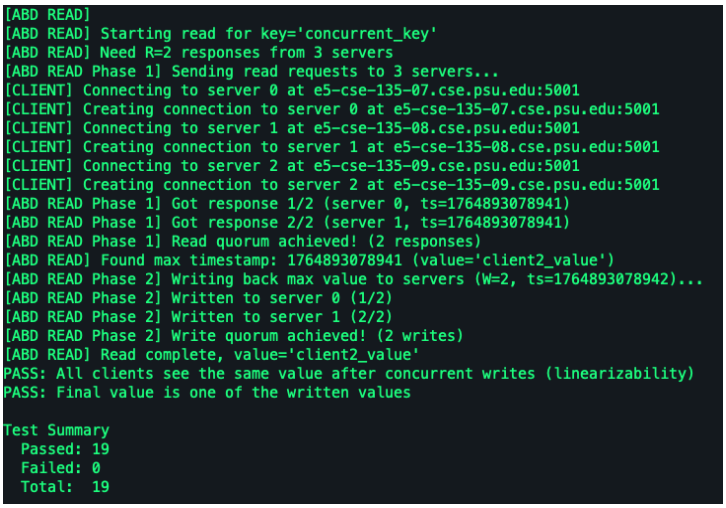
1. Correctness Testing

To verify the correctness of both protocols, we implemented test suites that validate fundamental correctness properties including linearizability, consistency, and proper handling of edge cases. The tests were executed on single-server, 3-server, and 5-server configurations to ensure correctness across different deployment scales.

The testing files comprehensively validate both the ABD (non-blocking) and Blocking (lock-based) protocols across a wide range of scenarios to ensure correctness, consistency, and robustness. For ABD, the test suite includes cases that verify basic write/read behavior, overwrites, multi-key independence, handling of empty or non-existent keys, and strong consistency guarantees through concurrent writes and read-after-write visibility checks. It also tests sequential operations, large payload handling, and support for special characters. The Blocking protocol suite mirrors many of these functional checks, basic operations, overwrites, multiple keys, empty/non-existent values, but additionally emphasizes lock correctness, including mutual exclusion, sequential consistency after lock release, and parallelism across different keys. Together, these test cases ensure that both protocols behave correctly under normal, edge-case, and concurrent execution scenarios. You can find these test cases in the `/tests` folder in the codebase.

All test cases passed successfully for both ABD and Blocking protocols across all server configurations (1, 3, and 5 servers). Below are a sample test case for 3 servers on both the protocols along with the total results.





2. Performance Evaluation: Saturation Throughput and Latency Analysis

To determine the saturation point and sustainable throughput, we conducted experiments with increasing client counts (ranging from 15 to 80 clients) and measured throughput (operations per second) and latency (median and 95th percentile) for both GET and PUT operations. The saturation point is identified as the client count beyond which increasing the number of clients does not result in a significant increase in throughput. This indicates that the system has reached its maximum capacity.

Once the saturation point was identified, we used the throughput and latency measurements at that client count to compare the performance of ABD and Blocking protocols. We evaluated two workload patterns:

- 90% GETs, 10% PUTs: Read-heavy workload typical of many real-world applications
- 10% GETs, 90% PUTs: Write-heavy workload that stresses the write path

Each experiment was run for 60 seconds. We tested on three different server configurations: 1 server, 3 servers, and 5 servers.

2.2 Results: 90% Gets, 10% Puts (Read-Heavy Workload):

The following tables show the throughput and latency measurements as the number of clients increases. The saturation point is identified when throughput flattens or decreases with additional clients.

1 server configuration:

Clients	ABD Throughput (ops/sec)	Blocking Throughput (ops/sec)	ABD GET Latency (median/95th)	ABD PUT Latency (median/95th)	Blocking GET Latency (median/95th)	Blocking PUT Latency (median/95th)
20	3526.27	2062.38	4302 / 11481	3366 / 10494	6777 / 12609	6674 / 12467
30	5591.90	4946.63	4285 / 11481	3330 / 10540	5310 / 6589	5283 / 6486
40	6415.55	6250.52	4604 / 11743	3559 / 10711	5990 / 9193	5894 / 8973
50	6797.56	7082.02	5067 / 12579	3883 / 11358	6507 / 9283	6401 / 9081
60	8179.87	8483.65	5534 / 13019	4180 / 11620	6907 / 8288	6818 / 8126
70	8801.83	8218.08	6880 / 13836	5156 / 12047	7423 / 10675	7334 / 10471
80	8480.12	9583.43	8807 / 15155	6628 / 12965	8054 / 10498	7960 / 10375

3 Server Configuration:

Clients	ABD Throughput (ops/sec)	Blocking Throughput (ops/sec)	ABD GET Latency (median/95th)	ABD PUT Latency (median/95th)	Blocking GET Latency (median/95th)	Blocking PUT Latency (median/95th)
20	2514.65	1606.15	7200 / 13066	5059 / 8300	9379 / 33048	9414 / 33607
30	3876.42	2167.11	7236 / 12723	4984 / 8744	9591 / 13129	9664 / 13191
40	3485.07	3423.30	11569 / 17151	6816 / 11694	8353 / 12680	8352 / 13093
50	3528.55	2404.53	13602 / 19733	9108 / 15317	14193 / 21105	14177 / 21127
60	3614.62	4048.83	16858 / 23205	11839 / 17763	13867 / 20260	13769 / 20162
70	3416.12	301.32	20258 / 26639	15375 / 21479	17425 / 25086	17290 / 24939
80	3429.33	2404.53	23932 / 30832	18436 / 25324	16425 / 21086	17325 / 22086

5 Server Configuration:

Clients	ABD Throughput (ops/sec)	Blocking Throughput (ops/sec)	ABD GET Latency (median/95th)	ABD PUT Latency (median/95th)	Blocking GET Latency (median/95th)	Blocking PUT Latency (median/95th)
20	2041.37	1881.85	7998 / 17232	6152 / 15459	8035 / 10759	8046 / 10947
30	2616.18	2743.47	10006 / 18769	7277 / 16181	10288 / 15025	10297 / 15021
40	2644.38	2622.97	14367 / 23982	10966 / 19812	13665 / 20908	13655 / 21030
50	2295.73	2520.00	20507 / 34496	14899 / 24185	19293 / 26291	19297 / 26287
60	2306.20	2411.02	24287 / 36113	17747 / 27500	24503 / 32963	24496 / 33031
70	2076.40	1730.57	31136 / 46605	21954 / 43500	30703 / 117464	30633 / 116780
80	2029.27	1036.62	37277 / 53917	28869 / 52468	62576 / 159630	62436 / 164302

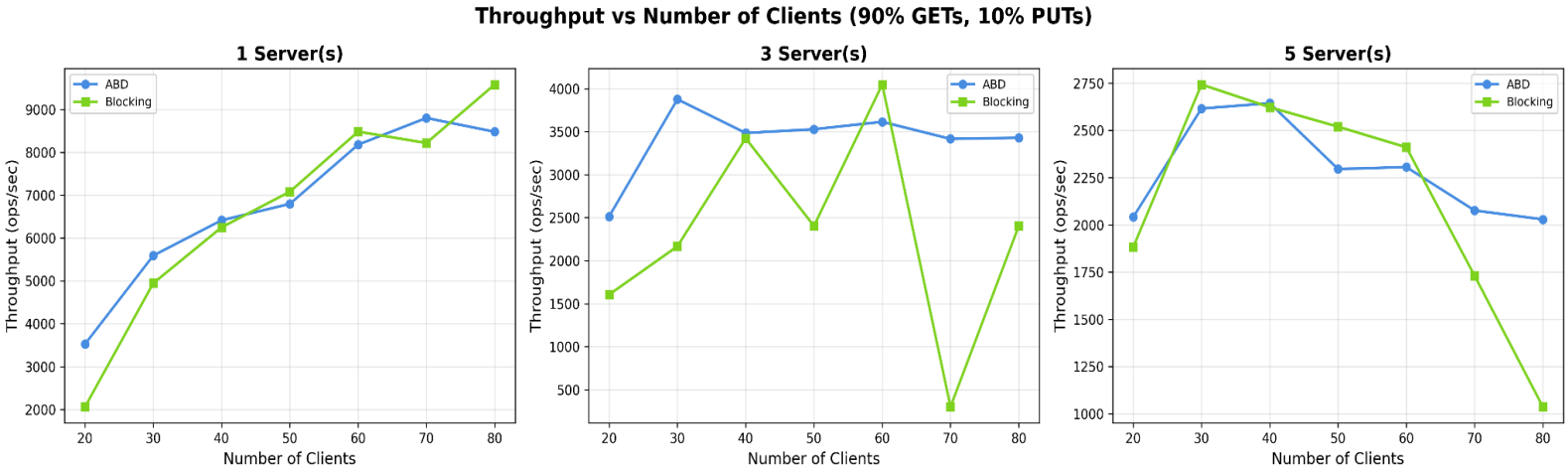
Saturation point (90% GETs):

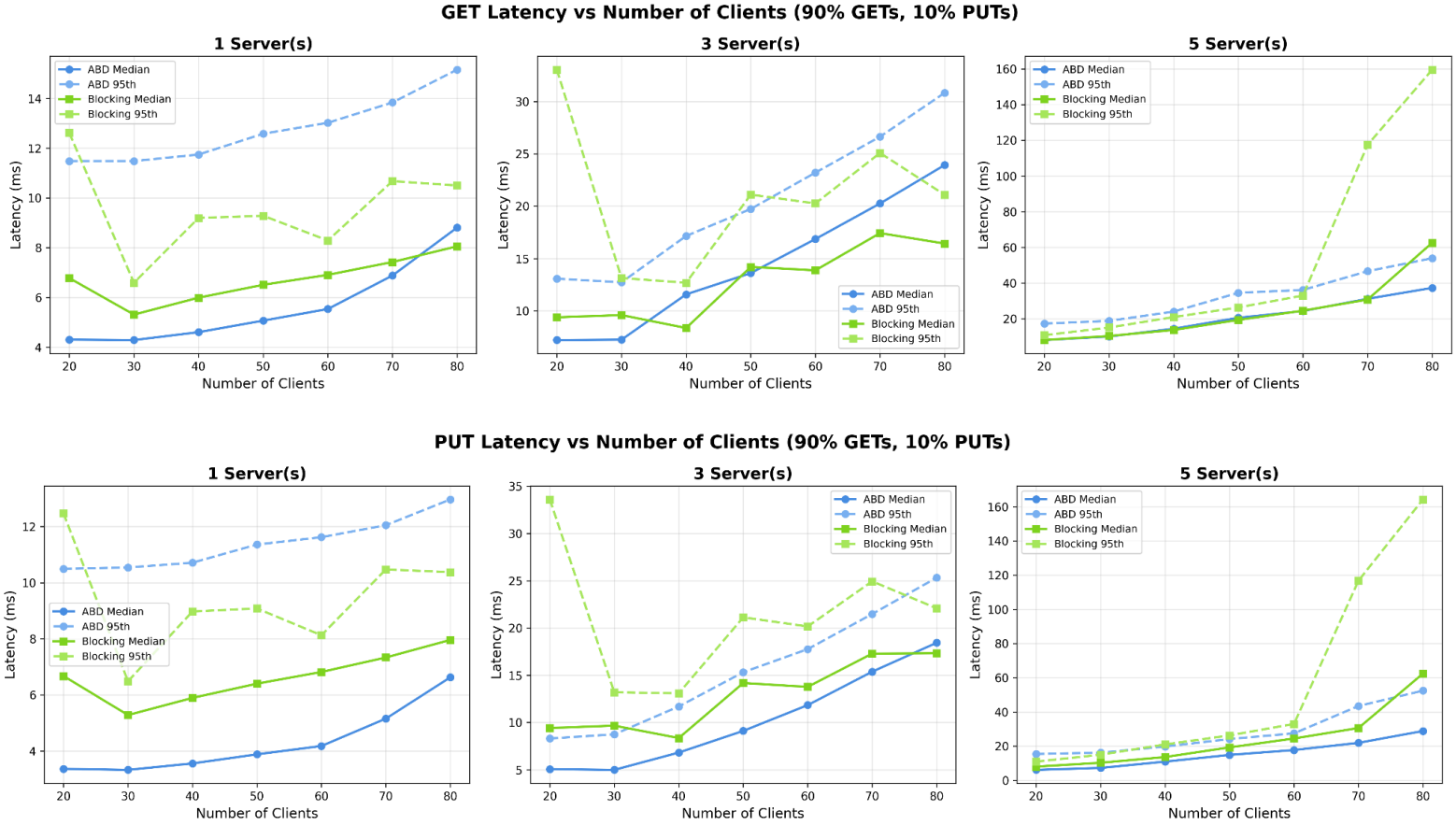
Configuration	ABD Saturation Point	ABD Saturated Throughput	Blocking Saturation Point	Blocking Saturated Throughput
1 Server	70	8801.83	60	8483.65
3 Servers	30	3876.42	40	3423.30
5 Servers	30	2616.18	30	2743.47

Performance comparison at Saturation Point:

Configuration	Protocol	Throughput (ops/sec)	GET Median Latency (µs)	GET 95th Latency (µs)	PUT Median Latency (µs)	PUT 95th Latency (µs)
1 Server	ABD	8801.83	6880	13836	5156	12047
1 Server	Blocking	8483.65	6907	8288	6818	8126
3 Servers	ABD	3876.42	7236	12723	4984	8744
3 Servers	Blocking	3423.30	8353	12680	8352	13093
5 Servers	ABD	2616.18	10006	18769	7277	16181
5 Servers	Blocking	2743.47	10288	15025	10297	15021

Below are the lots for the data mentioned in the above tables:





2.3 Results: 10% Gets, 90% Puts (Write-Heavy Workload):

1 Server Configuration:

Clients	ABD Throughput (ops/sec)	Blocking Throughput (ops/sec)	ABD GET Latency (median/95th)	ABD PUT Latency (median/95th)	Blocking GET Latency (median/95th)	Blocking PUT Latency (median/95th)
20	3516.23	3194.57	5934 / 7409	5025 / 6490	5324 / 10123	5310 / 9849
30	7889.57	3794.17	4083 / 6449	3261 / 5353	6768 / 13154	6657 / 12990
40	9927.07	4092.35	4421 / 5896	3457 / 4839	8328 / 22638	8217 / 22320
50	12091.00	3598.15	4846 / 6384	3764 / 5039	8633 / 30653	8526 / 30525
60	11992.70	5616.63	5652 / 8250	4346 / 6632	10125 / 14484	10017 / 14465
70	11764.80	6287.53	6951 / 10755	5316 / 8873	10859 / 14449	10739 / 14400

80	10696.90	6694.02	8767 / 13190	6716 / 11040	11024 / 15561	10915 / 15584

3 Servers Configuration:

Clients	ABD Throughput (ops/sec)	Blocking Throughput (ops/sec)	ABD GET Latency (median/95th)	ABD PUT Latency (median/95th)	Blocking GET Latency (median/95th)	Blocking PUT Latency (median/95th)
20	2492.20	2171.37	7344 / 19840	5800 / 15055	6873 / 13546	6861 / 13502
30	3695.75	1335.72	8160 / 18051	6191 / 14581	11705 / 80609	11773 / 81550
40	4475.68	3266.30	8709 / 20349	6431 / 16258	9024 / 28849	8998 / 28156
50	4516.22	4457.80	11330 / 21198	8303 / 17956	10442 / 15935	10434 / 15907
60	4259.03	3876.10	16398 / 25475	12763 / 21407	13397 / 21122	13314 / 21087
70	3950.97	4110.42	21160 / 32860	16420 / 25629	16783 / 23286	16767 / 23239
80	3842.17	2651.88	25098 / 35923	19344 / 28285	21053 / 99056	21015 / 101102

5 Servers configuration:

Clients	ABD Throughput (ops/sec)	Blocking Throughput (ops/sec)	ABD GET Latency (median/95th)	ABD PUT Latency (median/95th)	Blocking GET Latency (median/95th)	Blocking PUT Latency (median/95th)
20	2346.80	917.73	7962 / 17255	6058 / 15083	8751 / 81752	8682 / 82414
30	2998.90	2251.50	10288 / 19753	7519 / 16408	10394 / 16852	10399 / 17142
40	2627.61	2368.52	16946 / 26907	13054 / 21627	13627 / 23429	13715 / 24125
50	2559.10	2502.55	24221 / 35623	17938 / 26925	19385 / 27212	19405 / 27127
60	2449.10	2299.07	22321 / 33423	17731 / 27925	24536 / 35391	24620 / 36076
70	2465.00	2269.60	38205 / 52733	26383 / 35456	29491 / 38863	29522 / 38921
80	2331.82	2302.82	44534 / 60149	30384 / 40446	34635 / 45563	34528 / 45525

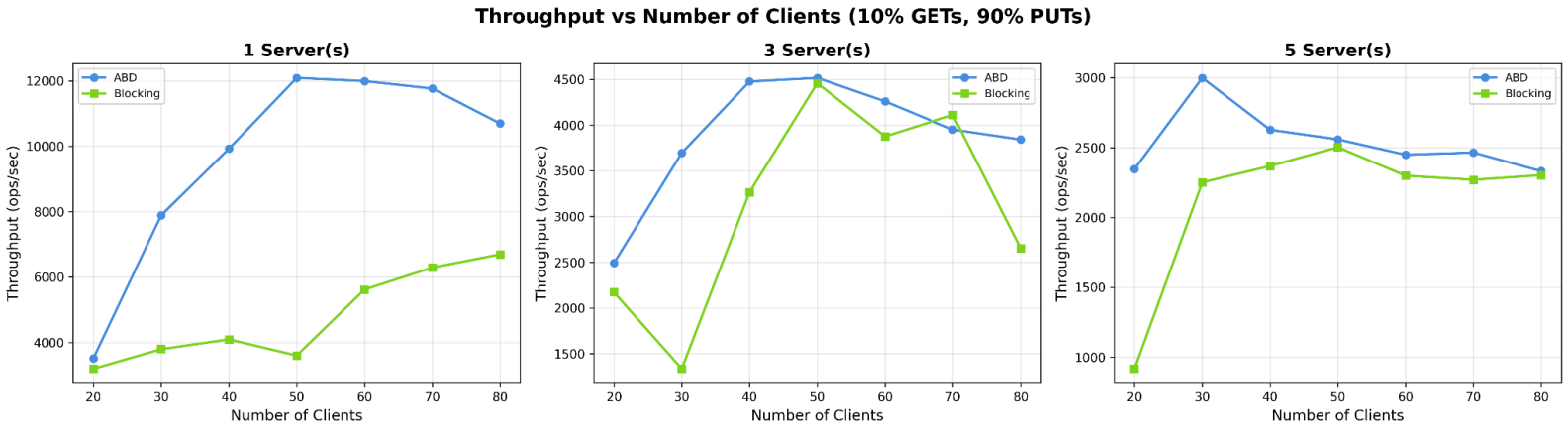
Saturation Point (90% PUTs):

Configuration	ABD Saturation Point	ABD Saturated Throughput	Blocking Saturation Point	Blocking Saturated Throughput
1 Server	50	12091.00	40	4092.35
3 Servers	40	4475.68	20	2171.37
5 Servers	30	2998.90	50	2502.55

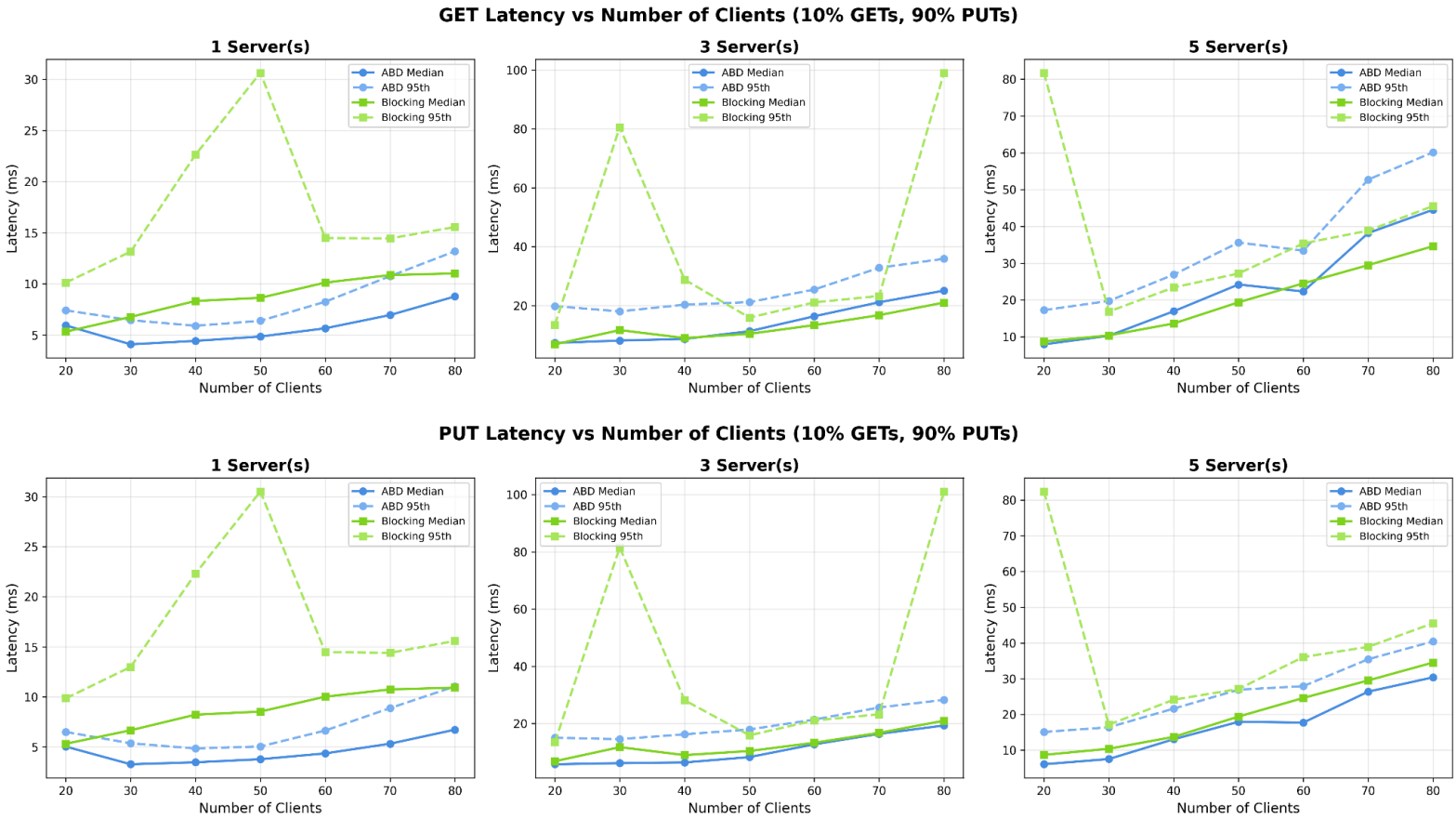
Performance Comparison at Saturation Point:

Configuration	Protocol	Throughput (ops/sec)	GET Median Latency (µs)	GET 95th Latency (µs)	PUT Median Latency (µs)	PUT 95th Latency (µs)
1 Server	ABD	12091.00	4846	6384	3764	5039
1 Server	Blocking	4092.35	8328	22638	8217	22320
3 Servers	ABD	4475.68	8709	20349	6431	16258
3 Servers	Blocking	2171.37	6873	13546	6861	13502
5 Servers	ABD	2998.90	10288	19753	7519	16408
5 Servers	Blocking	2502.55	19385	27212	19405	27127

Below are the figures based on the date from tables:



If we observe some of the plots above, throughput increases with clients monotonically. But the saturation point we considered was somewhere in between. The reason for this is that, even



though the throughput is increased, the failure rate is also increased along with the clients. So, we choose a saturation point that is the culmination of the throughput and success calls count to the server.

3. Crash Impact Evaluation:

To evaluate the impact of client crashes on system performance, we conducted experiments where one client crashes after a specified duration (20 seconds) while other clients continue operating. This simulates a realistic failure scenario where a client process terminates unexpectedly. We measure:

- Throughput before crash: Operations per second before the crash occurs
- Throughput after crash: Operations per second after the crash
- Latency before crash: Median and 95th percentile latencies before the crash
- Latency after crash: Median and 95th percentile latencies after the crash
- Failed operations: Number of operations that failed due to the crash

The experiments were conducted with multiple clients (30-70 clients) where one client crashes after 20 seconds, and the total test duration is 60 seconds.

ABD Protocol Crash Impact Results:

Configuration	Clients	Throughput Before Crash (ops/sec)	Throughput After Crash (ops/sec)	Latency Before (median/ 95th)	Latency After (median /95th)	Failed Ops (before/after)	Throughput Change	Latency Change
1 Server	70	4522.25	4344.15	13260 / 21289	13198 / 24392	0 / 0	-3.94%	-0.47%
3 Servers	30	2659.15	2486.55	11517 / 16169	11352 / 17750	0 / 0	-6.49%	-1.43%
5 Servers	40	1751.90	1417.22	22327 / 36414	23509 / 47957	0 / 3	-19.10%	+5.29%

Blocking Protocol Crash Impact Results:

Configuration	Clients	Throughput Before Crash (ops/sec)	Throughput After Crash (ops/sec)	Latency Before (median /95th)	Latency After (median/ 95th)	Failed Ops (before/after)	Throughput Change	Latency Change
1 Server	60	4325.10	4064.93	12862 / 18480	13684 / 19786	0 / 1	-6.02%	+6.39%
3 Servers	30	1897.05	1872.85	15058 / 19654	13477 / 20419	0 / 0	-1.28%	-10.50%
5 Servers	40	1870.05	1750.78	20319 / 26902	20215 / 33119	0 / 0	-6.38%	-0.51%

The crash impact evaluation reveals important insights into the resilience and recovery behavior of both protocols

Key Observations:

- Failure Resilience: Both protocols generally handle client crashes gracefully with minimal failed operations (0-3 failures across all configurations).
- Throughput Impact: Throughput decreases range from 1.28% to 19.10%, with ABD showing more variability across server configurations.
- Latency Behavior: Latency changes are mixed, with some configurations showing improvement (likely due to reduced contention) and others showing degradation.

- Protocol Comparison: The blocking protocol shows more consistent behavior across configurations, while ABD's performance varies more significantly with server count.
- Server Count Effect: Higher server counts (5 servers) tend to show larger performance impacts, particularly for ABD, suggesting increased complexity in distributed crash recovery.

Summary:

This comprehensive evaluation of the ABD and Blocking protocols for distributed key-value storage reveals distinct performance characteristics and trade-offs across correctness, performance, and fault tolerance dimensions.

Both protocols successfully maintain linearizability and consistency across all test scenarios. The ABD protocol's wait-free design ensures non-blocking operations, while the Blocking protocol's lock-based approach provides mutual exclusion guarantees. All correctness tests passed across 1, 3, and 5 server configurations, demonstrating robust implementation of both protocols.

Throughput Analysis:

- Read-Heavy Workload (90% GETs): ABD generally achieves higher throughput at saturation for single-server configurations (8801.83 vs 8483.65 ops/sec), while Blocking performs better in 5-server configurations (2743.47 vs 2616.18 ops/sec). The optimal protocol choice depends on deployment scale.
- Write-Heavy Workload (90% PUTs): ABD significantly outperforms Blocking in write-heavy scenarios, achieving 2-3x higher throughput in single-server configurations (12091.00 vs 4092.35 ops/sec). This advantage diminishes but remains present in distributed configurations.

Latency Characteristics:

- GET Operations: Blocking protocol shows lower 95th percentile latencies in read-heavy workloads, particularly in single-server configurations (8288 vs 13836 μs), suggesting better tail latency characteristics for reads.
- PUT Operations: ABD demonstrates superior write latency, especially in write-heavy workloads, with median latencies up to 2x lower than Blocking in some configurations.

Server Scaling:

- Throughput generally decreases with increasing server count for both protocols, reflecting the overhead of distributed coordination.
- ABD shows more pronounced throughput degradation in distributed settings, while Blocking maintains more consistent relative performance.

Fault Tolerance:

Both protocols demonstrate strong resilience to client crashes:

- Failure Rate: Extremely low failure rates (0-3 operations) across all crash scenarios, indicating robust error handling.
- Recovery Behavior: Throughput impacts range from 1.28% to 19.10%, with most configurations showing moderate degradation (3-7%).
- Protocol Comparison: Blocking protocol shows more consistent crash recovery behavior across configurations, while ABD's impact varies more significantly with server count.

Github link:
https://github.com/sbanda99/distributed-kvstore/tree/main