

web*performance*

---

# Client Sample Engineer's Report

Michael Czeiszperger

March 1, 2026

## Executive Summary

The system under test was <http://beta.client.com>, a SaaS reservation platform. The test simulated four distinct company reservation configurations running concurrently, ramping from 100 to 1,500 virtual users over approximately one hour.

The system delivered strong performance up to approximately 1,100 concurrent users, at which point throughput plateaued and response times began to degrade. Peak reservation processing reached 681.8 cases per minute with 1,100 users, up from approximately 600 cases/min in the 2024/2025 testing cycle. This improvement occurred despite the capacity ceiling dropping from 2,000 to approximately 1,100 concurrent users and is attributable to the frontend redesign, which streamlined the reservation workflow, reduced form steps, and lowered think times between actions.

The primary bottleneck remains database write operations during reservation confirmation. At high load levels, the Confirm Reservation equivalent pages exhibited response times exceeding 25 seconds, almost certainly due to database locking contention. Addressing this bottleneck through lock duration reduction, resource sharding, or optimistic locking represents the highest-impact opportunity for future performance gains.

### Key Performance Metrics

Metric	2025 Test	2026 Test
Peak Concurrent Users (Optimal)	1,200+	900
Capacity Ceiling	>1,200 users	~1,100 users
Peak Throughput (Hits/sec)	~250	656
Peak Reservations/min	~600	681.8
Avg Page Duration (at capacity)	<1.1 sec	2.6 sec
Error Rate (at capacity)	0%	0%
Peak Bandwidth	~120 Mbps	~340 Mbps

Important: The 2025 and 2026 concurrent user numbers are not directly comparable due to the reduced think times in the 2026 test. The key business metric—reservations processed per minute—increased by approximately 13%, representing a genuine capacity improvement on the same hardware.

### 2026 Performance Thresholds

Optimal (0–900 users)	Moderate (900–1,100)	High Load (1,100–1,300)	Critical (1,300+)
Consistent sub-2s response times	Gradual increase to 2–3 seconds	Response times 3–4 seconds	Exceeded 4s threshold (4.7s at 1,300)

## Test Configuration

---

The test was configured as a stepped ramp from 100 to 2,000 virtual users, adding 200 users per ramp, with a 5-minute hold duration and 1-minute ramp duration at each level. The total test duration was approximately 2 hours. This configuration was designed to complete testing within a compressed timeframe while maintaining accuracy comparable to previous years.

The test design encompassed four distinct company reservation configurations to simulate realistic traffic diversity. While the client decided to limit testing to four configurations (down from a larger potential set) to reduce costs, this provides a reasonable sample size for identifying performance patterns.

### Test Case Distribution

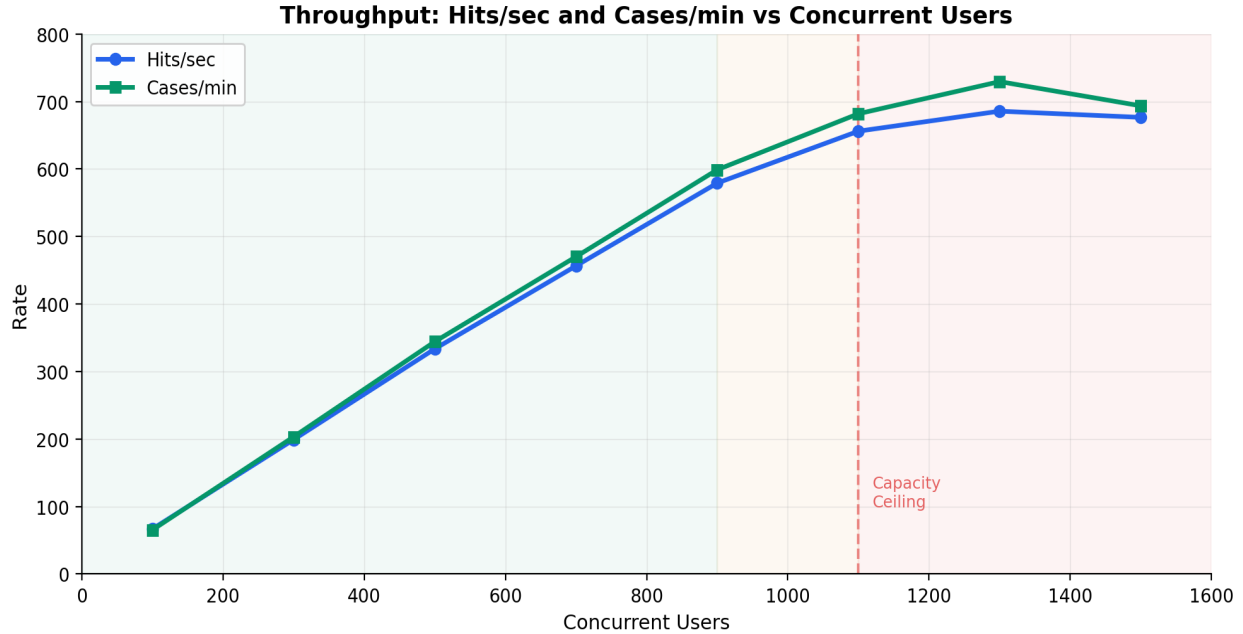
The test included 15 test cases across five workflow categories: CSR operations (update building, add device, edit device), end-user device management for organizations A through D, account creation for organizations A through D, and guest access for organizations A through D. Each test case was weighted to simulate realistic traffic distribution.

# Throughput Analysis

Throughput is the system's ability to process requests as load increases. Ideally, throughput scales linearly: adding X users yields a proportional increase in hits per second. When throughput flattens, the system has reached its processing capacity, and additional users will experience slower response times.

## Throughput by Load Level

Users	Hits/sec	Cases/min	Pages/sec	Avg Dur	Max Dur	Std Dev
100	66.7	65.2	6.1	1,547ms	6,411ms	1,267ms
300	199.0	203.2	18.3	1,563ms	7,212ms	1,276ms
500	333.5	344.2	30.7	1,581ms	6,723ms	1,280ms
700	456.4	470.0	42.2	1,625ms	7,969ms	1,339ms
900	579.1	598.8	53.6	1,827ms	76,080ms	1,682ms
1,100	656.0	681.8	60.7	2,609ms	26,868ms	2,779ms
1,300	685.6	729.4	64.3	4,680ms	75,518ms	5,179ms
1,500	676.4	693.5	62.7	4,628ms	79,231ms	6,137ms



Key observations from the throughput data:

**Linear scaling through 900 users.** Hits/sec scaled nearly linearly from 67 to 579, and cases/min grew proportionally from 65 to 599. Average page duration remained remarkably stable at 1.5–1.8 seconds through this range.

**Throughput plateau at 1,100 users.** Hits/sec flattened abruptly at 656, and cases/min peaked at 682. Average response time jumped to 2.6 seconds—a 43% increase over the 900-user level. This inflection point marks the system's effective capacity ceiling.

**Diminishing returns beyond 1,100 users.** Adding 200 more users to reach 1,300 yielded only a 4.5% throughput increase (656 → 686 hits/sec) but a 79% increase in average response time (2.6s → 4.7s). At 1,500 users, throughput actually declined slightly to 676 hits/sec while errors began to appear.

**Response time variability is a key concern.** Standard deviation nearly quintupled from 1,267ms at 100 users to 6,137ms at 1,500 users. This means user experience becomes highly unpredictable under heavy load—some users see sub-2-second responses while others wait over a minute.

***A note on year-over-year comparisons:** The 2026 hits/sec metric is substantially higher than previous years because the think time between reservation steps was reduced to reflect the streamlined frontend. This means users complete reservations faster, generating more server activity per concurrent user. Direct comparison of hits/sec or concurrent user counts across years is misleading—the key comparable metric is reservations per minute, which increased ~13%.*

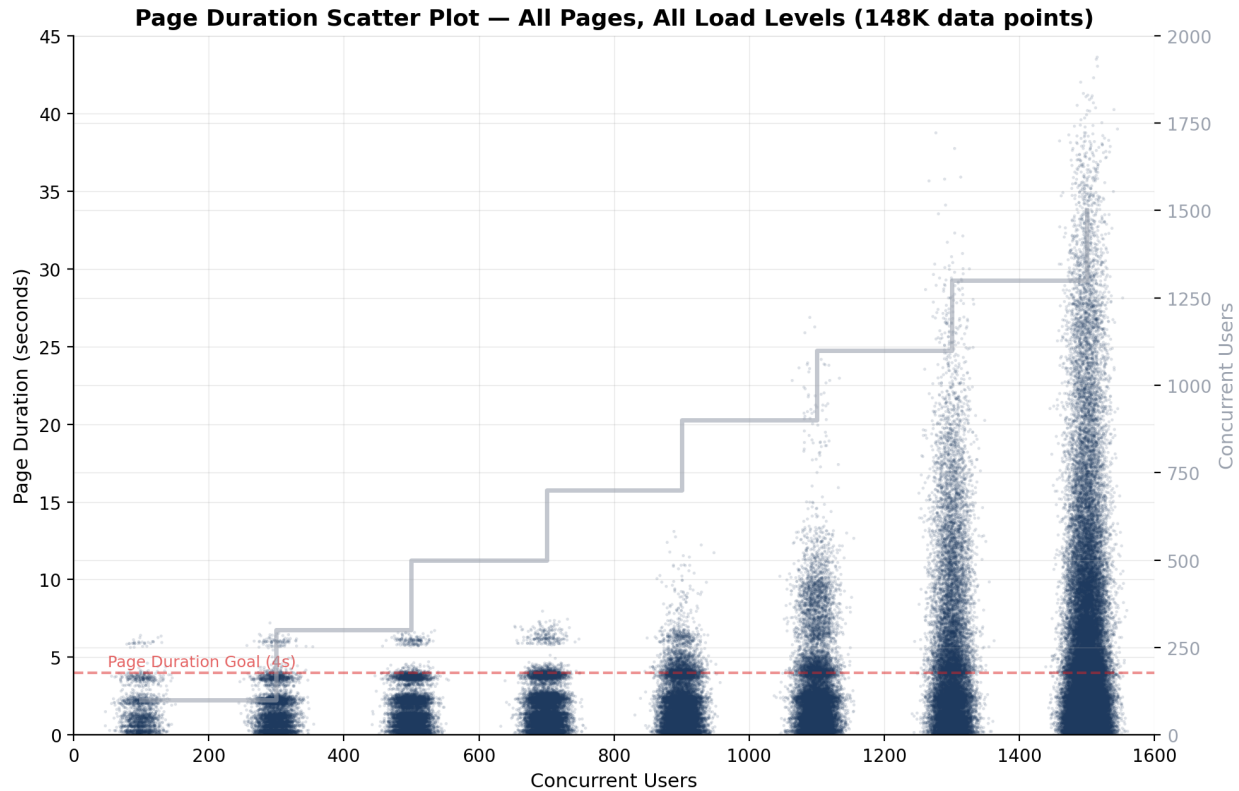
To improve the accuracy of future tests, it is worth collecting real user metrics (for example, from Google Analytics) to calibrate think times rather than estimating them. This would make year-over-year comparisons significantly more reliable.

## Page Performance Analysis

Average page performance was comparable to last year through the 700-user level, at which point average load time began to climb. By 1,300 users, the average page duration reached 4.7 seconds, exceeding the 4-second performance goal. This degradation reflects the increased system throughput (more requests per user due to reduced think times) rather than a regression in server processing capability.

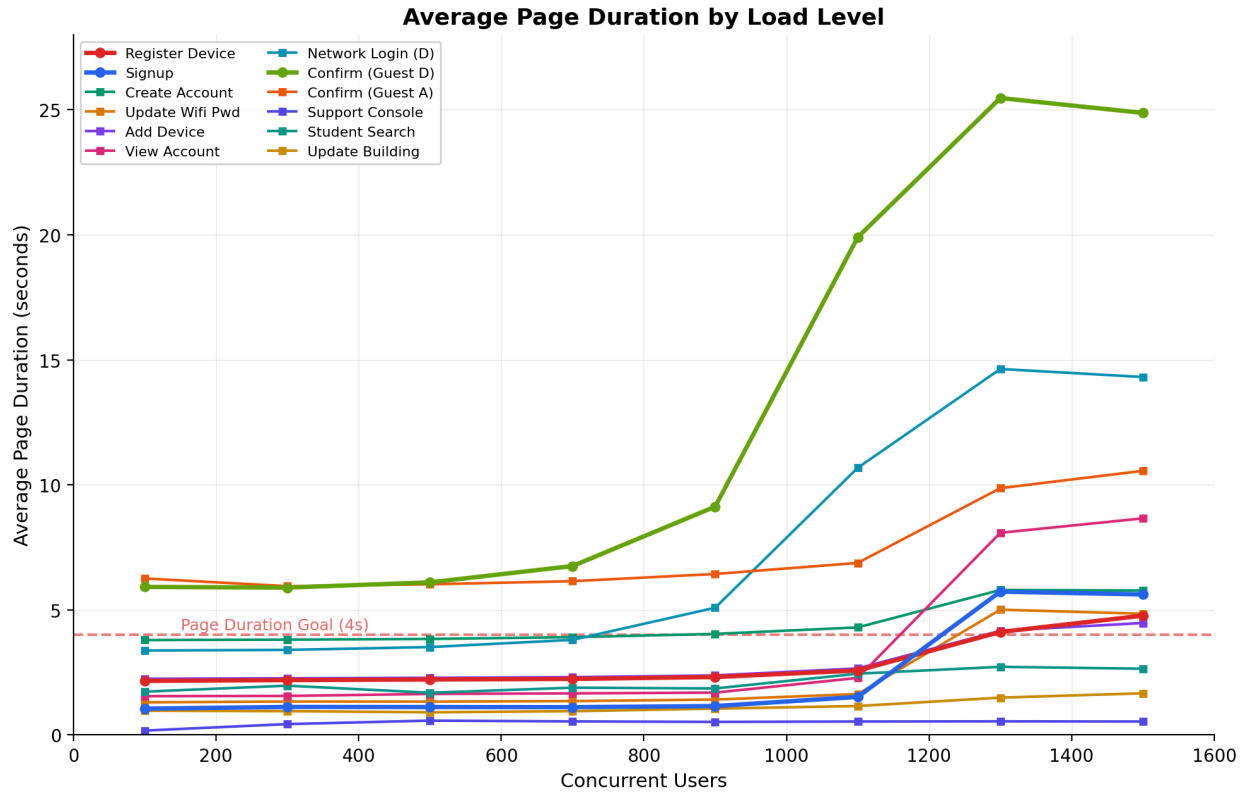
### Response Time Distribution

The scatterplot data from all four years of testing show that the 2026 test maintained tight response times up to approximately 700 concurrent users. Beyond that level, the upper range of response times expanded dramatically, reaching 10 seconds, then 20, and eventually peaking at approximately 79 seconds for the slowest individual pages. This widening distribution is particularly important: even when the average remains under 5 seconds, a significant number of individual users experienced response times of 20–40 seconds.



### Average Duration by Page and Load Level

The chart below isolates the average response time for key pages across all load levels. Pages that involve database writes (Confirm, Register Device) show the steepest degradation curves, while read-only pages (Support Console, Student Search) remain relatively flat.



### Slowest Pages at 1,300 Concurrent Users

Page	Avg Dur	Max Dur	Status	Test Case
Confirm/Register (Guest D equiv.)	25,466ms	38,744ms	Critical	www-guest-D
Network Login (Create D)	14,633ms	30,982ms	Slow	www-create-D
Network Login (Device B)	11,903ms	30,008ms	Slow	www-device-B
Register Device	10,771ms	27,415ms	Slow	www-create-A
Guest Confirmation (B)	10,485ms	27,852ms	Slow	www-guest-B
Device Config (D)	10,335ms	28,660ms	Slow	www-device-D
Guest Confirmation (A)	9,867ms	26,758ms	Slow	www-guest-A
Guest Confirmation (C)	9,715ms	25,060ms	Warning	www-guest-C
View Account	8,083ms	28,986ms	Warning	www-device-B
Network Portal (Create C)	8,121ms	29,077ms	Warning	www-create-C

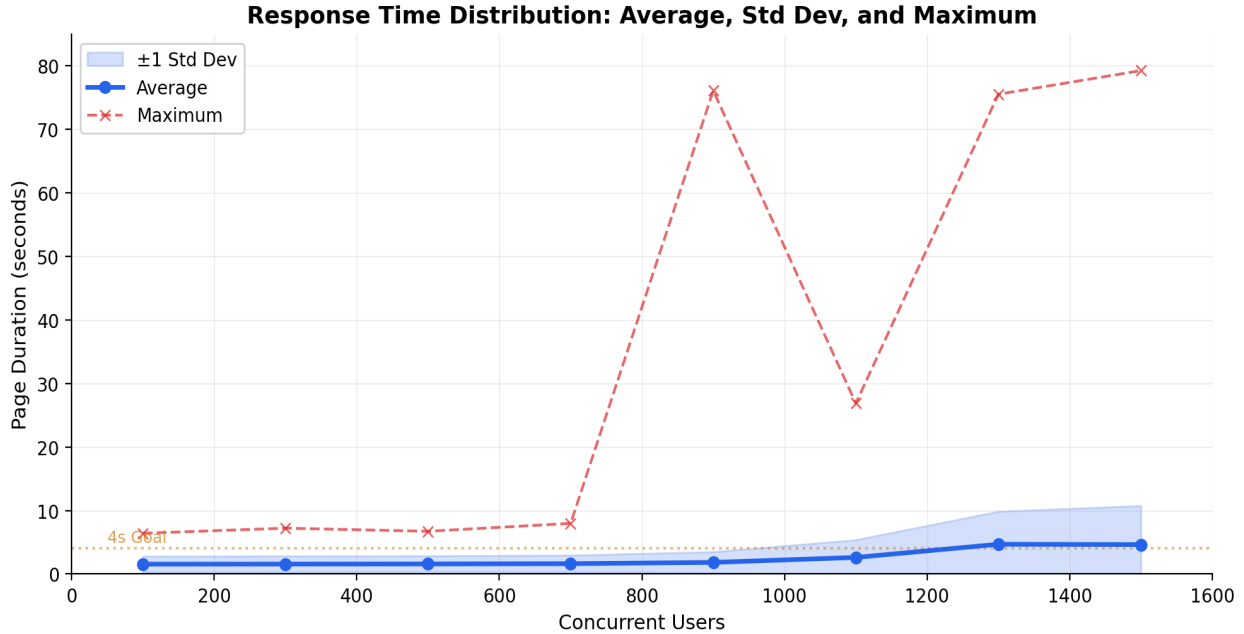
The Confirm/Register page variants (database write operations) are consistently the worst performers, with the Guest D registration flow averaging over 25 seconds at 1,300 users. This pattern strongly points to database contention as the root cause, consistent with previous years' findings.

# Database Bottleneck Analysis

The reservation confirmation step, the database write that completes each reservation, has been the system's primary bottleneck in every year of testing. In 2026, this pattern is even more pronounced because the streamlined frontend delivers more reservations per minute to the same database infrastructure.

## Register Device Response Time Progression

Users	Avg Dur	Max Dur	Std Dev	Δ Avg	Status
100	2,162ms	2,675ms	149ms	—	OK
300	2,194ms	2,692ms	133ms	+1.5%	OK
500	2,217ms	2,760ms	142ms	+1.0%	OK
700	2,241ms	2,882ms	135ms	+1.1%	OK
900	2,316ms	3,820ms	197ms	+3.3%	OK
1,100	2,577ms	5,733ms	489ms	+11.3%	Warn
1,300	4,117ms	25,777ms	3,427ms	+59.8%	Slow
1,500	4,770ms	36,429ms	5,242ms	+15.9%	Critical



The data tells a clear story: the Register Device page scales linearly with under 3% variation from 100 to 700 users, then begins to degrade at 900 (standard deviation increases 46%), hits a significant inflection at 1,100 (average jumps 11%, max response time doubles), and falls off sharply at 1,300 where the average nearly doubles and the maximum reaches 26 seconds.

## Remediation Recommendations (Prioritized)

- 1. Optimize database locking strategy (High impact, moderate effort).** The current write pattern likely uses pessimistic locking, which creates contention as concurrent writes increase. Switching to optimistic locking with retry logic would allow the database to process more concurrent writes. This is the highest-ROI change because it addresses the root cause directly.
- 2. Reduce lock duration (High impact, low effort).** Review the transaction scope around reservation writes. If business logic (validation, notifications, etc.) runs inside the same transaction as the database write, moving non-critical operations outside the transaction would reduce the time each write holds locks.
- 3. Database resource sharding (High impact, high effort).** Partitioning reservation data across multiple database instances or table partitions would distribute write contention. This is the most architecturally significant change and would provide the largest capacity increase, but requires more planning and testing.
- 4. Scale database instance (Moderate impact, low effort).** Increasing the database instance size (CPU, memory, IOPS) would provide immediate relief but has diminishing returns and does not address the fundamental locking contention. Consider this as a short-term bridge while implementing the above optimizations.

## Test Case Performance Breakdown

The table below shows each test case's performance at the 1,100-user capacity ceiling. Durations represent the full workflow completion time (all pages in the test case), not individual page loads.

Test Case	Completions	Avg Duration	Max Duration	Failures
csr-update-building	55	1m 48s	2m 3s	0
csr-add-device	65	2m 14s	2m 40s	0
csr-edit-device	84	1m 8s	1m 25s	0
www-device-A	247	2m 3s	2m 26s	0
www-device-B	242	2m 5s	2m 29s	0
www-device-C	299	1m 42s	2m 2s	0
www-device-D	535	52s	1m 2s	0
www-create-A	454	1m 18s	1m 36s	0
www-create-B	366	1m 41s	2m 6s	0
www-create-C	374	1m 41s	2m 2s	0
www-create-D	396	1m 30s	1m 51s	0
www-guest-A	36	1m 16s	1m 32s	0
www-guest-B	39	1m 16s	1m 30s	0
www-guest-C	142	13s	18s	0
www-guest-D	75	33s	40s	0

**Notable patterns:** Organization D workflows (www-device-D, www-guest-D, www-create-D) consistently complete faster than their A/B/C counterparts. The www-device-D test case completed 535 iterations at an average of 52 seconds, compared to 247 completions at 2m 3s for www-device-A. This disparity warrants investigation. It may indicate differences in organization D's reservation configuration complexity, database query patterns, or data volume. The csr-add-device workflow is the slowest CSR operation at 2m 14s average, suggesting the device addition process involves heavier server-side processing than updates or edits.

## Error Analysis

The system ran cleanly through 900 concurrent users with zero page failures. Errors began appearing at 1,300 users and increased significantly at 1,500. To properly assess system reliability, these errors must be categorized by cause.

### Error Summary by Load Level

Users	Failed Pages	Failure Rate	Retries	JWT Failures	Data Exhausted	Abandoned
100–900	0	0.000%	Minor	0	0	0
1,100	0	0.000%	7	0	0	0
1,300	5	0.03%	6	5	0	1
1,500	537	0.82%	21	37	500+	273

### Error Categories Explained

**Benign: Connection retries.** Messages like “Retried a request after 0 seconds” indicate the load balancer or server briefly dropped a connection, and the request was automatically retried. These are transparent to real users and are normal under load. They appeared at every level in small numbers.

**Server-side: JWT bearer token failures.** At 1,300 users, 5 requests failed to extract a JWT token from the server response. This means the server returned an unexpected response (likely an error page or timeout) instead of the expected authentication token. This is a genuine indicator of server stress and confirms the capacity ceiling.

**Test infrastructure: Dataset exhaustion.** At 1,500 users, over 500 errors occurred when the test ran out of unique email addresses in the test dataset. This is a test harness limitation, not a server problem. It inflated the 1,500-user failure count and likely depressed throughput at that level. Future tests should provision a larger dataset to ensure the test infrastructure does not become the bottleneck before the server does.

**Cascading failures: Abandoned transactions.** At 1,500 users, 273 transactions were abandoned—meaning virtual users could not complete their workflows. This is a combination of server-side stress (JWT failures preventing subsequent steps) and dataset exhaustion (no valid email addresses available to continue).

**Bottom line:** The system ran error-free through 1,100 users. The errors at 1,300 users (5 JWT failures out of 19,281 pages = 0.03%) confirm the capacity ceiling but are not alarming. The 1,500-user error count is significantly inflated by test data exhaustion and should not be interpreted as a server reliability issue at that level.

## Server Response Time (TTFB)

---

Time to First Byte (TTFB) measures the time between sending a request and receiving the first byte of the response. This isolates server processing time from network transfer and page rendering, making it a valuable diagnostic metric for identifying whether bottlenecks are in server computation or payload delivery.

Users	Avg TTFB	Max TTFB	Change
100	61ms	64ms	—
700	218ms	338ms	+257%
1,100	215ms	345ms	Stable
1,300	220ms	429ms	+2.3%

For the Support Console page, TTFB increased from 61ms at 100 users to approximately 218ms at 700+ users, then remained essentially flat through 1,300 users. This is significant because it tells us that the initial server response time is not the primary source of degradation—the server begins processing requests at roughly the same speed regardless of load level. The slowdowns observed in total page duration are primarily occurring in the database write phase and in page payload delivery under congestion.

## User Queuing Behavior

---

The test data captures the number of virtual users in a “waiting” state (queued for processing) and their average wait time at each load level. This metric directly correlates with real-world user experience: waiting users represent people staring at a loading spinner.

Users	Waiting Users	% Waiting	Avg Wait
100	11	11.0%	1.3s
300	35	11.7%	1.7s
500	41	8.2%	1.1s
700	71	10.1%	1.6s
900	79	8.8%	1.3s
1,100	208	18.9%	4.1s
1,300	204	15.7%	5.8s
1,500	129	12.9%*	2.6s*

\* The 1,500-user level shows lower waiting metrics because many users had already been abandoned due to dataset exhaustion and were no longer actively queuing.

The jump from 79 waiting users (8.8%) at 900 users to 208 waiting users (18.9%) at 1,100 users is another confirmation of the capacity ceiling. At 1,300 users, the average wait time of 5.8 seconds means a meaningful fraction of users are experiencing delays before their requests even begin processing.

## Test Limitations and Caveats

---

Every load test involves simplifications and assumptions that should be understood when interpreting results. The following factors may have influenced these results:

**Think time estimation.** Think times (the pauses between user actions that simulate reading and form-filling) were estimated based on the streamlined frontend design. Because real user behavior data was not available, these estimates may not accurately reflect production usage patterns. If real users take longer between steps than simulated, the system would support more concurrent users; if they are faster, the capacity ceiling would be lower. Collecting real user metrics from Google Analytics or similar tools would significantly improve accuracy in future tests.

**Four-organization sample.** Testing was limited to four company reservation configurations. While this is a reasonable sample, it does not capture the full diversity of configurations in production. Organizations with more complex reservation setups may produce different performance profiles.

**Test data exhaustion at 1,500 users.** The test dataset ran out of unique email addresses during the 1,500-user level, generating over 500 errors and 273 abandoned transactions. This artificially depressed throughput metrics and inflated error counts at the highest load level. Results at 1,500 users should be treated as a lower bound of actual system capability at that level. Future tests should provision at least 2x the expected number of unique data records.

**Single test run.** Results represent a single test execution. Server performance can vary due to database cache state, background processes, and other factors. For critical capacity planning decisions, consider running multiple iterations to establish confidence intervals.

**Network conditions.** The test was conducted from a load generation environment with low-latency connectivity to the test server. Real users connecting over the public internet will experience additional latency from network hops, geographic distance, and local network conditions.

## Conclusions and Recommendations

---

The 2026 test demonstrates that the frontend redesign successfully increased reservation processing throughput by approximately 13% (from ~600 to ~682 cases/min) on the same infrastructure. The system performs well within its optimal range of 0–900 concurrent users, with consistent sub-2-second response times and zero errors.

The capacity ceiling at approximately 1,100 concurrent users, while lower than the 2025 test's 1,200+ concurrent user threshold, is not a regression; it reflects the higher per-user throughput generated by the streamlined frontend. The system processes more work with fewer concurrent users.

### Priority Actions

- 1. Address database write contention** (high priority). This has been the bottleneck for four consecutive years of testing and is the single highest-impact improvement available. Focus on reducing lock duration, implementing optimistic locking, and evaluating sharding options.
- 2. Provision larger test datasets** (before next test). The email dataset exhaustion at 1,500 users compromised results at the highest load level. Ensure future tests have sufficient unique test data to reach at least 2,000 users without data-related failures.
- 3. Implement real user monitoring** (medium priority). Deploying analytics to capture actual user think times, session durations, and workflow patterns would dramatically improve test accuracy and make year-over-year comparisons meaningful.
- 4. Investigate organization D performance disparity** (low priority). Organization D workflows consistently outperform A, B, and C by a wide margin. Understanding why may reveal optimization opportunities applicable to all organizations.