Stress & Load Tests

Final Version

Matan Hazan 315198796

Omer Kempner 322217472

Ori Kintzlinger 318929213

Arye Shapiro 313578379

Benny Skidanov 322572926

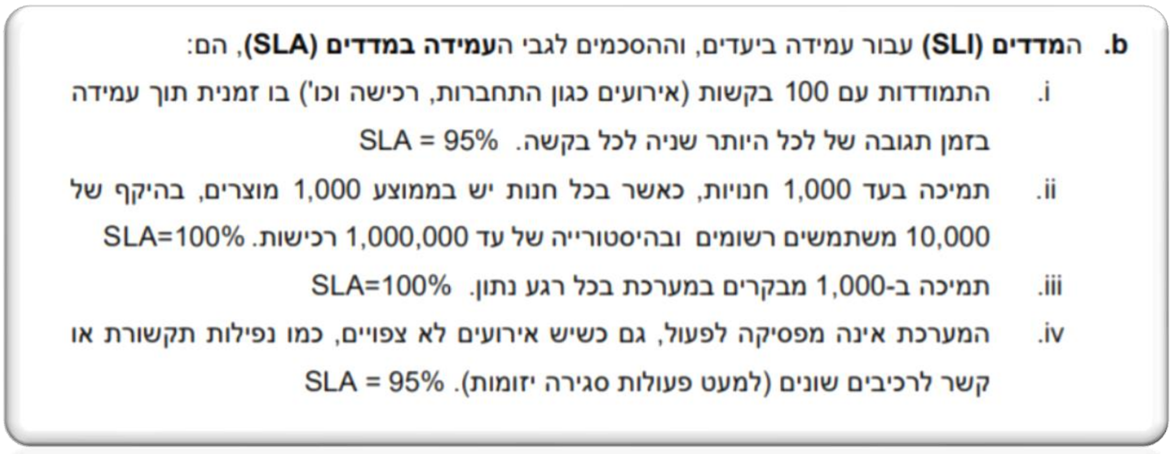
Contents

Introduction

This document purpose is to present and analyze the results of the stress & load tests that were perform on the system to check the performance and durability of our system toward mass of requests.

Formal Requirements

The formal requirements as described in the version specification file :



b. המדדים (SLI) עבור עמידה ביעדים, וההסכמים לגבי העמידה במדדים (SLA), הם:

i. התמודדות עם 100 בקשות (אירועים כגון התחברות, רכישה וכו') בו זמנית תוך עמידה בזמן תגובה של לכל היותר שניה לכל בקשה. SLA = 95%

ii. תמיכה בעד 1,000 חנויות, כאשר בכל חנות יש בממוצע 1,000 מוצרים, בהיקף של 10,000 משתמשים רשומים ובהיסטוריה של עד 1,000,000 רכישות. SLA=100%

iii. תמיכה ב-1,000 מבקרים במערכת בכל רגע נתון. SLA=100%

iv. המערכת אינה מפסיקה לפעול, גם כשיש אירועים לא צפויים, כמו נפילות תקשורת או קשר לרכיבים שונים (למשל פעולות סגירה יזומות). SLA = 95%

Tests Description

Our stress & load tests were performed as follows :

We performed one comprehensive test that contained the following steps :

i. Register as a user
ii. Login
iii. Create a store
iv. Add 100 products to the store
v. Perform 100 purchases

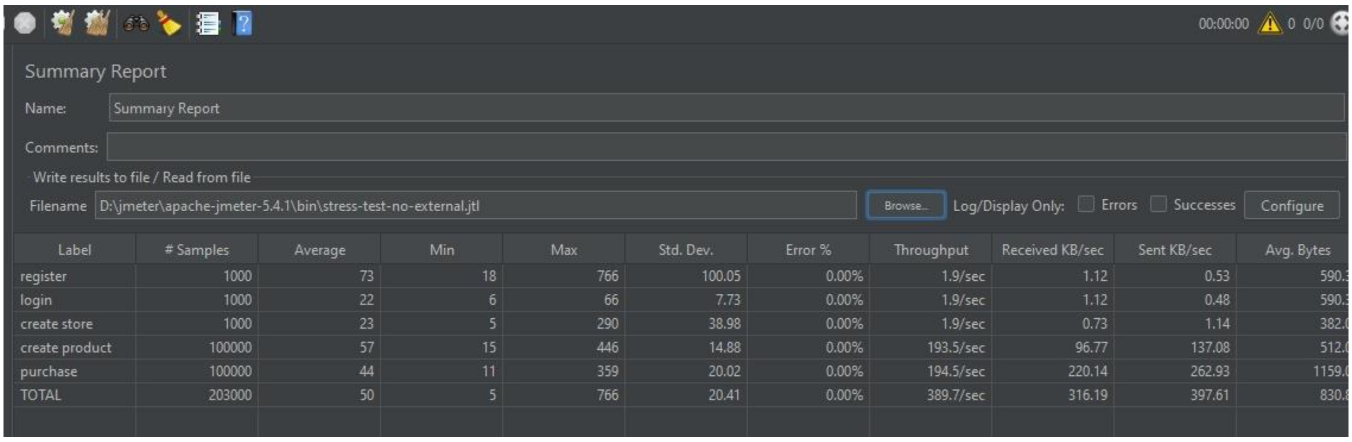
x1000 iterations (concurrently)

In addition, we performed a test that performed a constant purchase.

Lastly, we performed a simple test to handle 100 *Get Products* requests.

Results

The results of the comprehensive test when performed **without external systems** :



00:00:00 0 0/0

Summary Report

Name: Summary Report

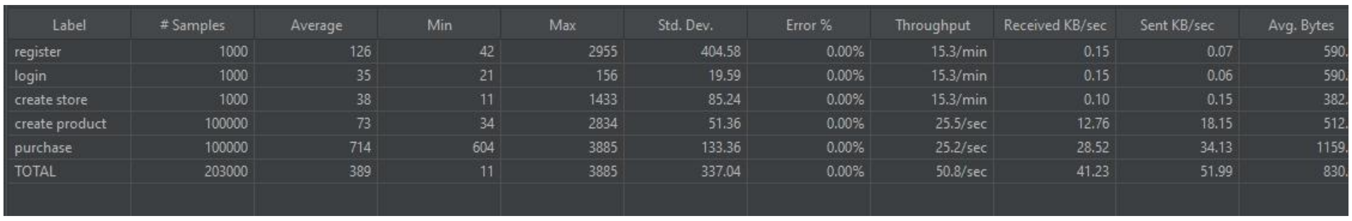
Comments:

Write results to file / Read from file

Filename D:\jmeter\apache-jmeter-5.4.1\bin\stress-test-no-external.jtl Browse... Log/Display Only: Errors Successes Configure

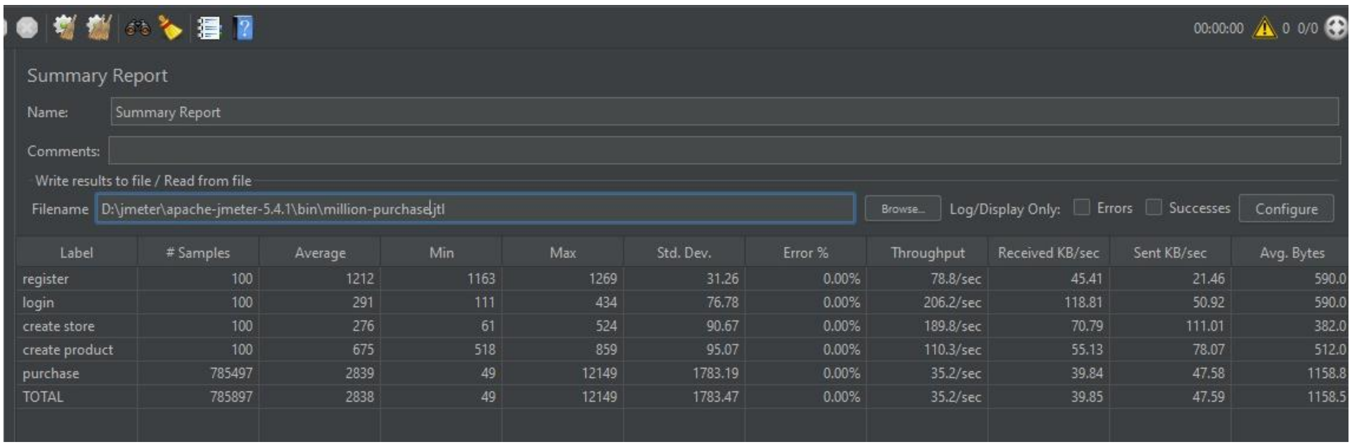
Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
register	1000	73	18	766	100.05	0.00%	1.9/sec	1.12	0.53	590.3
login	1000	22	6	66	7.73	0.00%	1.9/sec	1.12	0.48	590.3
create store	1000	23	5	290	38.98	0.00%	1.9/sec	0.73	1.14	382.0
create product	100000	57	15	446	14.88	0.00%	193.5/sec	96.77	137.08	512.0
purchase	100000	44	11	359	20.02	0.00%	194.5/sec	220.14	262.93	1159.0
TOTAL	203000	50	5	766	20.41	0.00%	389.7/sec	316.19	397.61	830.8

The results of the comprehensive test when performed **with external systems** :



Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
register	1000	126	42	2955	404.58	0.00%	15.3/min	0.15	0.07	590.
login	1000	35	21	156	19.59	0.00%	15.3/min	0.15	0.06	590.
create store	1000	38	11	1433	85.24	0.00%	15.3/min	0.10	0.15	382.
create product	100000	73	34	2834	51.36	0.00%	25.5/sec	12.76	18.15	512.
purchase	100000	714	604	3885	133.36	0.00%	25.2/sec	28.52	34.13	1159.
TOTAL	203000	389	11	3885	337.04	0.00%	50.8/sec	41.23	51.99	830.

The results of the constant purchase test :



00:00:00 0 0/0

Summary Report

Name: Summary Report

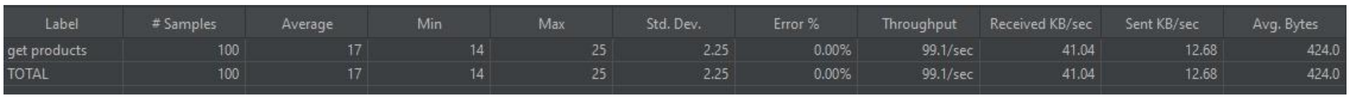
Comments:

Write results to file / Read from file

Filename D:\jmeter\apache-jmeter-5.4.1\bin\million-purchase.jtl Browse... Log/Display Only: Errors Successes Configure

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
register	100	1212	1163	1269	31.26	0.00%	78.8/sec	45.41	21.46	590.0
login	100	291	111	434	76.78	0.00%	206.2/sec	118.81	50.92	590.0
create store	100	276	61	524	90.67	0.00%	189.8/sec	70.79	111.01	382.0
create product	100	675	518	859	95.07	0.00%	110.3/sec	55.13	78.07	512.0
purchase	785497	2839	49	12149	1783.19	0.00%	35.2/sec	39.84	47.58	1158.8
TOTAL	785897	2838	49	12149	1783.47	0.00%	35.2/sec	39.85	47.59	1158.5

The results of the simple test that handles 100 *Get Products* requests :



Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
get products	100	17	14	25	2.25	0.00%	99.1/sec	41.04	12.68	424.0
TOTAL	100	17	14	25	2.25	0.00%	99.1/sec	41.04	12.68	424.0

Analysis

Comprehensive test

Comparing the results, we can confidently declare that our bottleneck lies in the utilization of the external systems (payment & shipping) .
The purchase action, when performed without the external systems, takes **44 milliseconds** on average, compared to **714 milliseconds** (approximately 16 times slower) when performed with external systems.

The bottleneck is also reflected in the **Max** column, which specifies how long took did the slower purchase take :

- **without external systems** - 359 milliseconds
- **with external systems** - 3,885 milliseconds (almost 11 times slower than the purchase without external systems, and ~ 4 seconds)

Constant purchase test

As we can see, our system dealt with **785,897** purchases (until we stopped the test) .
Each purchase took 2839 milliseconds on average (~ 2.8 seconds) which is slower than the findings in the comprehensive test.
we assumed that it happens due to the massive load on the single database.
As a result, we suggest the utilization of a **Distributed Database** to deal with the requirement of 1,000,000 purchases.

Note : We would like to point out the fact that we would probably met the requirement of 1,000,000 purchases if the purchases would have been spread across a longer period of time.

Simple Test

As requested in the formal requirements, our system response time to each *Get Products* request was indeed less than 1 second, we even reached an average response time of **17 milliseconds**, when the slowest response time (**Max**) stands at 25 milliseconds.