Τεχνολογίες Υπηρεσιών Λογισμικού

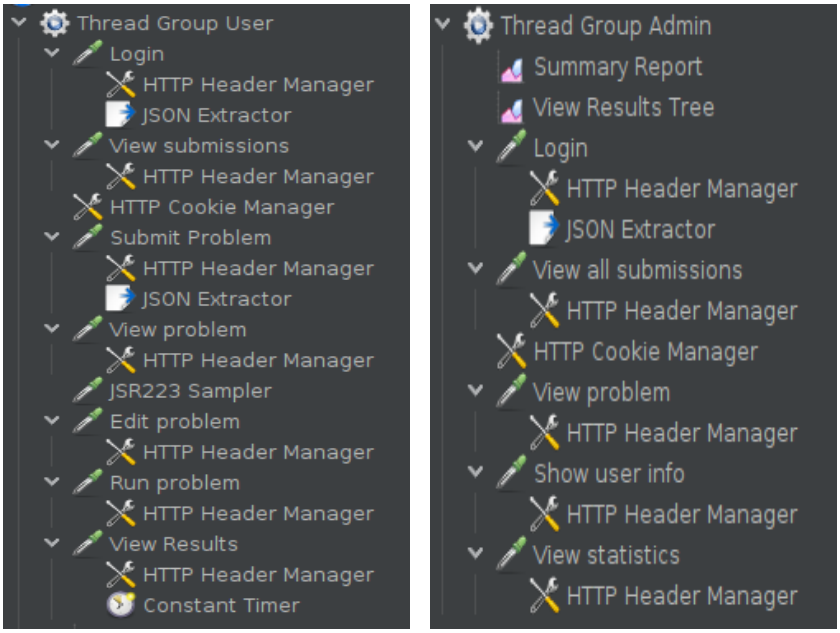
Ομάδα 19

Stress Testing με JMeter

1 Περιγραφή - Use cases

Πραγματοποιούμε stress testing χρησιμοποιώντας δύο use cases :

1. **User**: Login - View submissions - Submit problem - View problem - Edit problem - Run problem - View results
2. **Admin**: Login - View all submissions - View problem - Show user info - View statistics

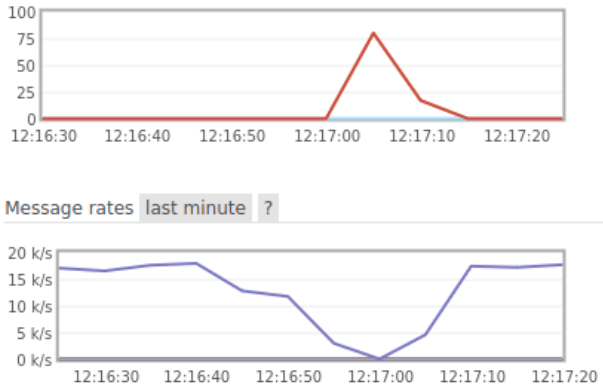


Εικόνα 1: Thread groups

Θα τρέξουμε τα παραπάνω use cases για thread groups μεγέθους 5, 50, 100 και 1000 νημάτων με ramp-up period 5 seconds (δηλαδή σε 5 seconds συνολικά, όλοι οι χρήστες μπάινουν στο σύστημα) και loop count ίσο με 1 (κάθε thread τρέχει μια φορά το εκάστοτε use case). Σημειώνουμε ότι η τελευταία κλήση του πρώτου use case (view results), για να εξάγει αποτελέσματα θα πρέπει αυτά πρώτα να έχουν παραχθεί. Θέλοντας να λάβουμε υπόψη τις καθυστερήσεις α)επίλυσης και β)αναμονής στην ουρά, εισάγουμε - πριν την κλήση του view results - ικανό χρόνο καθυστέρησης με έναν constant timer του jmeter.
Ενδεικτικά, παραθέτουμε, από το UI του Jmeter, τις μετρήσεις για το πρώτο use case γα 50 threads και το μήκος της ουράς των προβλημάτων, από το UI του RabbitMQ, που φθάνει περίπου στα 75 (δεδομένου ότι η υλοποίησή μας υποβάλει στην ουρά τόσο το αρχικό πρόβλημα όσο και το ενηρωμένο, μετά την κλήση του endpoint για update του problem).

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/...	Sent KB/sec	Avg. Bytes
Login	50	4169	119	4841	1171.91	0.00%	5.1/sec	2.53	1.11	504.0
View submis...	50	208	7	4471	714.56	0.00%	5.2/sec	1.26	1.72	248.0
Submit Probl...	50	1859	50	4813	1840.36	0.00%	5.0/sec	30.98	30.59	6331.0
View problem	50	33	2	133	38.16	0.00%	7.2/sec	44.25	2.66	6276.0
JSR223 Sam...	50	1	0	11	1.66	0.00%	7.2/sec	0.00	0.00	.0
Edit problem	50	671	165	3879	994.53	0.00%	7.0/sec	2.12	57.99	308.0
Run problem	50	258	51	4828	655.64	0.00%	7.9/sec	2.17	3.02	283.0
View Results	50	9	7	15	2.07	0.00%	9.6/sec	1.78	3.40	189.0
TOTAL	400	901	0	4841	1642.81	0.00%	19.1/sec	32.95	38.17	1767.4

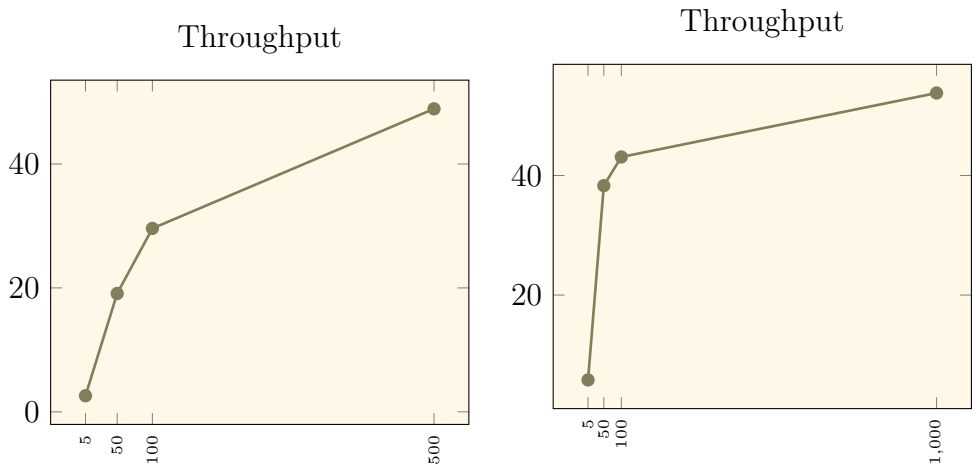
(a) Μετρήσεις



(b) Μήκος ουράς

Εικόνα 2: Πίνακας μετρήσεων και μήκος της ουράς προβλημάτων

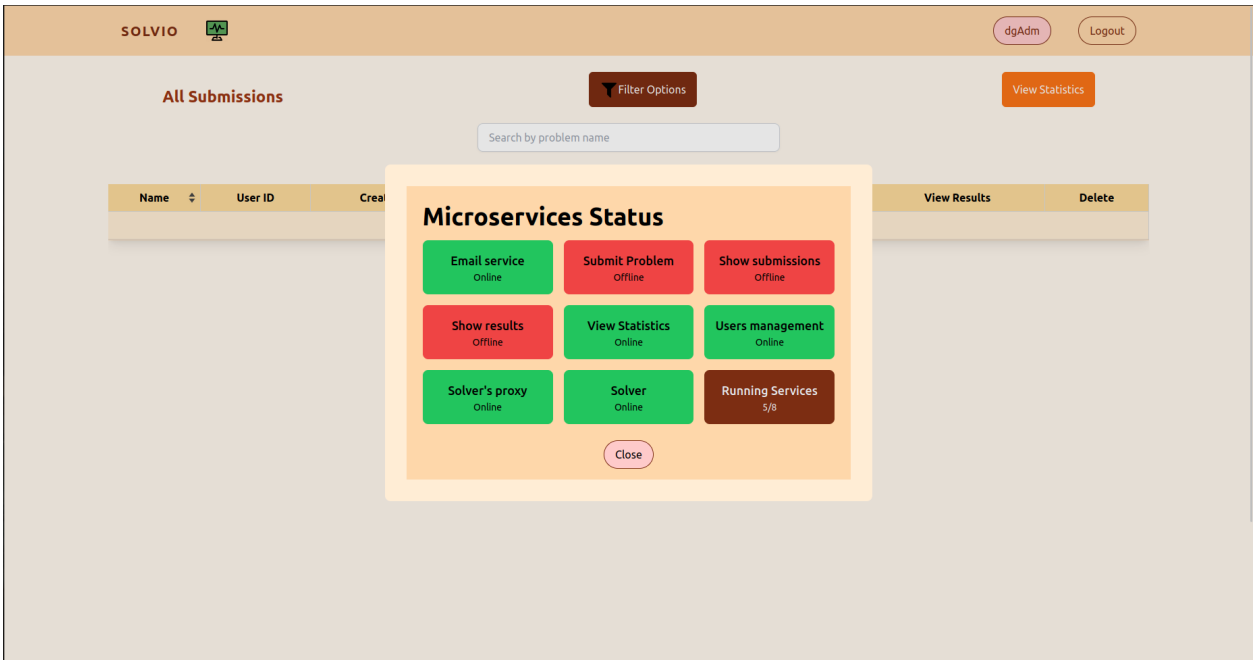
Στη συνέχεια, παρουσιάζονται στα παρακάτω διαγράμματα, οι μετρήσεις throughput για τα δύο use cases και για τα διάφορα thread groups. Παρατηρούμε ότι με αύξηση του αριθμού των νημάτων που υποβάλλουν αιτήματα, το throughput αυξάνεται.



(a) User use case (b) Admin use case

Εικόνα 3: Throughput για τα δύο use cases με διαφορετικά thread groups

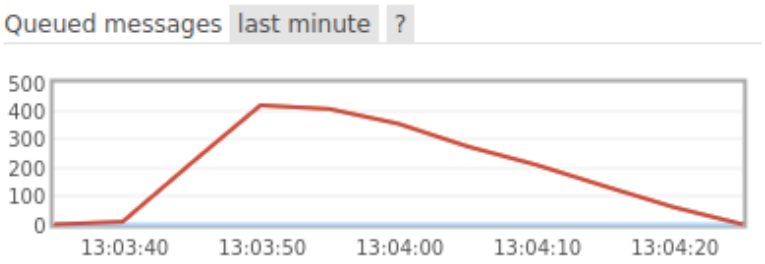
Για 1000 νήματα, το πρώτο use case οδηγεί τα microservices που φαίνονται παρακάτω να πέφτουν. Τα microservices αυτά είναι αρκετά heavy και είναι αναμενόμενο να πέφτουν αν υποφέρουν από υπερφόρτωση (ενώ ταυτόχρονα δεν έχουμε λάβει μέτρα για την αντικατάστασή τους / χρήση container orchestration system / χρήση load balancing σε συνδυασμό με το API Gateway). Επιπλέον, πιθανώς ο nginx καταλήγει να γίνεται bottleneck. Για τους λόγους αυτούς, χρησιμοποιούμε 500 νήματα για την μέτρηση παραπάνω.



(a) Μετρήσεις

Εικόνα 4: Offline microservices

Με 500 νήματα, για το πρώτο use case, το μήκος της ουράς φαίνεται παρακάτω :



Εικόνα 5: Πρώτο use case : ουρά για 500 νήματα