Partie A

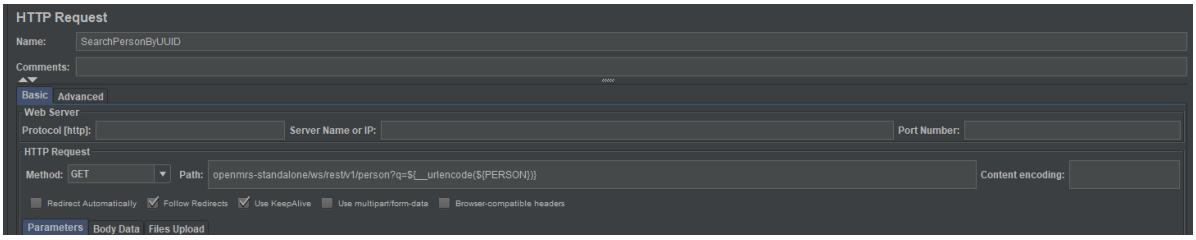
Dans la partie A, combien de demandes d'URL sont envoyées ? Toutes les demandes sont-elles traitées avec succès ? Vous devez fournir des captures d'écran de votre plan de test, avec les captures d'écran pour les parties A.f, A.g et A.h (votre plan de test), ainsi les captures des résultats e.g. de View Results Tree et Summary Report dans votre rapport. Ajouter aussi les fichiers testA.jmx et resA.jtl.

110 demandes sont envoyées, toutes traitées avec succès.

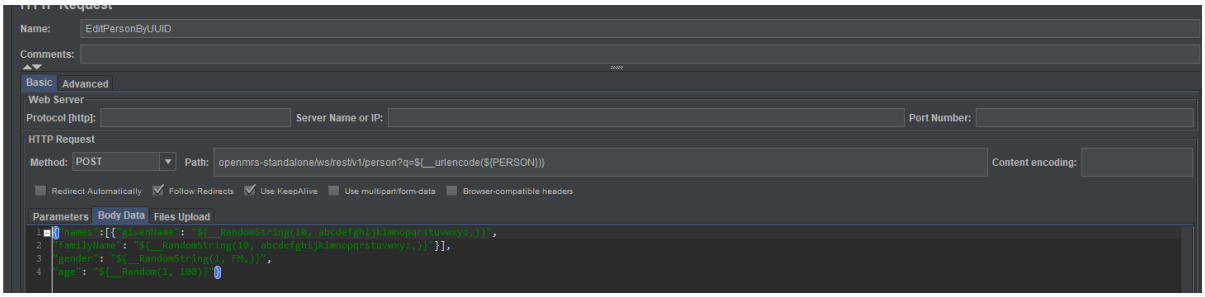
Summary Report

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
AddPerson	10	53	41	115	20.95	0.00%	34.0/min	0.62	0.19	1121.0
DeletePerson	10	32	28	38	2.90	0.00%	33.8/min	1.13	0.19	2052.0
SearchPersonByName	10	21	17	29	3.88	0.00%	33.7/min	0.38	0.16	694.3
SearchPersonByUUID	10	13	10	17	2.01	0.00%	33.4/min	0.11	0.18	205.0
EditPersonStatic	10	140	122	215	27.38	0.00%	33.3/min	0.55	0.24	1020.0
EditPersonByUUID	10	39	34	58	7.91	0.00%	33.4/min	0.56	0.24	1024.9
GetEncounterOfPatient	10	15	9	29	5.10	0.00%	33.4/min	1.77	0.18	3262.2
GetEncounterByType	10	39	7	96	39.30	0.00%	33.2/min	0.22	0.16	413.2
GetConceptIsYes	10	26	22	33	4.13	0.00%	33.2/min	0.32	0.16	595.0
GetConceptByName	10	32	27	42	4.69	0.00%	32.6/min	0.41	0.16	764.6
GetObservationOfPati...	10	27	6	38	10.64	0.00%	32.5/min	4.87	0.17	9186.1
TOTAL	110	40	6	215	37.37	0.00%	5.0/sec	9.11	1.68	1849.0

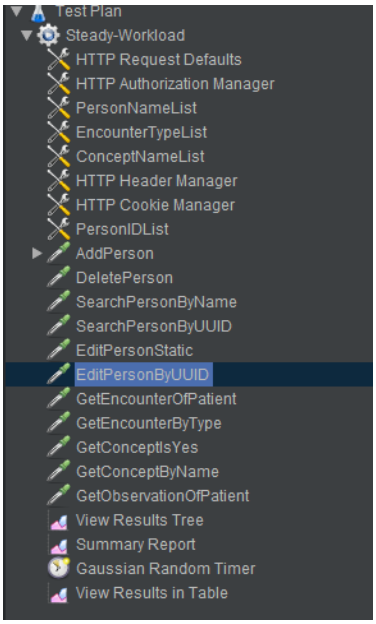
A.f



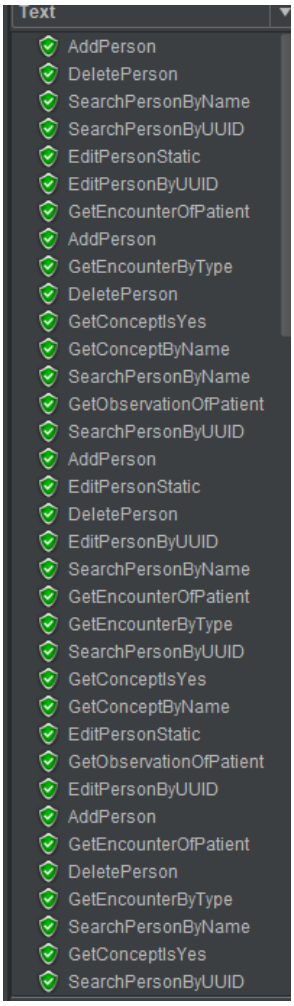
A.g



A.h



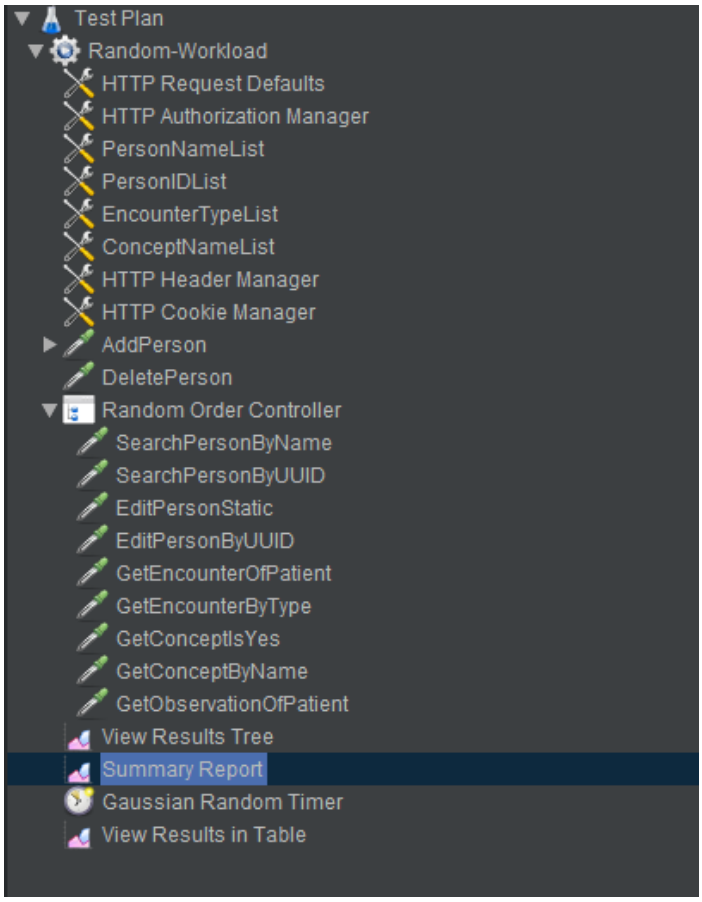
ViewResult Tree

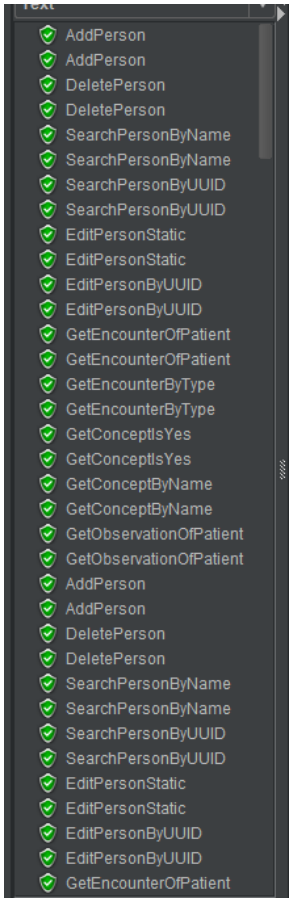


Partie B

Pour effectuer une charge de travail aléatoire dans la partie B, nous utilisons Random Order Controller. Pouvez-vous utiliser d'autres composants pour effectuer des charges de travail aléatoires? Vous devez fournir une capture d'écran de deux possibilités dans votre rapport. Ajouter aussi les fichiers testB.jmx et resB.jtl pour chaque charge de travail (avec les noms un peu différents).

Utilisation de Random Order Controller

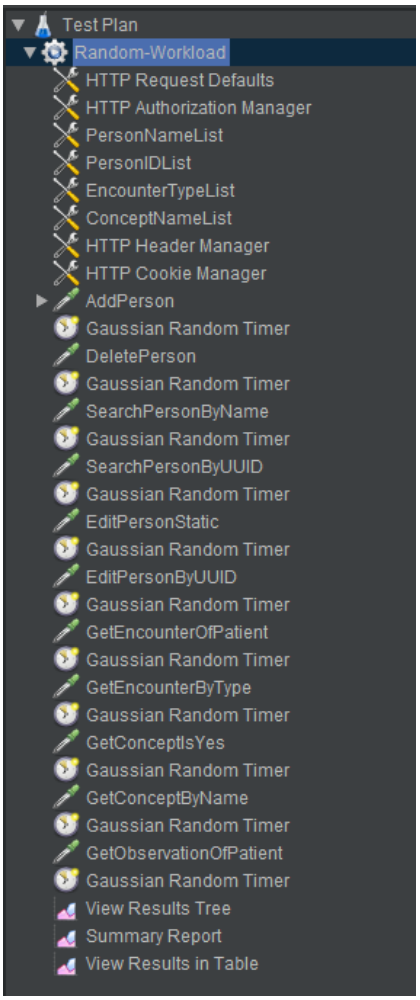


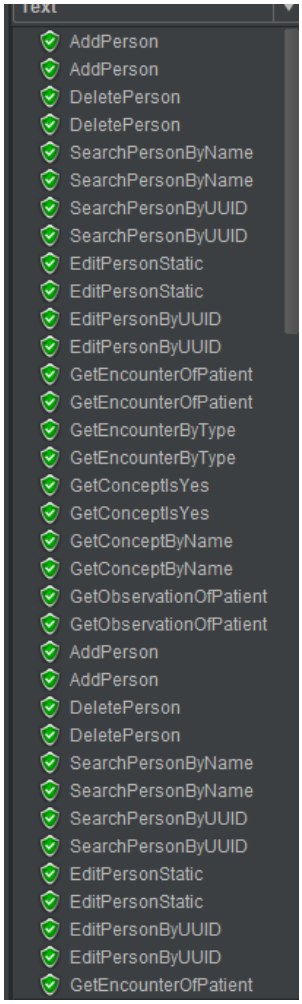


Summary Report

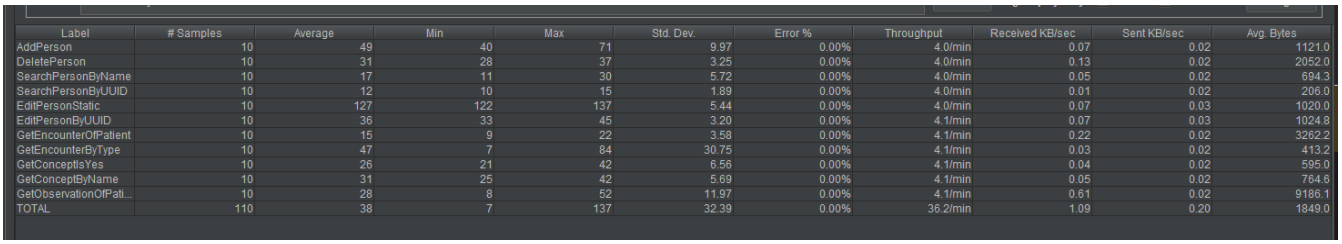
Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
AddPerson	10	44	39	57	5.05	0.00%	36.4/min	0.66	0.21	1121.0
DeletePerson	10	31	25	47	5.83	0.00%	36.4/min	1.22	0.21	2052.0
SearchPersonByUUID	10	14	10	18	2.49	0.00%	35.8/min	0.12	0.19	206.0
EditPersonStatic	10	166	147	203	14.46	0.00%	32.6/min	0.54	0.23	1020.0
EditPersonByUUID	10	36	32	43	3.66	0.00%	36.8/min	0.61	0.26	1025.1
SearchPersonByName	10	18	14	23	2.76	0.00%	35.1/min	0.40	0.17	694.3
GetEncounterByType	10	28	5	79	29.11	0.00%	36.4/min	0.24	0.18	413.2
GetEncounterOfPatient	10	13	8	18	3.32	0.00%	40.6/min	2.16	0.22	3262.2
GetObservationOfPati...	10	26	6	38	11.29	0.00%	39.1/min	5.85	0.21	9186.1
GetConceptIsYes	10	21	19	25	1.96	0.00%	39.3/min	0.38	0.19	595.0
GetConceptByName	10	26	23	30	2.39	0.00%	39.3/min	0.49	0.19	764.6
TOTAL	110	38	5	203	42.70	0.00%	5.8/sec	10.40	1.91	1849.0

Utilisation de Gaussian RandomTimer





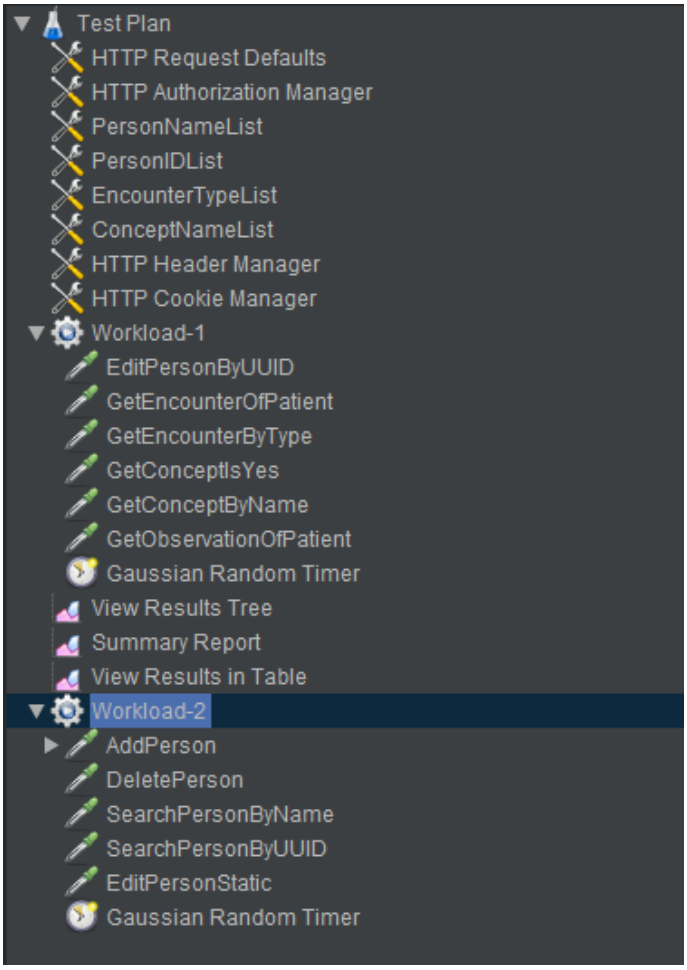
Summary Report

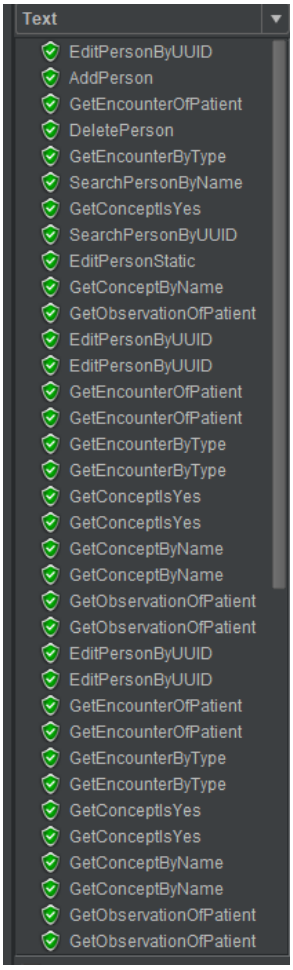


Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
AddPerson	10	49	40	71	9.97	0.00%	4.0/min	0.07	0.02	1121.0
DeletePerson	10	31	28	37	3.25	0.00%	4.0/min	0.13	0.02	2052.0
SearchPersonByName	10	17	11	30	5.72	0.00%	4.0/min	0.05	0.02	694.3
SearchPersonByUUID	10	12	10	15	1.89	0.00%	4.0/min	0.01	0.02	206.0
EditPersonStatic	10	127	122	137	5.44	0.00%	4.0/min	0.07	0.03	1020.0
EditPersonByUUID	10	36	33	45	3.20	0.00%	4.1/min	0.07	0.03	1024.8
GetEncounterOfPatient	10	15	9	22	3.58	0.00%	4.1/min	0.22	0.02	3262.2
GetEncounterByType	10	47	7	84	30.75	0.00%	4.1/min	0.03	0.02	413.2
GetConceptIsYes	10	26	21	42	6.56	0.00%	4.1/min	0.04	0.02	595.0
GetConceptByName	10	31	25	42	5.69	0.00%	4.1/min	0.05	0.02	764.6
GetObservationOfPati...	10	28	8	52	11.97	0.00%	4.1/min	0.61	0.02	9186.1
TOTAL	110	38	7	137	32.39	0.00%	36.2/min	1.09	0.20	1849.0

Partie C

Fournissez votre script de plan de test dans la partie C nommée testC.jmx, et votre fichier de résultat resC.jtl. Ajoutez les captures d'écran avec les tests et les résultats.

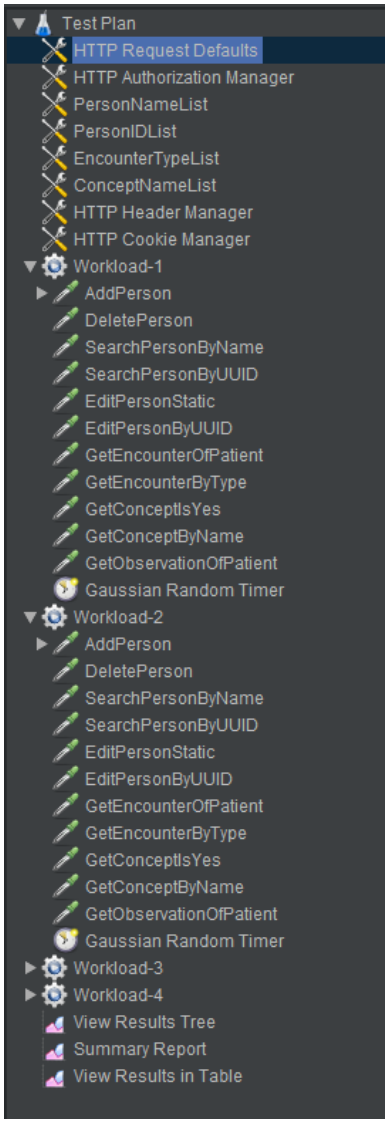


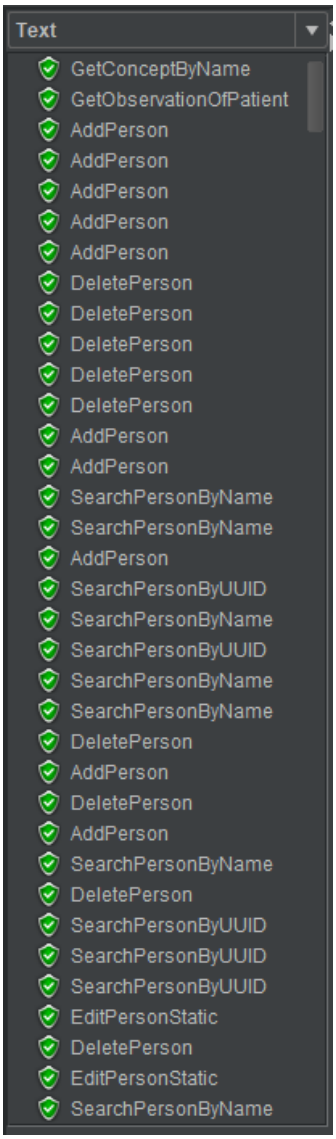


Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
EditPersonByUUID	10	50	42	76	9.36	0.00%	53.7/min	0.98	0.34	1119.0
AddPerson	1	43	43	43	0.00	0.00%	23.3/sec	25.46	7.88	1121.0
GetEncounterOfPatient	10	13	8	16	2.75	0.00%	53.5/min	2.80	0.29	3219.8
DeletePerson	1	32	32	32	0.00	0.00%	31.2/sec	62.62	10.77	2052.0
GetEncounterByType	10	25	6	107	36.90	0.00%	53.2/min	0.36	0.26	412.3
SearchPersonByName	1	24	24	24	0.00	0.00%	41.7/sec	24.94	11.88	613.0
GetConceptIsYes	10	22	22	24	0.87	0.00%	53.8/min	0.52	0.25	595.0
SearchPersonByUUID	1	13	13	13	0.00	0.00%	76.9/sec	15.47	24.19	206.0
EditPersonStatic	1	169	169	169	0.00	0.00%	5.9/sec	5.89	2.52	1020.0
GetConceptByName	10	30	27	48	5.96	0.00%	55.1/min	0.73	0.27	812.6
GetObservationOfPati...	10	26	7	48	11.70	0.00%	55.8/min	8.32	0.30	9165.2
TOTAL	65	30	6	169	25.93	0.00%	5.0/sec	11.83	1.58	2434.6

Partie D

Collectez les données de performances du processeur (en mode utilisateur) de la partie D et fournissez un graphique d'utilisation du processeur. Ajouter aussi les fichiers testD.jmx et resD.jtl et votre script pour créer le graphique et collecter les donnes de la performance (cpu, mémoire).





Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
AddPerson	15	57	41	97	14.25	0.00%	3.0/sec	3.34	1.03	1121.0
DeletePerson	15	34	26	67	9.99	0.00%	3.0/sec	6.05	1.04	2052.0
SearchPersonByName	15	16	11	25	4.03	0.00%	3.1/sec	2.51	0.87	843.1
SearchPersonByUUID	15	12	9	17	2.36	0.00%	2.9/sec	0.59	0.93	206.0
EditPersonStatic	15	230	154	442	83.98	0.00%	2.7/sec	2.71	1.16	1020.0
EditPersonByUUID	15	61	34	225	50.08	0.00%	2.8/sec	2.84	1.22	1024.9
GetEncounterOfPatient	15	15	10	23	4.11	0.00%	2.8/sec	7.65	0.89	2837.7
GetEncounterByType	15	20	6	102	29.25	0.00%	2.7/sec	1.09	0.80	412.7
GetConceptIsYes	15	23	20	31	2.95	0.00%	2.7/sec	1.58	0.77	595.0
GetConceptByName	13	34	27	79	13.39	0.00%	2.4/sec	1.62	0.73	685.7
GetObservationOfPati...	9	24	6	34	10.06	0.00%	1.8/sec	16.12	0.59	8938.3
TOTAL	157	49	6	442	69.04	0.00%	18.9/sec	28.34	6.31	1535.3

```python
def get_pid():
    output = subprocess.check_output('jps -l', shell=True)
    return re.search(b'(\d+) org\.openmrs\.standalone\.ApplicationController', output).group(1)


def perf(process_id):
    cpu_percentages = []
    memory_percentages = []
    time_stamps = []

    plt.style.use('ggplot')

    fig, ax1 = plt.subplots()
    ax1.set_xlabel('Time (s)')
    ax1.set_ylabel('CPU Usage (%)', color='blue')
    ax1.set_ylim(0, 100)

    ax2 = ax1.twinx()
    ax2.set_ylabel('Memory Usage (%)', color='green')
    ax2.set_ylim(0, 100)

    while True:
        psutil_capture_pid = psutil.Process(process_id)

        cpu_percent = psutil_capture_pid.cpu_percent(interval=1)
        memory_percent = psutil_capture_pid.memory_percent()

        cpu_percentages.append(cpu_percent)
        memory_percentages.append(memory_percent)
        time_stamps.append(len(time_stamps))

        ax1.plot(time_stamps, cpu_percentages, color='blue')
        ax2.plot(time_stamps, memory_percentages, color='green')
        plt.pause(0.05)


if __name__ == "__main__":
    pid = get_pid()
    perf(int(pid))
```

```
PS C:\Users\youne\Downloads\Labo5\Labo5\loadTests\loadTests\apache-jmeter-5.2.1\apache-jmeter-5.2.1\bin> ./jmeter -n -t
testD.jmx -l resD.jtl
Creating summariser <summary>
Created the tree successfully using testD.jmx
Starting standalone test @ Sun Apr 09 18:20:42 EDT 2023 (16810788427745)
Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump message on port 4445
summary +    401 in 00:00:17 =   23.5/s Avg:    48 Min:     5 Max:   323 Err:     0 (0.00%) Active: 20 Started: 50 Finis
hed: 30
summary +    149 in 00:00:03 =   50.0/s Avg:    69 Min:     4 Max:   387 Err:     0 (0.00%) Active: 0 Started: 50 Finish
ed: 50
summary =    550 in 00:00:20 =   27.4/s Avg:    54 Min:     4 Max:   387 Err:     0 (0.00%)
Tidying up ...    @ Sun Apr 09 18:21:03 EDT 2023 (16810788630 24)
... end of run
```