ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΡΟΗ Υ - ΕΠΙΔΟΣΗ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

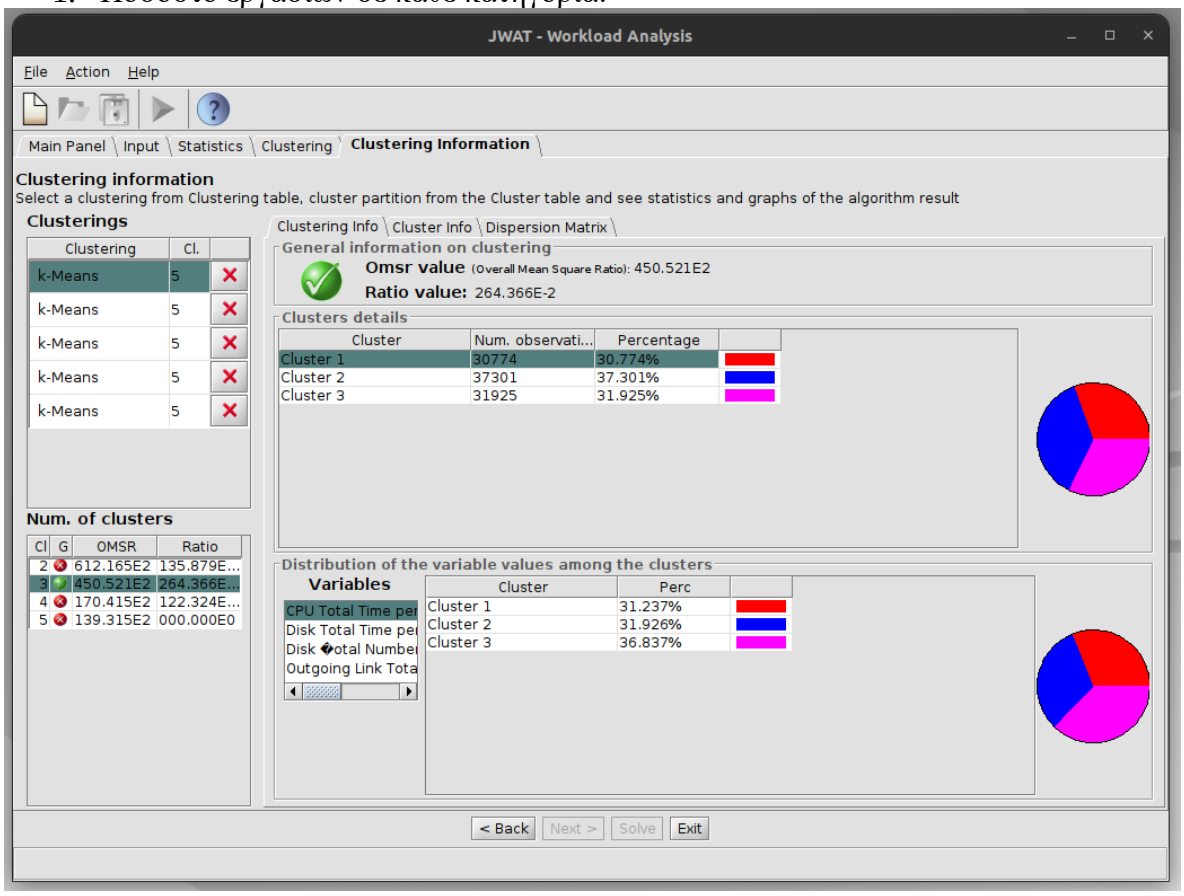
ΟΜΑΔΑ 20
Κωνσταντίνα Παπία 03120075
Γεωργακόπουλος Γεώργιος 03120827

ΑΣΚΗΣΗ 2

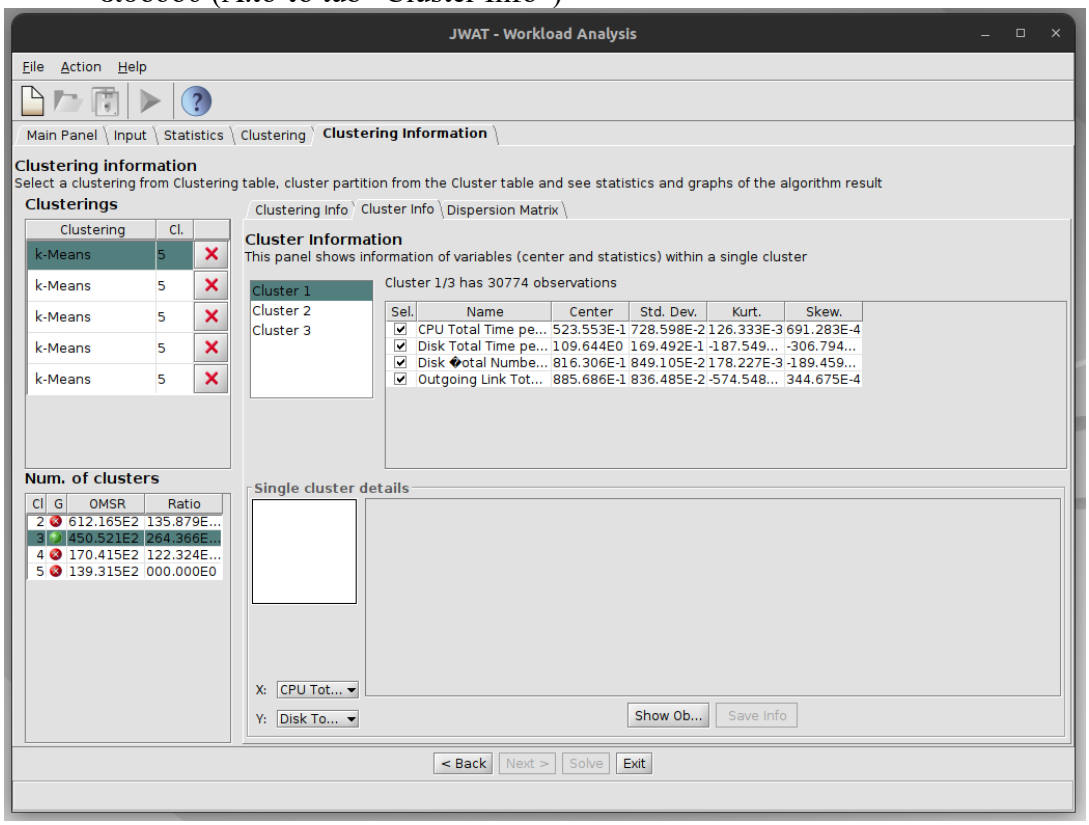
Χαρακτηρισμός Φορτίου:

Με βάση τις οδηγίες της εκφώνησης, χρησιμοποιώντας το εργαλείο JWAT και συγκεκριμένα το πακέτο JMT εξάγουμε τα εξής συμπεράσματα για τις ομάδες (κατηγορίες πελατών) :

1. Ποσοστό εργασιών σε κάθε κατηγορία:



2. Συνολικές μέσες τιμές - κέντρα (center) της κάθε ομάδας για τις 4 μεταβλητές εισόδου (Απο το tub “Cluster Info”)



- Συγκεντρωμένες όλες μαζι:

	name of variable	center			
Cluster 1					
true	CPU Total Time per Job	52.3553	7.28598	0.126333	0.0691283
true	Disk Total Time per Job	109.644	16.9492	-0.187549	-0.306794
true	Disk total Number of Visits per Job	81.6306	8.49105	0.178227	-0.189459
true	Outgoing Link Total Transmission Time per Job	88.5686	8.36485	-0.00574548	0.0344675
Cluster 2					
true	CPU Total Time per Job	44.1473	8.20788	2.4422	1.31587
true	Disk Total Time per Job	140.226	17.3161	-0.143629	-0.0811307
true	Disk total Number of Visits per Job	57.0544	6.51916	0.508299	0.184588
true	Outgoing Link Total Transmission Time per Job	78.7188	6.7621	0.22926	0.112511
Cluster 3					
true	CPU Total Time per Job	59.5157	9.22598	0.0893939	-0.310561
true	Disk Total Time per Job	164.463	15.3282	-0.158976	0.26331
true	Disk total Number of Visits per Job	69.5784	10.5495	-0.106026	0.0623742
true	Outgoing Link Total Transmission Time per Job	95.5826	8.92478	-0.123953	0.122598

Θεωρητικό Μέρος - Ανάλυση Επίδοσης:

Το μοντέλο προσομοίωσης προβλέπει ότι:

1. Η επεξεργασία στην CPU διακόπτεται αν προκύψει προσπέλαση στον δίσκο
2. Μετά την προσπέλαση, η επεξεργασία στην CPU συνεχίζεται
3. Στο τέλος της επεξεργασίας, έχουμε μετάδοση αποτελέσματος στο Internet
4. Οι τιμές των παραμέτρων υπολογίζονται άμεσα
5. Οι χρόνοι εξυπηρέτησης κατα την εισερχόμενη διεύθυνση σύνδεση είναι αμελητέα
6. Ο χρονος Εξυπηρέτησης στην CPU ακολουθεί Erlang 4
7. Οι άλλοι χρόνοι εξυπηρέτησης ακολουθούν εκθετικές κατανομές
8. Το όριο οπισθοχώρησης θα ακολουθεί κανονική κατανομή (μ=12, σ=3)

9. Αναγεννητική μέθοδο με βαθμό εμπιστοσύνης 95%

Με βάση αυτές τις ιδιότητες του μοντέλου ορίζουμε τις συνάρτήσεις: generate_task / cpu_service / disk_service / outgoing_link_service / calculate_confidence_interval

Κώδικας:

Η γενική δομή του κώδικα βασίζεται στο παράδειγμα 7.3 του βιβλίου
Ο κώδικας που χρησιμοποιήσαμε είναι ο εξής:

exercise2_team20.py > ...

```python
import numpy as np
import random as rd
import queue
import math

# Ορισμός των clusters με τις μέσες τιμές των μεταβλητών
clusters = {
    1: {
        'cpu_time': 52.3553,
        'disk_time': 109.644,
        'disk_visits': 81.6306,
        'outgoing_time': 88.5686
    },
    2: {
        'cpu_time': 44.1473,
        'disk_time': 140.226,
        'disk_visits': 57.0544,
        'outgoing_time': 78.7188
    },
    3: {
        'cpu_time': 59.5157,
        'disk_time': 164.463,
        'disk_visits': 69.5784,
        'outgoing_time': 95.5826
    }
}

# Ποσοστά των εργασιών σε κάθε κατηγορία
probabilities = [0.3077, 0.3730, 0.3193]
task_arrival_rate = 0.3  # Ρυθμός άφιξης εργασιών

def generate_task():
    cluster_id = np.random.choice([1, 2, 3], p=probabilities)
    cluster = clusters[cluster_id]
    cpu_time = np.random.normal(cluster['cpu_time'], cluster['cpu_time'] * 0.1)
    disk_time = np.random.normal(cluster['disk_time'], cluster['disk_time'] * 0.1)
    disk_visits = np.random.normal(cluster['disk_visits'], cluster['disk_visits'] * 0.1)
    outgoing_time = np.random.normal(cluster['outgoing_time'], cluster['outgoing_time'] * 0.1)

    return {
        'cluster_id': cluster_id,
        'cpu_time': cpu_time,
        'disk_time': disk_time,
        'disk_visits': disk_visits,
        'outgoing_time': outgoing_time
    }
```

```python
# Συνάρτηση που προσομοιώνει τον χρόνο εξυπηρέτησης της CPU με κατανομή Erlang 4
def cpu_service(cluster_id):
    shape = 4  # Erlang shape parameter
    mean_time = clusters[cluster_id]['cpu_time']
    rate = 1 / (mean_time / shape)
    return sum(rd.expovariate(rate) for _ in range(shape))

# Συνάρτηση που προσομοιώνει τον χρόνο εξυπηρέτησης του δίσκου με εκθετική κατανομή
def disk_service(cluster_id):
    mean_time = clusters[cluster_id]['disk_time']
    return rd.expovariate(1 / mean_time)

# Συνάρτηση που προσομοιώνει τον χρόνο εξυπηρέτησης της εξερχόμενης σύνδεσης με εκθετική κατανομή
def outgoing_link_service(cluster_id):
    mean_time = clusters[cluster_id]['outgoing_time']
    return rd.expovariate(1 / mean_time)
```

```python
# Δομές για την παρακολούθηση του συστήματος
cpu_queue = queue.Queue()
disk_queue = queue.Queue()
outgoing_queue = queue.Queue()
completed_tasks = []

total_response_times = {1: 0, 2: 0, 3: 0}
task_counts = {1: 0, 2: 0, 3: 0}
dropped_tasks = 0

# Παρακολούθηση αναγεννητικών κύκλων
regenerative_cycles = []
current_cycle_response_times = []

current_time = 0
max_cycles = 1000  # Μέγιστος αριθμός αναγεννητικών κύκλων
cycle_check_interval = 20  # Διάστημα ελέγχου αναγεννητικών κύκλων (κάθε 20 κύκλους)
max_steps = 1000000  # Μέγιστος αριθμός βημάτων για να αποφευχθεί το άπειρο βρόχο

```

```python
def calculate_confidence_interval(data, confidence=0.95):
    n = len(data)
    if n == 0:
        return float('nan'), float('nan'), float('nan'), float('nan')
    mean = np.mean(data)
    stderr = np.std(data, ddof=1) / math.sqrt(n)
    margin_of_error = stderr * 1.96  # 95% confidence
    return mean, margin_of_error, mean - margin_of_error, mean + margin_of_error

steps = 0
while len(regenerative_cycles) < max_cycles and steps < max_steps:
    # Προσομοίωση άφιξης νέας εργασίας
    if np.random.random() < task_arrival_rate:
        task = generate_task()
        task['arrival_time'] = current_time
        theta = np.random.normal(12, 3)
        #print(theta)
        # Ματαίωση εργασίας αν οι εργασίες στο σύστημα είναι περισσότερες από το όριο θ
        if cpu_queue.qsize() + disk_queue.qsize() + outgoing_queue.qsize() > theta:
            dropped_tasks += 1
        else:
            cpu_queue.put(task)
    #print("step=", steps)
    #print("cpu_queue=", cpu_queue.qsize())
    # Processor Sharing για την CPU
    if not cpu_queue.empty():
        task = cpu_queue.get()
        cluster_id = task['cluster_id']
        remaining_cpu_time = cpu_service(cluster_id)
        task['cpu_time'] -= remaining_cpu_time

        # Αν έχει υπόλοιπο CPU time, ξαναβάζουμε την εργασία στην ουρά CPU, αλλιώς πάμε στο δίσκο
        if task['cpu_time'] > 0:
            cpu_queue.put(task)
        else:
            # Μετακίνηση στο δίσκο αν έχει επισκέψεις στο δίσκο
            if task['disk_visits'] > 0:
                disk_queue.put(task)
            else:
                # Αλλιώς μετακίνηση στην εξερχόμενη σύνδεση
                outgoing_queue.put(task)
```

```python
    # Εξυπηρέτηση δίσκου με FIFO
    if not disk_queue.empty():
        task = disk_queue.get()
        cluster_id = task['cluster_id']
        disk_time = disk_service(cluster_id)
        task['disk_visits'] -= 35
        current_time += disk_time

        # Επιστροφή στην CPU ή στην εξερχόμενη σύνδεση
        if task['disk_visits'] > 0:
            cpu_queue.put(task)
        else:
            outgoing_queue.put(task)

```

```
    # Εξυπηρέτηση εξερχόμενης σύνδεσης με FIFO
    if not outgoing_queue.empty():
        task = outgoing_queue.get()
        cluster_id = task['cluster_id']
        outgoing_time = outgoing_link_service(cluster_id)
        current_time += outgoing_time

        # Ολοκλήρωση εργασίας
        response_time = current_time - task['arrival_time']
        total_response_times[cluster_id] += response_time
        task_counts[cluster_id] += 1
        completed_tasks.append(task)
        current_cycle_response_times.append(response_time)

    steps += 1

# Έλεγχος αναγεννητικού κύκλου
if cpu_queue.qsize() + disk_queue.qsize() + outgoing_queue.qsize() < 10:
    if len(current_cycle_response_times) > 0:
        mean_response_time = np.mean(current_cycle_response_times)
        regenerative_cycles.append(mean_response_time)
        current_cycle_response_times = []

    # Έλεγχος διαστήματος εμπιστοσύνης κάθε 20 κύκλους
    if len(regenerative_cycles) >= cycle_check_interval:
        mean, margin_of_error, lower_bound, upper_bound = calculate_confidence_interval(regenerative_cycles)
        if margin_of_error < 0.1 * mean:
            print("The confidence interval is less than 10 percent of the mean. Stopping simulation.")
            break

# Προσθήκη σταθερού βήματος χρόνου για την προσομοίωση
current_time += 1
print("Steps: "+ str(steps))
# Υπολογισμός συνολικών χρόνων απόκρισης και ρυθμών απόδοσης
total_tasks = sum(task_counts.values())
overall_response_time = sum(total_response_times.values()) / total_tasks if total_tasks > 0 else float('nan')
print(f"Total completed tasks: {total_tasks}")
print(f"Total dropped tasks: {dropped_tasks}")
print(f"Total time: {current_time}")
throughput = total_tasks/current_time
#if current_time > 0 else float('nan')

print(f"Overall response time: {overall_response_time:.2f}")
print(f"Overall throughput: {throughput:.6f} tasks/unit time")
```

```
print(f"Overall response time: {overall_response_time:.2f}")
print(f"Overall throughput: {throughput:.6f} tasks/unit time")

for cluster_id in clusters:
    if task_counts[cluster_id] > 0:
        cluster_response_time = total_response_times[cluster_id] / task_counts[cluster_id]
        cluster_throughput = task_counts[cluster_id] / current_time
        print(f"Cluster {cluster_id} - Average response time: {cluster_response_time:.4f}, Throughput: {cluster_throughput:.4f} tasks/unit time")
    else:
        print(f"Cluster {cluster_id} - No completed tasks")

# Υπολογισμός βαθμού χρησιμοποίησης πόρων
# Συνολικός αριθμός εργασιών που ολοκληρώθηκαν χωρίς να οπισθοχωρήσουν, χωρισμένος με τον συνολικό αριθμό των βημάτων και τον αριθμό των πόρων
total_resource_utilization = (sum(task_counts.values()) - dropped_tasks) / (steps * 3)
print(f"Total resource utilization: {total_resource_utilization:.2f}")

# Υπολογισμός ποσοστού εργασιών που οπισθοχωρούν
backward_moving_tasks_percentage = dropped_tasks / (sum(task_counts.values()) + dropped_tasks) * 100
print(f"Percentage of tasks that moved backward: {backward_moving_tasks_percentage:.4f}%")
```

Ο κώδικας μπορεί επίσης να βρεθεί στον ακόλουθο github repository:
https://github.com/ntua-el20827/NTUA_Computer_System_Performance

Σχόλια:

- Αρχικά ορίζονται τρεις κατηγορίες (clusters) με τις μέσες τιμές για τους χρόνους επεξεργασίας από την CPU, τον δίσκο και την εξερχόμενη σύνδεση, καθώς και τις επισκέψεις στον δίσκο με βάση τα αποτελέσματα απο το JWAT, όπως αναλύθηκε παραπάνω
- Χρησιμοποιούμε βοηθητικές συναρτήσεις, όπως την generate_task, που δημιουργεί μια νέα εργασία επιλέγοντας τυχαία ένα cluster βάσει των προκαθορισμένων πιθανοτήτων και τις cpu_service, disk_service και outgoing_link_service που

προσομοιώνουν τους χρόνους εξυπηρέτησης για την CPU, τον δίσκο και την εξερχόμενη σύνδεση αντίστοιχα

- Οπως και στον κώδικα του βιβλίου, χρησιμοποιούμε μια συνάρτηση ελέγχου ολοκλήρωσης κύκλου.
- Σχετικά με την αναγεννητική μέθοδο, κάθε 20 αναγεννητικούς κύκλους υπολογίζουμε το διάστημα εμπιστοσύνης. Ορίζουμε εναν αναγεννητικό κύκλο μεταξύ διαδοχικών "ιδιων" σημείων του συστήματος. Ενα τέτοιο σημείο χαρακτηρίζεται απο το πλήθος των εργασιών (συγκεκριμένα εμείς το θέτουμε σε μικρότερο του 10)
- Για την παρακολούθηση των γεγονότων σχηματίζουμε δομές όπως λίστες και ουρές, οι οποίες θα χρησιμοποιηθούν μέσα στον κύριο βρόχο προσομοίωσης, κατα τον οποίο εκτελούνται τα εξής: αφιξη εργασίας, επεξεργασία απο τα στοιχεία του συστήματος, μετακίνηση εργασιών μεταξύ ουρών, ολοκληρωση εργασίας ,ελέγχος ορίου αναγεννητικών κύκλων, υπολογισμός διαστήματος εμπιστοσύνης.
- Στο τέλος εκτυπώνουμε τα στατιστικά που υπολογίσαμε απο την προσομοίωση

Αποτελέσματα:

Εκτελώντας τον παραπάνω κώδικα παίρνουμε τα εξής αποτελέσματα:

```
george@glaptop:~/Workshop/uni/8sem/eplab/ex2$ python3 exercise2 b.py
The confidence interval is less than 10 percent of the mean. Stopping simulation.
Steps: 810
Total completed tasks: 229
Total dropped tasks: 4
Total time: 99921.54376646414
Overall response time: 1558.97
Overall throughput: 0.002292 tasks/unit time
Cluster 1 - Average response time: 1764.5100, Throughput: 0.0007 tasks/unit time
Cluster 2 - Average response time: 1332.6767, Throughput: 0.0009 tasks/unit time
Cluster 3 - Average response time: 1645.6178, Throughput: 0.0007 tasks/unit time
Total resource utilization: 0.09
Percentage of tasks that moved backward: 1.7167%
```

Σχόλια:

- Τα αποτελέσματα της προσομοίωσης μπορούν να κυμανθούν σημαντικά ανάλογα με τις παραμέτρους του συστήματος όπως οι αναγεννητικοί κύκλοι, ο ρυθμός άφιξης και άλλοι παράγοντες.
- Όλοι οι χρόνοι που χρησιμοποιούνται είναι σε msec (σύμφωνα και με την εκφώνηση)