PARALLÈLISATION MAXIMALE AUTOMATIQUE

Projet pratique en système d'exploitation

TOUATI Mounia – OUAFI Rabah : L3 CILS

Genèse du projet

- **Objectif :** Développer une librairie en Python pour automatiser la parallélisation maximale de systèmes de tâches

- **Brainstorming :**

Sémaphore
Threading
Multiprocessing

Graphviz
Networkx
Matplotlib

On a choisi :
Threading
Graphviz

Organisation du travail

01.

Brainstorming

La première séance de projet

02.

Documentation

Sur les différents points du projet et les techniques possibles.

03.

Choix e la technique

Threading : facile à comprendre.
Sélection des librairies nécessaires.

04.

Implémentation des classes de base

Classes Task et TaskSystem, les méthodes simples: getDependencies, constructeurs

05.

Pseudo codes des méthodes difficiles

Telles que Run, Verification des Entrées et draw

06.

Implémentation des méthodes

Implémentation des méthodes et la validation du code avec des tests.

Travail d’équipe

Répartition des taches

Réunions: Call Discord Ou en TD

Fusion et correction des Codes

Difficultés rencontrées

- Le choix de la technique
- L’implémentation du parallélisme
- Dessin du graphe d’exécution en parallèle
- Charge de travail

Structure du code

Classes	Attributs	Méthodes
Task	name = "" | reads = [] writes = [] | run = None (fonction)	runtask()
TaskSystem	listtask = [] | precedences ={}	getDependencies(task.name) runSeq() run() draw() run_task(task) conditionBernstein() verification() parCost() detTestRnd()

Étapes de parallélisme

ÉTAPE 1

Initialiser les ensembles pour les taches en cours et les taches achevées

ÉTAPE 2

Exécuter les taches sans dépendances en simultané à l'aide de threads

```
def run(self):
    global executed_task
    executed_task = set()  # un set des taches terminées
    launched_task = set()  # un set des taches en cours d'execution

    #on parcours les liste des taches totale
    while len(executed_task) < len(self.listtask):
        for t in self.listtask:
            if t.name not in launched_task:
                if self.stats_dep(t):
                    launched_task.add(t.name)
                    threading.Thread(name=t.name, target=self.run_task, daemon=True, args=(t,)).start()

# fonction pour exécuter les taches et lister les taches terminées
def run_task(self, t):
    t.runtask()
    executed_task.add(t.name)  # ajout de la tache à la liste des terminée après la fin de son execution

def stats_dep(self, t):
    if len(self.getDependencies(t.name)) == 0:
        etat = True
    else:
        etat = True
        for dep in self.getDependencies(t.name):
            if dep not in executed_task:
                etat = False
    return etat
```

ÉTAPE 3

Synchroniser et conclure l'exécution des taches parallélisées

ÉTAPE 4

Traiter les taches dépendantes après achèvement des prérequis

SEQUENTIEL VS PARALLELISME

Exécution des tâches en séquentiel puis en parallèle

```
t1 = Task("T1", ["M1", "M2"], ["M3"], runT1)

t2 = Task("T2", ["M1"], ["M4"], runT2)

t3 = Task("T3", ["M3", "M4"], ["M1"], runT3)

t4 = Task("T4", ["M3", "M4"], ["M5"], runT4)

t5 = Task("T5", ["M4"], ["M2"], runT5)

t6 = Task("T6", ["M5"], ["M5"], runT6)

t7 = Task("T7", ["M1", "M2", "M4"], ["M4"], runT7)

t8 = Task("T8", ["M1", "M3"], ["M5"], runT8)

sys = TaskSystem([t1, t2, t3, t4, t5, t6, t7, t8], {'T1': [], 'T2': ["T1"], 'T3': ["T2"], 'T4': ["T2"],
                                                    'T5': ["T3", "T4"], 'T6': ["T4"], 'T7': ["T5", "T6"], 'T8': ["T7"]})

sys.parCost()

sys.draw()
```

```
/usr/local/bin/python3 /Users/mouniatouati/Downloads/maxp.py
mouniatouati@MacBook-Air-de-Mounia ~ % /usr/local/bin/python3 /Users/mouniatouati/Downloads/maxp.py
Calcul du coût des systèmes en cours...
 liste des temps du système séquentiel : [14.0295, 14.0328, 14.0265, 14.0304, 14.0315]
 Temps moyen d'execution du run séquentiel : 14.03014
 liste des temps du système parallèl: [10.0683, 10.0651, 10.0681, 10.0639, 10.0651]
 Temps moyen d'execution du run parallèl : 10.0661
mouniatouati@MacBook-Air-de-Mounia ~ %
```

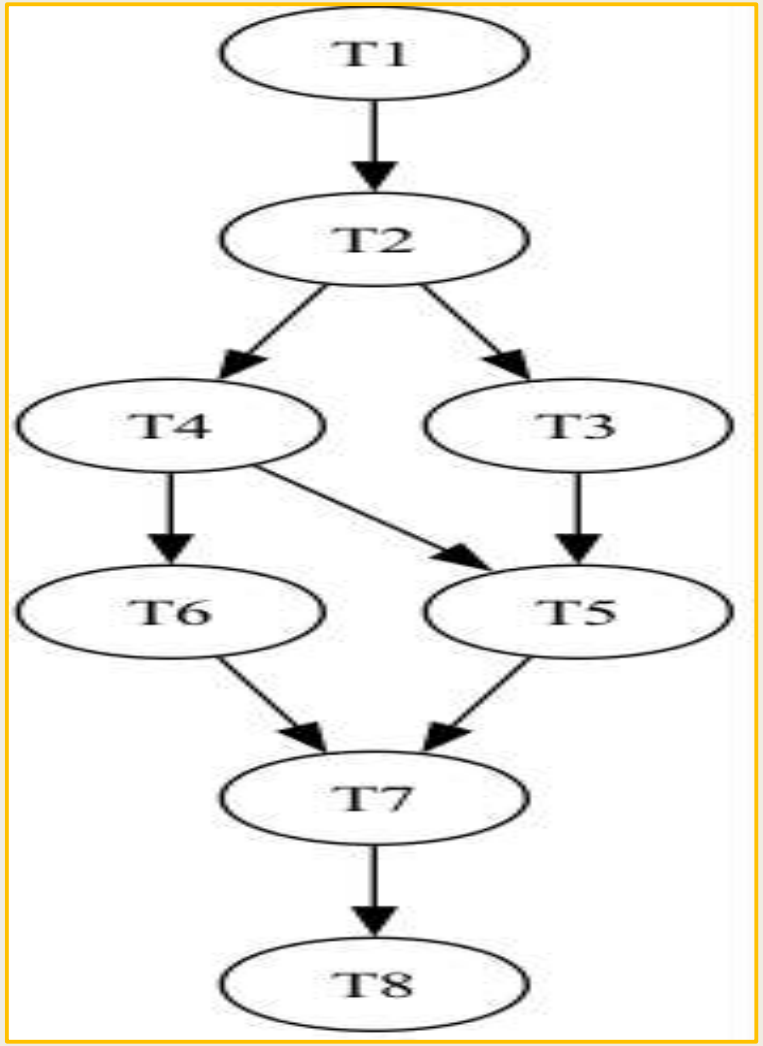
Temps moyen d’exécution en parallèle (10,0661s)

Temps moyen d’exécution en séquentiel (14,03014s)

Test randomisé et jeu de valeurs

```
mouniatouati@MacBook-Air-de-Mounia ~ % /usr/local/bin/python3 /Users/mouniatouati/Downloads/maxp.py
Test du système avec des jeux de valeurs en cours..
 Teste 1 sur 5, Voici les valeurs de départ pour chaque variable:
 M1=5 M2=9 M3=5 M4=10 M5=3
 Les Valeurs des variables après execution du séquentiel
 M1=-47 M2=259 M3=1 M4=-12123 M5=-42
 Les Valeurs des variables après execution du parallèl
 M1=-47 M2=259 M3=1 M4=-12123 M5=-42
 Teste 2 sur 5, Voici les valeurs de départ pour chaque variable:
 M1=1 M2=6 M3=8 M4=7 M5=9
 Les Valeurs des variables après execution du séquentiel
 M1=-2 M2=41 M3=3 M4=-75 M5=-1
 Les Valeurs des variables après execution du parallèl
 M1=-2 M2=41 M3=3 M4=-75 M5=-1
 Teste 3 sur 5, Voici les valeurs de départ pour chaque variable:
 M1=1 M2=1 M3=5 M4=2 M5=10
```

GRAPHE DES TÂCHES EXÉCUTÉES EN PARALLÈLE



MERCI DE VOTRE ATTENTION