Parallélisation maximale automatique

Projet - Systèmes d'exploitation

Samy Beniddir, Shayicoul Akbare, Zacharya Bougdou
L3 Miage
2021/2022

Sommaire

1. Introduction
2. Notre approche
3. Notre organisation
4. Nos difficultés
5. Focus sur une fonction
6. Conclusion

Introduction - Rappel du projet

Définition :

- Un système de tâches est de parallélisme maximal s'il contient le minimum des contraintes de précédence nécessaires pour assurer son déterminisme.

Introduction - Rappel du projet

Étapes :

1. Lister les interférences entre les différentes tâches avec les conditions de Bernstein
2. Déterminer à partir des interférences les relations de précédences à imposer pour éviter les interférences
3. Supprimer les arcs qui n'entraînent pas d'interférence
4. Ajouter les relations de précédences trouver pour éviter les interférences
5. Supprimer les redondances = supprimer les chemins les plus courts

Approche / Nos choix

- On a suivis les étapes de la parallélisation maximal
  - Division des étapes en différentes fonctions

- Nous n’avons pas utilisé de sémaphores

Organisation

Samy	Shayicoul	Zacharya
Fonction d'obtention des précédences d’une tâche	Fonction de suppression d’interférences	Condition de Bernstein
Fonction run	Fonction de suppression des redondances	Fonction de parallélisation des tâches

Session en vocal sur discord + LiveShare sur VSCode

Nos difficultés

- Élimination de redondance (Nous somme sur la bonne voie pour y arriver !)
  - Difficultés à identifier le chemin le plus court
  - Difficultés à les retirer
  - Difficultés à parcourir tout le graphe

- Suppression des interférences (Problèmes résolu)
  - Difficultés à parcourir les précédences

Focus sur une fonction

-> Fonction de la suppression d'interférence

-+Pseudo-code (notre point de départ**):**

vérifier condition Bernstein
SI interférence ALORS
vérifier que t1 n'est pas déjà une précédence de t2
SINON
je passe t1 en précédence de t2
SINON
aucune interférence

Focus sur une fonction

```
def SupprInter(nomt1,nomt2):
    if not bernstein(nomt1, nomt2):#vérifie si interferance
        if(dic[nomt2.name] == []): #si t2 n'a pas de precedences
            print(dic)
            dic[nomt2.name].append(nomt1.name)
            print(dic)
            return False
        for i in dic[nomt2.name]: #parcours les précédences de la tache 2
            if(i == nomt1.name): #Et ce que t1 est deja dans les precedences de t2 ?
                print("relations de précédence entre",nomt1.name,nomt2.name)
                return False
            else:
                print(dic)
                dic[nomt2.name].append(nomt1.name) #ajout de t1 dans les précédence de t2
                print(dic)
                break
    else:
        print("Aucune interferance entre les deux taches "+nomt1.name+ " et " +nomt2.name)
```

Focus sur une fonction

-> Fonction parallélisation des tâches

```
def ParaTache(nomt1,nomt2):#Paralléliser les taches qui n'ont pas d'interférences
    if bernstein(nomt1, nomt2):#pas d'interférance
        print(dic)
        dic[nomt2.name].remove(nomt1.name) #suppression de t1 dans les précédences de t2
        print(dic)
```

Conclusion

- Notre compréhension du sujet
  - identifier les différentes étapes de la parallélisation maximale
  - Compréhension de l'utilité de la parallélisation maximale

- Notre ressenti face au projet
  - Aimer nous documenter sur python
  - travail de groupe