Laborator 03

Exerciții

- 1. Compilati si rulati codul din raceCondition.c.
 - Parametrii sunt: (la fel pentru restul programelor)
 - N numărul de elemente (argument 1).
 - printLevel controlează câte elemente sunt printate (argument 2).
 - P numărul de thread-uri (argument 3). Este suprascris cu 2.
 - Atenţie: Compilaţi fără -O3 (optimizările vor şterge mai tot)
 - o Ce rezultat obtineti?
 - Dar cu N mare? (1000-10.000)
 - Dar rulat de câteva ori cu N = 1?
- 2. Rulati programul folosind scriptul testCorrectnessIntensive.sh.
 - Parametrii:
 - Program secvential pentru comparatie (poate fi si cel paralel)
 - Program paralel ce va fi rulat de multe ori
 - N va fi dat ca parametru programelor secvential si paralel
 - Numărul de rulări ale programului paralel
 - Optional, numărul de thread-uri (default 1 2 4 8)
 - Vom rula cu N mic (chiar 1) şi Număr de rulări mare (10.000)
 - o Care este rezultatul?
- 3. Rezolvati race condition-ul folosind mutex.
 - Testaţi iarăşi cu scriptul.
- 4. Modificati programul barrier.c.
 - Se doreşte ca mereu afişarea să se facă într-o anumită ordine.
 - Se vor folosi doar bariere.
 - Programul este gata doar după ce este testat cu scriptul oferit (10.000).
- 5. Modificati programul semaphoreSignal.c.
 - Se doreste ca mereu afisarea să se facă într-o anumită ordine.
 - Se vor folosi doar semafoare.
 - Programul este gata doar după ce este testat cu scriptul oferit (10.000).
- 6. Rezolvati problema din **deadLock1.c** fără a scoate lock de pe mutex.
 - De ce avem dead lock?
- 7. Rezolvați problema din **deadLock2.c** fără a scoate lock de pe mutexA și mutexB.
 - De ce avem dead lock?
 - Mai avem si dacă scoatem acele sleep-uri?
 - o Rulati de foarte multe ori cu scriptul pentru a fi siguri (100.000) blocheaza (fara sleepuri)
- 8. Rezolvati problema din **deadLock3.c** fără a scoate lock de pe mutex.
 - De ce avem dead lock?
 - ATENTIE: În unele limbaje se poate face lock pe un mutex de pe acelasi thread care a făcut o dată lock.
- 9. Paralelizati programul sumVectorValues.c.
 - Măsuratii timpii de executiei ai variantei secventiale si variantei voastre.



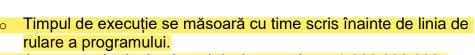


- la N mare, sunt probleme Obs: scriptul nu mai sterge fisierele de out.x.x, ar trebui sa le stearga si sa le lase doar pe cele

-> pe la testul 200 se

diferite

Arhitecturi Paralele



- Avem nevoie de timpi mari deci vom rula cu 1.000.000.000.
- Nu este obligatoriu să obțineți scalabilitate.
- Dacă nu obţineţi scalabilitate, explicaţi de ce nu?
- Exercițiul este gata doar după ce ați testat cu scriptul (N și număr de rulări mari aproximativ 10.000)

Exercițiile de la 1 la 9 sunt **obligatorii**. Conceptele explorate sunt esențiale pentru obținerea notei **minime** de promovare.

Vă recomandăm, pentru a crește șansele de a obține o notă cât mai mare să explorați și următoarele exerciții:

- 10. Implementați programul precedent în așa fel încât să scaleze.
 - Hint: Mai multe sum.
- 11. Paralelizați programul prepStrassen.c.
 - Veti folosi mai multe functii diferite pentru thread-uri.
 - Fiind funcții diferite, numărul de thread-uri poate fi fix.
 - Se vor folosi doar bariere pentru sincronizare.
 - Extrem de mare atentie la dependinte read-read, read-write, write-write.
 - Afișarea trebuie să fie identică la oricât de multe rulări.

