

## Naše otázky

::Otázka 1::

Prečo je v poriadku, že ten istý proces je dlhšie vykonávaný na väčšom počte vlákien procesora ako v prípade, keď je počet jadier procesora menší? {

~%50% Lebo aj viacjadrový procesor je nútený striedať činnosti rôznych programov.

~%50% Lebo viacjadrový procesor má určité "režijné náklady", ktoré ho spomaľujú.

~%-100% Lebo viacjadrový procesor má príliš vysoké "režijné náklady".

~%-100% Lebo vždy musí čakať na to, kým iný program dokončí manipuláciu z dátami. }

::Otázka 2::

Ako sa v programovacom jazyku nazývajú paradigmy používané pre programovanie viacjadrových programov? {

~%-100% rozhranie java.awt.io

~%-100% trieda implements

~%50% rozhranie (interface) Runnable

~%50% rozšírenie triedy Threads }

::Otázka 3::

Vyberte správne možnosti pre šifrovací algoritmus AES: {

~%-100% Je to prúdový šifrovací algoritmus.

~%-100% Veľkosť šifrovacích blokov závisí na verzii algoritmu.

~%33% Je to blokový šifrovací algoritmus.

~%33% Šifruje dáta vždy po 16B blokoch.

~%34% Veľkosť kľúča závisí na verzii algoritmu.

~%-100% Veľkosť kľúča je vždy 32B. }

::Otázka 4::

Vyberte vstavané triedy v Jave, ktoré je možné využívať pre riadenie prístupu vlákien k častiam kódu: {

~%-100% AccessMonitor

~%-100% ThreadController

~%50% ReentrantLock

~%50% Semaphore

~%-100% Synchronizer }

## 2. Prezentácia

### Otázka 1

Aký je hlavný rozdiel medzi vláknom (Thread) a procesom (Process)?

- A) Vlákno zdieľa pamäť s ostatnými vláknami v tom istom procese, zatiaľ čo procesy majú vlastnú pamäť. ✓
- B) Vlákna bežia na viacerých CPU jadrách, zatiaľ čo procesy bežia len na jednom.
- C) Procesy sú vhodnejšie pre sieťovú komunikáciu, zatiaľ čo vlákna pre výpočtové úlohy.
- D) Multithreading a Multiprocessing fungujú v Pythone úplne rovnako.

### Otázka 3

Aká je hlavná úloha Minimax algoritmu v Connect 4?

- A) Vyhodnotiť všetky možné ťahy a vybrať najlepší ťah pre AI. ✓
- B) Náhodne vybrať jeden zo stípcov pre AI.
- C) Počítat skóre iba pre aktuálny ťah, bez pohľadu na budúce stavy.
- D) Minimalizovať počet ťahov hráča bez ohľadu na víťazstvo AI.

### Otázka 2

V paralelnej implementácii konverzie obrázka na odtiene šedej, akú úlohu zohráva objekt Queue z knižnice multiprocessing?

- A) Služi len na počítanie aktívnych procesov a nemá vplyv na výsledný obrázok.
- B) Poskytuje synchronizáciu medzi procesmi a umožňuje hlavnému procesu zbierať čiastkové výsledky z dcérskych procesov. ✓
- C) Funguje ako zámok (mutex), ktorý zabráňuje viacerým procesom súčasne pracovať na rovnakej časti obrázka.
- D) Ukladá všetky výsledky do databázy pre spracovanie po dokončení všetkých procesov.

## Otázka 4

Ktoré z nasledujúcich tvrdení o Multiprocessingu je správne?

- A) Multiprocessing spúšťa viaceré vlákna v rámci jedného procesu.
- B) Každý proces má svoju vlastnú pamäť a interpret.**
- C) Multiprocessing je vhodný iba pre I/O operácie.
- D) Všetky procesy v Multiprocessingu vždy zdieľajú globálnu pamäť.

### 3. Prezentácia

Ktoré z nasledujúcich tvrdení sú správne ohľadom paralelného implementovania Bubble Sortu pomocou Odd-Even Sort?

- Nepoužívajú sa bariéry na synchronizáciu vlákien
- Každé vlákno musí spracovávať celé pole naraz

Aké je využitie bariéry v paralelnom Odd-Even Sort algoritme?

- Umožňuje vlákna bežať bez akýchkoľvek zdržaní
- Každé vlákno je nezávislé a nemusí čakať na ostatné

ako je definovaný kľúč pri algoritme Hill Cipher?

- B) Náhodné číslo
- C) Sekvencia bitov
- D) Jedno slovo

Aký je hlavný účel funkcie `pthread_create` v knižnici Pthread?

- A) Zastavenie všetkých vlákien a ich synchronizácia.
- B) Vytvorenie nového vlákna a jeho spustenie.
- C) Získanie výsledku z vlákna po jeho ukončení.
- D) Uvoľnenie pamäte alokovanej pre vlákno.

#### 4. Prezentácia

Ktorý synchronizačný mechanizmus v Kotlině by ste použili, ak potrebujete obmedziť počet súčasne bežiacich úloh na maximálne 3?

- Mutex.
- Semaphore(3)
- AtomicInteger
- CoroutineScope s launch

Chcete spracovať veľké množstvo úloh, ktoré trvajú rôzne dlhý čas, a nechcete blokovať hlavné vlákno. Čo by ste mali použiť?

- CoroutineScope a launch.
- asyn na spracovanie, await na získanie výsledkov
- Vytvorenie množstva Thread objektov manuálne
- Použitie runBlocking, aby sa hlavné vlákno zastavilo a počkalo na všetky úlohy



## Aký je hlavný dôvod použitia CoroutineScope namiesto priameho vytvárania Thread?

- Zlepšenie compatibility s Java kódom.
- Efektívne riadenie a rušenie paralelných úloh
- Každá korutina beží vo vlastnom vlákne, rovnako ako Thread
- Znižuje režijné náklady na prepínanie medzi vláknami
- Je to jediná možnosť na vytváranie vlákien v Kotlin

## Ktoré z nasledujúcich možností sú vhodné na paralelné spracovanie v Kotlin?

- Použitie CoroutineScope s launch na vykonávanie paralelných úloh.
- Použitie async a await na výpočet viacerých hodnôt naraz.
- Vytvorenie veľkého množstva Thread objektov manuálne.
- Použitie volatile na synchronizáciu komplexných zdieľaných dát.
- Použitie Mutex na ochranu kritickej sekcie pred súčasným prístupom viacerými vláknami.

## 5. Prezentácia

### Otázky

- 1. Pri paralelnom násobení matíc v Go sa na vytváranie súbežných úloh používajú \_\_\_\_\_.
- Gorutiny
- 2. Aká je časová zložitosť IJK algoritmu na násobenie dvoch štvorcových matíc veľkosti  $n \times n$ ?
- $O(n^3)$
- $O(n \cdot \log n)$
- $O(n^2)$
- $O(2^n)$

## Otázky

- 3. Ktoré 3 metódy používa `sync.WaitGroup` spôsob synchronizácie v Golangu ?
  - `Add()`
  - `Done()`
  - `Wait()`
  - `Start()`
- 4. Akým príkazom sa nastaví počet CPU Jadier, ktoré sa budú používať pri exekúcii paralelného programu v jazyku Golang ?
  - `runtime.GOMAXPROCS(n)`
  - `runtime.MAXTHR(n)`
  - `runtime.NumCPU(n)`
  - Nedá sa

## 6. Prezentácia

### Otázka 1

Ako Rust zabezpečuje bezpečnosť pamäte?

- A) Použitím garbage collectoru na správu alokácie a dealokácie pamäte počas behu programu.
- B) Implementáciou systému vlastníctva a kontroly zapožičania na validáciu referencií počas kompilácie.**
- C) Vyžadovaním manuálnej správy pamäte programátorom, podobne ako v jazykoch ako C.
- D) Povolením neobmedzeného prístupu k pamäti, spoliehajúc sa na programátora, že sa vyhne chybám.

21/24

### Otázka 2

Na čo sa využíva `Barrier` pri paralelnom spracovaní v Rust?

- A) Umožňuje vláknam spustiť sa automaticky bez potreby volať `thread::spawn`.
- B) Používa sa na dočasné uloženie výsledkov spracovania, aby sa šetrila operačná pamäť.
- C) Zabezpečuje, že všetky vlákna dokončia určitú fázu predtým, než ktorákoľvek z nich prejde k ďalšej fáze.**
- D) Nahrádza potrebu mutexov alebo iných foriem uzamykania pri písaní do spoločnej premennej.

### Otázka 3

Ktorá z nasledujúcich možností môže viesť k zníženiu efektivity paralelného tréningu MLP?

- A) Rozdelenie údajov do väčších skupín na spracovanie v každom vlákne
- B) Používanie vysoko výkonných optimalizátorov
- C) Neoptimalizovaná synchronizácia medzi vláknami**
- D) Zníženie počtu tréningových epoch

### Otázka 4

Ktorý z nasledujúcich parametrov MLP má najväčší vplyv na výpočtovú zložitosť počas tréningu?

- A) Počet neurónov vo výstupnej vrstve
- B) Veľkosť údajov (BATCH\_SIZE)
- C) Počet parametrov modelu (váhy a offset)**
- D) Aktivačná funkcia vo výstupnej vrstve

## 7. Prezentácia

1. Ktorý z nasledujúcich príkazov OpenMP zabezpečí, že premenná bude upravovaná bezpečne len jedným vláknom naraz?

- a) `#pragma omp parallel for reduction(+:x)`
- b) `#pragma omp shared(x)`
- c) `#pragma omp critical` ✓**
- d) `#pragma omp atomic cancel`



2. Aký bude dôsledok nastavenia `parallel = 0` pri volaní funkcie `multiply_matrix(...)`?

- a) OpenMP automaticky použije maximálny počet vlákien
- b) Funkcia sa vykoná sekvenčne bez použitia paralelizácie ✓
- c) Spôsobí chybu počas kompilácie
- d) Matice sa vynásobia pomocou GPU

3. Ktorý z nasledujúcich mechanizmov v OpenMP slúži na zabezpečenie bezpečného prístupu viacerých vlákien k jednej premennej bez použitia critical sekcie?

- a) `#pragma omp atomic` ✓
- b) `#pragma omp shared`
- c) `#pragma omp for`
- d) `#pragma omp collapse`

4. Prečo môže mať zmena poradia cyklov (napr. z IJK na KJI) v násobení matíc zásadný vplyv na výkon, aj bez paralelizácie?

- a) Mení sa výstupná hodnota výslednej matice
- b) Zmení sa počet operácií násobenia
- c) Ovplyvní sa spôsob prístupu do pamäte (cache locality) ✓
- d) Umožní automatickú vektorovú optimalizáciu pre GPU