Programowanie I R
Zadania seria 14

03.06.2025

Programowanie współbieżne i asynchroniczne

Zadanie 1: callspeed – Czas współbieżnego i równoległego wywołania funkcji

Napisz funkcję `fcall`, która przyjmuje jako argumenty wywołania liczbę naturalną n, pewną funkcję oraz ewentualne argumenty wywołania tej funkcji. Funkcja `fcall` powinna n-krotnie wywołać funkcję przekazaną jej jako drugi argument z ewentualnymi jej argumentami i zwrócić łączny czas trwania wszystkich n wykonań tej funkcji.

Napisz także funkcje `fcallThread` i `fcallProcess`, działające podobnie, jak funkcja `fcall`. Funkcja `fcallThread` powinna jednak realizować każde z n wykonań funkcji będącej jej argumentem w osobnym wątku, zaś funkcja `fcallProcess` – w osobnym procesie.

Przygotuj ponadto kilka funkcji realizujących dowolne czasochłonne zadania (np. tworzenie dużej listy liczb pseudolosowych i sortowanie jej za pomocą metody `sort`, tworzenie takiej listy i sortowanie jej metodą bąbelkową, tworzenie macierzy liczb pseudolosowych i jej diagonalizacja z wykorzystaniem pakietu NumPy). Funkcje powinny być napisane tak, by ich wykonanie zajmowało co najmniej kilka sekund.

Korzystając ze wszystkich tych funkcji, napisz program `callspeed`, przyjmujący jako argument wywołania liczbę naturalną k. Program powinien wykonać każdą z napisanych przez Ciebie czasochłonnych funkcji na trzy sposoby, używając do tego funkcji `fcall`, `fcallThread` i `fcallProcess` z liczbą wykonań równą k, i wypisać na ekranie czas trwania tych wykonań w każdym przypadku.

Opracowanie: Bartłomiej Zglinicki

Zadanie 2: racing - Porównanie wydajności pakietów obliczeniowych

Czasami nie mamy czasu na pełną analizę optymalnego zastosowania obliczeń współbieżnych i równoległych pod daną architekturę i chcemy skorzystać z gotowych rozwiązań. Z pomocą przychodzą nam pakiety obliczeniowe, takie jak `numpy`, `scipy`, `pytorch`, `jax`. Warto więc wiedzieć, jak radzą sobie one z danymi problemami numerycznymi.

Proszę przygotować program porównujący wydajność z wykorzystaniem powyższych bibliotek na zwektoryzowanych obliczeniach: rozwiązywania problemu liniowego $A\vec{x} = \vec{y}$, rozkładu na wartości własne $A = \sum_\lambda \lambda M_\lambda$, iloczynu trzech macierzy macierzy $A \times B \times C$.

Program na wejściu powinien przyjmować wymiar n macierzy oraz ich liczbę b (wymiar batcha). Proszę zadbać o to, by każda możliwa operacja była zwektoryzowana. Na wyjściu proszę wypisać porównanie czasu obliczeniowego.

Sugestie: Istnieje wiele sposobów obliczania iloczynu macierzy, np przez funkcje `matmul`, `einsum`, `sum`. Czy uwzględnienie liczb zespolonych zmieni czas obliczeniowy? Co jeśli macierze będą kolejno symetryczne, hermitowskie lub niehermitowskie.

UWAGA Z powodu ograniczeń pamięci na komputerach OKWF proszę instalować pakiet `pytorch` ograniczony jedynie do obliczeń na CPU. W przypadku instalcji pełnego pakietu na komputerach z systemem linux instalator pobiera bardzo duże biblioteki CUDA. By wybrać konkretną wersję wystarczy podać:
`pip install torch --index-url https://download.pytorch.org/whl/cpu`

Opracowanie: Bartosz Kasza

Zadanie 3: linkscraper – Wydobywanie hiperłączy ze stron internetowych

Napisz program `linkscraper`, przyjmujący dwa argumenty wywołania: łańcuch tekstowy reprezentujący adres URL strony internetowej oraz liczbę naturalną. Program powinien przeanalizować kod HTML strony o podanym adresie i odnaleźć w nim wszystkie hiperłącza (reprezentowane przez znaczniki `a` z atrybutem `href`) do stron internetowych (adres zaczyna się od `http://` lub `https://`), a następnie dla każdego z nich wypisać adres strony, na którą wskazuje (czyli wartość atrybutu `href`).

Gdy liczba naturalna przekazana programowi jako drugi argument wywołania jest równa 1, po wykonaniu tego zadania praca programu powinna się zakończyć. Jeśli liczba ta jest równa 2, program powinien wykonać to samo zadanie dla wszystkich stron, do których odnośniki znalazł na przeanalizowanej stronie; gdy jest ona równa 3, program powinien wykonać to samo zadanie również dla stron, do których odnośniki znalazł na stronach linkowanych na pierwotnej stronie itd.

Program powinien wykorzystywać współbieżność: każda ze stron powinna zostać pobrana i przeanalizowana w osobnym wątku.

Wskazówka: Kod HTML strony WWW o zadanym adresie URL możesz uzyskać, posługując się na przykład modułem `urllib.request`. Hiperłącza występujące w kodzie HTML najłatwiej odszukać posługując się wyrażeniami regularnymi lub parserem HTML (np. `Beautiful Soup`).

Opracowanie: Bartłomiej Zglinicki