Ćw 1 - ProcessingUnitAccess

sobota, 23 marca 2024 13:21

Program ma symulować działanie algorytmów planowania dostępu do procesora dla zgłaszających się procesów.
Zbadać średni czas oczekiwania procesów dla różnych algorytmów planowania:
1. FCFS - „First Come First Served”
2. SJF (bez wywłaszczania) - „Shortest Job First”
3. SRTF (SJF z wywłaszczaniem)
4. rotacyjnego (z możliwością wyboru kwantu czasu) - „Round Robin”
Należy samodzielnie sformułować założenia symulacji.

Wskazówki:
• Algorytmy należy sprawdzać dla tych samych danych testowych (tj. tych samych ciągów zgłaszających się procesów)
• Ciągów testowych powinno być więcej (20? 50?); wynikiem będą wartości średnie.
• W każdym ciągu będzie N procesów o losowych długościach fazy procesora (rozkład długości faz dobrać tak, by odpowiadał sytuacji w rzeczywistym systemie, w którym nie jest równomierny), zgłaszających się w losowych momentach (dobrać parametry tak, by mogła powstać kolejka procesów oczekujących na przydział procesora).
• Możliwa reprezentacja procesu: klasa (numer, długość fazy procesora, moment zgłoszenia się, czas oczekiwania /początkowo równy 0/...)

Uzyskane wyniki należy wytłumaczyć i być gotowym na wyciągnięcie z nich wniosków. Wymagana możliwość sterowania parametrami symulacji. Przy zaliczeniu należy być przygotowanym na ew. pytania dotyczące materiału omówionego na wykładzie i związanego z tematem zadania.

- Jednym z możliwych podejść jest przyjęcie bezwymiarowej jednostki czasu, która będzie odpowiadała jednej iteracji w pętli. Wtedy w każdej iteracji można zmniejszać czas potrzebny do wykonania o 1, o ile żądanie aktualnie jest wykonywane przez CPU.

- Na potrzeby wykazania pewnej słabości SJF sugerowałbym podawanie najdłuższego czasu oczekiwania jednego procesu.
- Podobnie w przypadku RR należałoby przyjąć pewną jednostkę (być może wartość z przedziału (0,1], być może więcej niż 1) czasu dla przełączania pomiędzy obsługą procesów (to oczywiście również należy uwzględnić w innych algorytmach), co pozwoli zauważyć pewne słabości RR przy zbyt małym kwancie czasu.
- Wyliczanie średniego czasu od rozpoczęcia wykonywania pewnego zadania do jego zakończenia również pozwoli wykazać różnice w zachowaniu FCFS w stosunku do algorytmów wywłaszczających takich jak SJF czy RR.

- Warto rozważyć czy lepiej podawać wartości uśrednione pewnych parametrów czy np. sumę (np. liczbę przełączeń pomiędzy żądaniami).

- Odnośnie algorytmu SJF należy zastanowić się, jak oznaczyć procesy zagłodzone (takie, które utknęły w kolejce na zbyt długi czas, w teorii mógłby on być nieskończony).

- FCFS to oczywiście kolejka FIFO, więc wydaje się być naturalną strukturą do wykorzystaniu przy realizacji tego algorytmu (choć oczywiście można posługiwać się zwykłą tablicą czy jakąś kolekcją).
- Do SJF w jakiś sposób będziecie Państwo prawdopodobnie używali posortowanych kolekcji, więc być może możliwe jest zastosowanie wydajnych metod wyszukiwania elementów w zbiorach uporządkowanych (np. wyszukiwania binarnego).

FCFS (First Come First Served)

- przydział czasu w kolejności zgłaszania się procesów
- najprostsza implementacja (kolejka FIFO)
- brak wywłaszczania
- kłopotliwy w systemach z podziałem czasu
- przykład:
  - Procesy i ich zapotrzebowanie na CPU: P1 (24), P2 (3), P3 (3)
  - Jeżeli procesy przybyły w kolejności P1, P2, P3: · czas oczekiwania: P1 = 0, P2 = 24, P3 = 27 · średni czas oczekiwania: (0 + 24 + 27)/3 = 17.
  - Jeżeli procesy przybyły w kolejnooeci P2, P3, P1: · czas oczekiwania: P1 = 6, P2 = 0, P3 = 3 · średni czas oczekiwania: (6 + 0 + 3)/3 = 3
- efekt: krótkie procesy są wstrzymywane przez długie
- zalety: sprawiedliwy, niski narzut systemowy
- wady: długi oeredni czas oczekiwania i wariancja czasu oczekiwania, nieakceptowalny w systemach z podziałem czasu

SJF (Shortest Job First)

- Algorytm wiąże z każdym procesem długość jego najbliższej fazy procesora. Procesor jest przydzielany najkrótszemu.
- Przy równych fazach procesora mamy FCFS
- SJF jest optymalny (dowód: umieszczenie krótkiego przed długim bardziej zmniejsza czas oczekiwania krótkiego niż zwiększa długiego)
- SJF może być:
  - niewywłaszczający
  - wywłaszczający (gdy dł. fazy nowego jest krótsza niż to, co zostało aktualnie wykonywanemu procesowi)
- Problem: jak oszacować długość przyszłej fazy procesora?
  - planista długoterminowy może wymagać jego zadeklarowani od procesów (zadania maj ą predefiniowany czas fazy); krótkoterminowy nie może (wielkie opóźnienia)
  - częste rozwiązania: dł. następnej fazy (t_{n+1})= średnia wykładnicza długości n faz poprzednich (założenie: dł fazy jest zależna od długości poprzednich faz):
    - $t_{n+1} = \Sigma_{i=0}^{n} a(1-a)^i t_{n-i}$ gdzie $a<1$.
  - powyższe rozwiązania jest adaptacyjne (dostosowuje się gdy zapotrzebowanie procesu ulega zmianie)

PLANOWANIE PRIORYTETOWE

- szczególnym przypadkiem jest SJF (w którym priorytet = 1/(dł nast. fazy procesora)).
- zwykle priorytet określa jednak liczba całkowita np. z przedziału [0,7] czy [0,4096] – niższa wartość = wyższy priorytet.
- problem: proces o niskim priorytecie może się nigdy nie wykonać (w jednym z przemysłowych systemów IBM proces czekał na wykonanie od 1967 do 1973...); rozwiązanie: postarzanie procesów (zmniejszenie priorytetu o 1 np. co 10 min.)
- może być wywłaszczające (ale niekoniecznie)
- określenie priorytetu:
  - zewnętrznie (przez funkcję systemową)
  - wewn. przez deklarację samego procesu (na podstawie np. wymagań: pamięć, procesor etc...)

PLANOWANIE ROTACYJNE (*Round Robin - RR*, pol. karuzelowe)

- ustala się kwant czasu (~10-100 ms)
- kolejka procesów jest traktowana jak cykliczna FIFO
- algorytm przegląda kolejkę i kolejno przydziela każdemu kwant czasu (jeśli proces się w nim nie zakończy – wraca do kolejki, a długość jego fazy procesora zmniejsza się o kwant czasu).
- algorytm jest wywłaszczający
- gdy jest N procesów a kwant czasu wynosi Q, max. czas oczekiwania może wynieść (N-1)Q
- efekty:
  - gdy Q rośnie nieograniczenie; planowanie rotacyjne → FCFS
  - gdy Q zmierza do 0 zachodzi *dzielenie procesora* – każdy proces dysponuje procesorem o prędkości 1/N rzeczywistego.

- a propos: podobny efekt dają tzw. procesory peryferyjne – z np. 10 zestawami rejestrów; na każde zadanie. Procesor peryf. wykonuje po 1 rozkazie każdego zadania. Zysk: procesor jest szybszy od pamięci – całość nie jest 10x wolniejsza...)
- Aspekty wydajnościowe:
  - wskazany jest długi kwant czasu w porównaniu z przełączaniem kontekstu (wpp zła wydajność)
  - reguła doświadczalna: kwant czasu powinien być *dłuższy* niż 80% faz procesora dla optymalnej wydajności.