Systemy operacyjne – zadanie 4

Zadanie 4 – badania algorytmów przydziału ramek:

Uwagi ogólne:

Zadanie 4 jest naturalnym rozwinięciem zadania 3. Do tej pory porównywaliśmy algorytmy zastępowania stron przy założeniu, że w systemie działa jeden proces generujący ciąg odwołań do stron. Oczywiste jest, że takie założenie jest dużym uproszczeniem w stosunku do prawdziwego systemu komputerowego. Podczas realizacji tego zdania należy zapoznać się i porównać strategie przydziału ramek dla wielu procesów działających jednocześnie w systemie operacyjnym.

Podobnie jak przy poprzednich zadaniach konieczne jest zapewnienie możliwości modyfikacji parametrów symulacji oraz zapewnienie możliwości generowania każdorazowo nowego ciągu odwołań.

Proszę pamiętać, że każdy proces generuje własny ciąg odwołań, która po połączniu należy traktować jako globalny ciąg odwołań. Należy zaznaczyć, że globalny ciąg odwołań powinien uwzględniać, że odwołania generowane przez różne procesy pojawiają się w różnych momentach czasu dla wszystkich procesów jednocześnie, nie jest więc wynikiem „sklejenia" wygenerowanych ciągów do formy prostej sekwencji [c(P1), c(P2), c(P3)], gdzie Px – kolejny proces działający w systemie, c – ciąg odwołań wygenerowany przez proces Px.

Istotne dla wygenerowanych ciągów odwołań jest, aby zachować zasadę lokalności odwołań dla każdego z procesów osobno oraz zapewnienie, że zbiory użytych stron dla procesów są rozłączne, tzn. nie istnieje taka strona, z której korzystałby więcej niż 1 proces.

Ponownie proszę o zastanowienie się nad parametrami symulacji oraz przeprowadzenie testów dla przypadków brzegowych. Analogicznie do poprzednich zadań konieczne jest aby każdy z algorytmów operował na tym samym ciągu odwołań.

Podczas badań algorytmów przydziału ramek należy również pamiętać o występowaniu zjawiska szamotania i konieczności przygotowania statystyk, które pozwolą na zaobserwowanie tego zjawiska podczas symulacji. Realizacja zbierania takich statystyk może polegać na wykrywaniu faktu, że w pewnym, z góry założonym czasie *w* (na potrzeby symulacji należy oczywiście przyjąć, że jedno odwołanie to 1 jednostka czasu) występuje więcej błędów strony niż przyjęty próg *e* (intuicyjnie można przyjąć tutaj ½ *w,* zachęcam jednak do eksperymentów). Przypominam również, że zjawisko szamotania polega na ciągłej wymianie aktywnych stron w ramkach, czego przyczyną najczęściej jest przydzielanie zbyt małej liczby ramek procesowi. Kluczowe znaczenie ma tutaj występowanie lokalności odwołań. W skrajnych przypadkach, kiedy użyty algorytm nie jest w stanie przeciwdziałać szamotaniu może dość do wstrzymania procesu, co również należy odnotować w statystykach.

Przetestowane powinny zostać 4 algorytmy, poniżej znajdziecie Państwo krótki opis, sugeruję jednak zapoznać się również z materiałami wykładowymi oraz literaturą.

1) **Przydział równy**

   Najprostszy sposób statycznego przydziału ramek. Niech N będzie liczbą procesów działających w systemie, a F liczbą ramek dostępnych w systemie.

$$f_i = \frac{F}{N}$$

f_i – liczba ramek przydzielona procesowi p_i

2) Przydział proporcjonalny

Przydział proporcjonalny polega na przydziale procesowi p_i liczba ramek f_i proporcjonalna do liczby używanych przez niego stron s_i.

$$S = \sum_{i=1}^{N} s_i$$

$$f_i = \frac{s_i}{N}$$

3) Sterowanie częstością błędów strony

Algorytm sterowania częstością błędów strony jest algorytmem dynamicznego przydziału ramek do procesu. Należy tutaj przyjąć 2 graniczne progi częstości błędów stron *l* oraz *u*, a samą częstość błędów (*PPF*, ang. page fault frequency) wyznaczymy:

$$PPF_i = \frac{e_i}{\Delta t}$$

Gdzie:

- e_i - liczba błędów strony generowanych przez proces p_i
- Δt – przyjęte okno czasowe dla pomiaru PPF

Na początku działania algorytmu przydzielamy każdemu procesowi ramki z użyciem standardowego algorytmu przydziału proporcjonalnego. W trakcie działania systemu (symulacji) monitorujemy współczynnik *PPF* dla każdego procesu, jeżeli dojdzie do sytuacji, że *PPF* przekorczy próg *u* dany proces otrzymuje dodatkową ramkę. W przypadku spadku *PPF* poniżej progu *l* dany proces zwalnia jedną z ramek. W przypadku, gdy nie ma dostępnych wolnych ramek dany proces jest wstrzymywany. Do algorytmu sterowania częstością błędów strony można wprowadzić modyfikację polegającą na akceptowaniu przekroczenia progu *u* przez wartość PPF, utrzymaniu aktywności procesu, a wstrzymywanie go dopiero po przekroczeniu pewnej wartości *h* (odpowiednio wysokiej), która określa, kiedy należy proces wstrzymać. W tym wariancie, procesowi przydzielamy dodatkowe ramki po przekroczeniu *u* ale wstrzymujemy dopiero przy przekroczeniu *h*.
Istotne dla symulacji tego algorytmu jest dopieranie odpowiednich wartości progowych oraz okna czasu.

4) Model strefowy

Model strefowy wykorzystuje koncepcję zbioru roboczego, który jest związany z występowaniem zjawiska lokalności odwołań. Ponownie przyjmujemy Δt, tym razem do wyznaczenia rozmiaru zbioru roboczego (*WSS*, ang. working set size). *WSS* jest liczbą stron, do których proces p_i wygenerował odwołania w czasie Δt. Liczbę aktualnie potrzebnych ramek *D* możemy wyznaczyć:

$$D = \sum_{i=1}^{N} WSS_i$$

Dopóki *D* jest mniejsze niż liczba dostępnych w systemie ramek każdy z procesów otrzymuje do wykorzystania tyle ramek, ile wynosi jego *WSS*. W momencie przekroczenia liczby dostępnych ramek przez współczynnik *D* jeden z aktywnych procesów musi zostać wstrzymany, a uwolnione ramki przekazane do pozostałych procesów (zgodnie z zasadą proporcjonalności). Strategie wyboru procesu do wstrzymania mogą być różne: zatrzymanie procesu o najmniejszym *WSS* (ryzyko konieczności wstrzymania wielu procesów), zatrzymanie procesu o największym *WSS* (ryzyko zatrzymania dużego, intensywnie pracującego procesu), zatrzymanie procesu o najniższym priorytecie (konieczność określania priorytetów) czy zatrzymanie procesu o wymaganej liczbie ramek (wzrost złożoności algorytmu + wprowadzenie pewnej losowości w wyborze procesu).

Równie istotne jest określenie momentu wyznaczania *WSS*. Obliczanie tej wartości przy każdym odwołaniu powodowałoby znaczące obciążenie systemu, więc byłoby nieefektywne (a wielu przypadkach zbędne). Rozsądnym wydaje się przyjęcie pewnej wartości *c* takiej, że $c < \Delta t$. Sugeruję poeksperymentować z konkretnymi wartościami zaczynając od $c = ½\ \Delta t$.

Uwagi końcowe:

Proszę, aby zaproponowane wartości stanowiły jedynie punkt wejścia podczas badań, oczekuję, że spróbujecie Państwo uruchomić symulacje z innymi wartościami i spróbujecie określić warunki, w których ujawniają się wady wymienionych algorytmów.

W przypadku dalszych pytań i wątpliwości pozostaję do Państwa dyspozycji i proszę o kontakt poprzez Teams lub mailowo.