Wykład 4
Zarządzanie procesami.
Planowanie procesora

Jarosław Koźlak

Planowanie przydziału procesora. Algorytmy planowania

Planowanie wywłaszczające

- Decyzje o przydziale procesora zapadają w następujących sytuacjach:

1. Proces przeszedł od stanu aktywności, do stanu czekania (np. z powodu zamówienia na we/wy, lub rozpoczęcia czekania na zakończenie któregoś z procesów potomnych)
2. Proces przeszedł od stanu aktywności do stanu gotowości (np. wskutek wystąpienia przerwania)
3. Proces przeszedł od stanu czekania, do stanu gotowości (np. po zakończeniu operacji we/wy)
4. Proces kończy działanie

Planowanie niewywłaszczeniowe i wywłaszczeniowe

- **Planowanie niewywłaszczeniowe** (ang, non preemptive) - planowanie uwzględniające tylko warunki 1 i 4.
- proces, który dostał procesor, nie odda go aż do swego zakończenia lub przejścia w stan czekania
- stosowane np. w Microsoft Windows
- nie wymaga wsparcia sprzętowego (np. zegara)

- **Planowanie wywłaszczeniowe** (ang. preemptive) - uwzględnia poza 1 i 4 także 2 i 3
- kosztowne - wymaga mechanizmów koordynacji

Ekspedytor (ang. dispatcher)

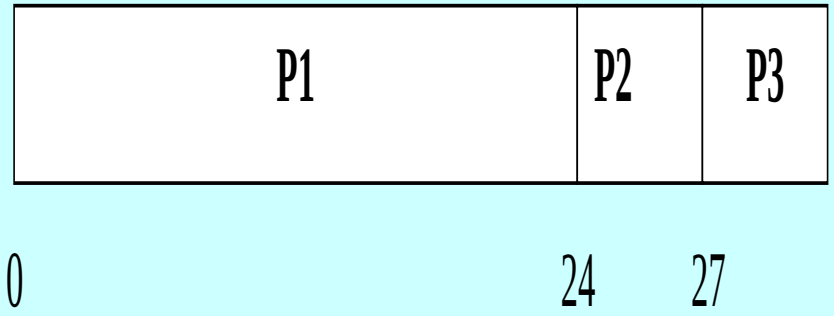
- **Ekspedytor -** przekazuje procesor do dyspozycji procesu wybranego przez planistę krótkoterminowego.
- Zadania ekspedytora:
- przełączanie kontekstu
- przełączanie do trybu użytkownika
- wykonanie skoku do odpowiedniej komórki w programie użytkownika, w celu wznowienia działania programu
- opóźnienie ekspedycji (ang. dispatch latency) - czas, jaki zajmuje ekspedytorowi wstrzymanie jednego procesu i uaktywnienie innego

Kryteria planowania

- **Wykorzystanie procesora -** procesor powinien być zajęty pracą
- **Przepustowość (ang. throughput) -** liczba procesów kończonych w jednostce czasu
- **Czas cyklu przetwarzania (ang. turnaround time) -** czas upływający między nadejściem procesu do systemu i zakończeniem przetwarzania procesu;
  suma czasów czekania na wejście do pamięci, czekania w kolejce procesów gotowych do wykonania, wykonania procesu przez procesor i wykonywania operacji we/wy
- **Czas oczekiwania -** suma okresów oczekiwania przez proces w kolejce procesów gotowych do wykonania (algorytm planowania ma wpływ na tę wielkość)
- **Czas odpowiedzi** (ang. response time) - czas upływający między wysłaniem żądania i pierwszą odpowiedzią

Algorytmy planowania. Pierwszy zgłoszony-pierwszy obsłużony (ang. first come- first served, FCFS)

- proces, który pierwszy zamawia procesor, pierwszy go otrzymuje
- wada: średni czas oczekiwania bywa b. długi
- Przykład:
- Proces Czas trwania fazy
- P1 24
- P2 3
- P3 3



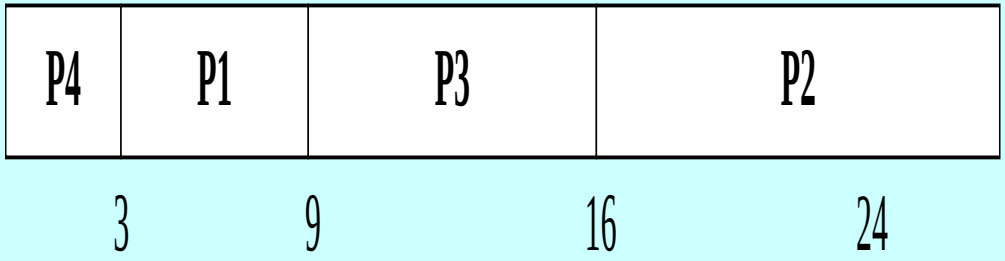
Algorytmy planowania. Pierwszy zgłoszony-pierwszy obsłużony (ang. first come- first served, FCFS) 2

- "nieszczęśliwa konfiguracja" - krótkie procesy muszą czekać na wykonanie długiego:
- Średni czas oczekiwania = (0+24+27) /3 = 17ms
- Dla "szczęśliwej konfiguracji" (odwrotna kolejność)
- Średni czas oczekiwania = (6+3+0) /3 = 3ms
- **Efekt konwoju (ang. convoy effect) -** wszystkie procesy czekają na zwolnienie procesora przez jeden wielki proces
- FCFS jest niewywłaszczający

Algorytmy planowania. "Najpierw najkrótsze zadanie" (ang. shortest job first , SJF)

- **z każdym procesem jest związana długość jego najbliższej fazy procesora**
- **wolny procesor zostaje przydzielony procesowi mającemu najkrótszą wstępną fazę**
- **jeżeli 2 procesy mają następne fazy procesora równej długości, wtedy stosuje się algorytm FCFS**
- Proces Czas trwania fazy
- P1 6
- P2 8
- P3 7
- P4 3
- **Średni czas oczekiwania = (3+16+ 9+0)/4 = 7ms**
- **Średni czas oczekiwania dla FCFS - 10.25ms**

Diagram Gantta:



Algorytmy planowania. "Najpierw najkrótsze zadanie" (ang. shortest job first , SJF) 2

- **Algorytm SJF jest optymalny -** daje minimalny średni czas oczekiwania dla danego zbioru procesów
- **Trudność w algorytmie:** określenie długości następnego zamówienia na przydział procesora
- w planowaniu długoterminowym może być to limit czasu procesu
- można szacować długość zamówienia
- Algorytm SJF może być **wywłaszczający** (gdy w kolejce pojawia się proces o krótszej następnej fazie procesora od tej, która została aktualnemu procesowi wykonywanemu), lub **niewywłaszczający**

Algorytmy planowania. "Najpierw najkrótsze zadanie" - realizacja przybliżona

- **Szacowanie długości następnej fazy procesora:**
- można przyjąć, że jej długość jest podobna do długości poprzednich faz procesora danego procesu:
- **Następna faza procesora:** średnia wykładnicza pomiarów długości poprzednich faz procesora

- $\tau_{n+1} = \alpha t_n + (1-\alpha)\ \tau_n$ (*)
- t_n - długość n-tej fazy procesora
- τ_{n+1} -przewidywana długość następnej fazy procesora
- τ_n - zawiera dane o minionej historii
- α - względna waga między niedawną i wcześniejszą historią
- $0 \leq \alpha \leq 1$,
- $\alpha = 1$ - uwzględnia się tylko ostatnie notowania wartości fazy (starsza historia jest pomijana),
- $\alpha = 0$ - niedawna historia nie ma wpływu na wynik (ostatnie
- wyniki miały charakter przejściowy),
- najczęściej przyjmuje się $\alpha = 0.5$ (jednakowa waga obu członów)
- wartość początkowa $\tau 0$ - stała lub średnia opisująca cały system
- Rozwinięcie wzoru **(*)**
- $\tau_{n+1} = \alpha t_n + (1-\alpha)\ \alpha\ t_{n-1} + \ldots + (1-\alpha)^j\ \alpha\ t_{n-j} + \ldots + (1-\alpha)^{n+1}\ \tau_0$

Planowanie priorytetowe

- każdemu procesowi przypisuje się priorytet
- procesor przydziela się temu procesowi, którego priorytet jest najwyższy
- w razie równych priorytetów - alg. FCFS

Proces	Czas trwania fazy	Priorytet
P1	10	3
P2	1	1
P3	2	3
P4	1	4
P5	5	2

- Kolejność:
- P2 P5 P1 P3 P4
- Średni czas oczekiwania: 8.2 ms

Planowanie priorytetowe 2

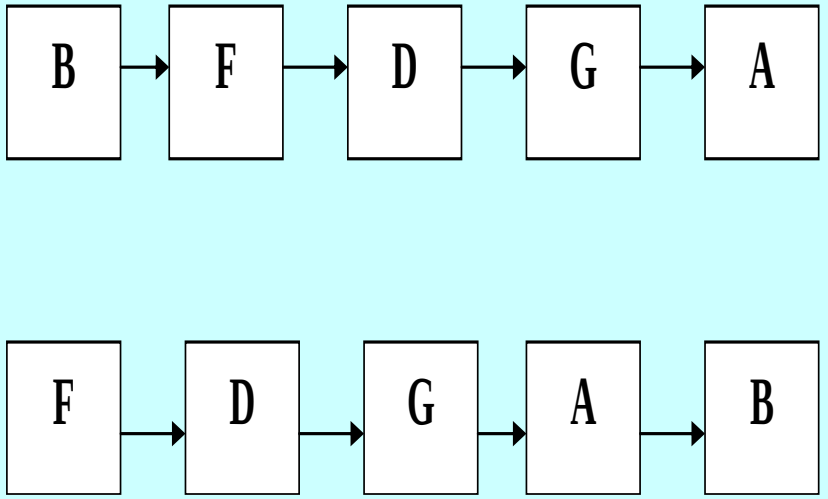
- Sposób definicji priorytetu:
  - wewnętrzny- używa się mierzalnej właściwości procesu (np. limity czasu, wielkość obszaru wymaganej pamięci, liczba otwartych plików, stosunek średniej fazy we/wy do średniej fazy procesora)
  - zewnętrzny- kryteria zewnętrzne wobec SO (np. ważność procesu, rodzaj i kwota opłat użytkownika, instytucja sponsorująca pracę itd.)
- Planowanie priorytetowe może być wywłaszczające lub niewywłaszczające
- Problem nieskończonego blokowania (ang. indefinite blocking), głodzenia (ang, starvation) - procesy niskopriorytetowe czekają w nieskończoność na procesor
- Rozwiązanie: postarzanie procesów niskopriorytetowych (ang. aging) - podwyższanie priorytetów długo oczekujących procesów

Planowanie rotacyjne (ang. round robin, RR)

- podobny do FCFS, dodano wywłaszczanie
- ustala się małą jednostkę czasu - kwant czasu (ang. time quantum) ; 10-100 ms
- kolejka procesów gotowych do wykonania - kolejka cykliczna
- każdy proces dostaje procesor na odcinek czasu, nie dłuższy od kwantu czasu, kiedy jest generowane przerwanie od zegara i proces jest usuwany na koniec kolejki procesów gotowych
- planista pobiera procesy w kolejności przyjścia FCFS
- Średni czas oczekiwania: dość wysoki

Planowanie rotacyjne 2

- Wydajność algorytmu mocno zależy od rozmiaru kwantu czasu
- Wpływ przełączania kontekstów na szybkość -> wskazane, by kwant czasu był długi w porównaniu z czasem przełączania kontekstu
- Jeśli kwant jest zbyt długi, to algorytm działa jak FCFS, co może prowadzić do nieakceptowalnie długiego oczekiwania na reakcję systemu
- Można przyjąć, że 80% faz procesora powinno być krótszych niż 1 kwant czasu



Planowanie gwarantowane (Guaranteed Scheduling) (1)

- realne obietnice szybkości przetwarzania dla użytkowników:
  - jeśli jest zalogowanych n użytkowników, to każdy użytkownik otrzymuje 1/n czasu procesora

- System musi dysponować następującymi informacjami:
  - jak wiele czasu CPU użytkownik otrzymał dla wszystkich swoich procesów od czasu zalogowania się
  - jak długo każdy użytkownik jest zalogowany
  - czas procesora przysługujący użytkownikowi jest ilorazem:
  - czas zalogowania użytkownika/ilość zalogowanych użytkowników

Planowanie gwarantowane (2)

- Określenie czasu przypadającego na proces/użytkownika
  - wyliczenie stosunku dotąd przyznanego czasu CPU do czasu przysługującego
    - współczynnik 0.5 oznacza, że użytkownik wykorzystał tylko połowę przysługującego mu czasu
    - współczynnik 2.0 oznacza, że użytkownik otrzymał dwa razy więcej czasu, niż mu przysługuje
    - proces jest uruchamiany z najniższym priorytetem, gdy jego współczynnik jest wyższy od współczynników innych procesów
- Systemy czasu rzeczywistego:
  - może być zastosowana podobna idea
  - procesy których `deadline` się zbliża otrzymują priorytety gwarantujące im pierwszeństwo wykonania

Planowanie loteryjne (ang. Lottery Scheduling)

- daje niezłe wyniki, przy znacznie prostszej od algorytmu **Guaranteed Scheduling** realizacji
- każdy proces dostaje los do rożnych rodzajów zasobów (np. właśnie CPU)
- podczas procesu planowania wybierany jest (metoda losową) los (bilet loteryjny ang. lottery tickets) i proces, który jest jego właścicielem dostaje zasób
- bardziej istotne procesy mogą mieć przyznaną większą ilość losów, co zwiększa ich szanse zwycięstwa w losowaniu

Planowanie loteryjne 2

- Właściwości planowania loteryjnego:
- planowanie loteryjne jest algorytmem bardzo czułym - jeżeli procesowi zwiększy się ilość losów, to jego szansa zwycięstwa już w następnym losowaniu zasobów wzrośnie
- współpracujące procesy mogą wymieniać losy między sobą np.
  - klient wysyła komunikat do procesu serwera i przechodzi w stan oczekiwania na wynik
  - może następnie przekazać swoje losy serwerowi co zwiększa szansę, że serwer zostanie szybko wykonany
  - po obsłużeniu zlecenia przez serwer, zwraca on losy klientowi,
  - klient zostaje uruchomiony
  - w przypadku braku klientów serwer nie potrzebuje losów

- Planowanie loteryjne - może być używane do rozwiązywania problemów, które trudno rozwiązać innymi metodami

Wielopoziomowe planowanie kolejek

- procesy są zaliczane do kilku grup np. procesy pierwszoplanowe (ang. foreground)- interakcyjne, oraz drugoplanowe (ang. background) - wsadowe.
- kolejka procesów gotowych jest rozdzielona na osobne kolejki
- w zależności od swych cech, procesy zostają na stałe przypisane do poszczególnych kolejek
- każda kolejka ma własny algorytm planujący (np. procesy pierwszoplanowe - alg. planowania rotacyjnego, procesy drugoplanowe - algorytm FCFS)
- konieczny jest algorytm planowania między kolejkami (np. stałopriorytetowe planowanie z wywłaszczaniem)
- inna możliwość - przyznanie każdej kolejce pewnego % czasu procesor

Planowanie wielopoziomowych kolejek ze sprzężeniem zwrotnym

- umożliwia przemieszczanie się procesów między kolejkami
- idea: rozdzielenie procesów o różnych rodzajach faz procesora
- proces, który zużywa za dużo czasu procesora, zostaje przeniesiony do kolejki o niższym priorytecie
- np., 3 kolejki, im zadanie ~w kolejce o niższym priorytecie, tym na dłużej może uzyskać procesor:
  - 1 - algorytm z kwantem 8
  - 2 - algorytm z kwantem 16
  - 3 - algorytm FCFS
- Parametry określające planistę wielopoziomowych kolejek ze sprzężeniem zwrotnym:
  - liczba kolejek
  - algorytm planowania dla każdej kolejki
  - metoda użyta do decydowania o awansie procesu do kolejki o wyższym priorytecie
  - metoda użyta do zdymisjonowania procesu do kolejki o niższym priorytecie
  - metoda wyznaczania kolejki, do której trafia proces

Planowanie: Polityka a mechanizmy

- generalnie przyjmuje się, że wszystkie procesy w systemie należą do różnych użytkowników i dlatego konkurują w dostępie do CPU

- w rzeczywistości zdarza się jednak, że jeden użytkownik uruchamia wiele procesów, które mogą współpracować ze sobą np.
  - proces uruchamia wielu potomków, którzy działają na jego rzecz np. system zarządzania bazą danych
  - każdy potomek pracuje nad innym zleceniem
  - proces macierzysty posiada wiedzę, na temat wagi zadań potomków, jednak generalnie algorytmy planowania nie wykorzystują tych związków między procesami
  - w konsekwencji planista rzadko podejmuje optymalne decyzje

- **rozwiązanie: separacja mechanizmów planowania i polityki planowania**
  - algorytmy planowania mogą być parametryzowane na różne sposoby, a parametry te są dobierane przez procesy użytkownika np.
  - przodek dysponuje funkcją systemową, która może zmieniać priorytety swoich potomków

Planowanie wieloprocesorowe

- Rozpatrywany przypadek: procesory identyczne (homogeniczne) pod względem wykonywanych funkcji

- Rozwiązania
  - aby uniknąć bezczynności procesorów przyjmuje się, że istnieje jedna wspólna kolejka procesów gotowych dla wszystkich procesorów
  - lub – aby zapobiec powstaniu wąskiego gardła (szczególnie istotne dla dużej liczby procesorów) każdy procesor ma własną kolejkę

Planowanie wieloprocesorowe 2

Dwie metody planowania:

- **każdy procesor sam planuje swoje działanie**
  - każdy procesor przegląda kolejkę procesów gotowych i wybiera proces do wykonania
  - konieczność właściwej realizacji obsługi kolejki procesów gotowych (np. nie dopuścić, by dwa procesory wybrały jednocześnie ten sam proces)
- **asymetryczne wieloprzetwarzanie (ang. asymmetric multiprocessing) wyróżniony jeden procesor - serwer główny**
  - serwer główny podejmuje wszystkie decyzje planistyczne, wykonuje operacje we/wy i czynności systemowe
  - pozostałe procesory wykonują tylko kod użytkowy

Planowanie wieloprocesorowe – Planowanie procesów

- Dobór konkretnego algorytmu planowania (RR, FCFS,...) traci na znaczeniu wraz ze wzrostem ilości procesorów

- W przypadku systemu z dużą ilością procesorów użycie FCFS lub priorytetowego (ze statycznymi priorytetami) może być wystarczające

Planowanie wieloprocesorowe – Planowanie wątków

- Różnica w porównaniu z planowaniem procesów:
  - Dla realizacji jednoprocesorowych: wątki ułatwiają strukturalizację programu oraz umożliwiają wykonywanie procesu nawet wtedy, gdy jakiś jego wątek czeka na operację we/wy
  - Dla realizacji wieloprocesorowych: Wątki wchodzące w skład jednego procesu zwykle blisko współpracują ze sobą, a zatem ich równoległe wykonywanie może znacząco zwiększać szybkość
    - Wątki nie muszą być blokowane w oczekiwaniu na wykonanie akcji przez inne, co powoduje wykonanie kosztowych funkcji systemowych oraz zmianę kontekstu wykonania na procesorach
    - Dla aplikacji wymagających znacznych interakcji między wątkami, niewielkie zmiany w zarządzaniu i szeregowaniu wątków mogą znacząco zmienić wydajność aplikacji

Planowanie wieloprocesorowe – Planowanie wątków (2)

- Podstawowe podejścia stosowane przy wieloprocesorowym planowaniu wątków:
  - Współdzielenie obciążenia (**Load Sharing**) – stosowana jest globalna kolejka gotowych do wykonania wątków, procesor gdy jest wolny wybiera wątek z kolejki
  - Szeregowanie grupowe (zespołowe) (**Gang scheduling**) – zbiór wątków tego samego procesu jest jednocześnie wykonywany na wielu procesorach
  - Rezerwacja procesorów (**Dedicated processor assignment**) – przydział wątków do wykonania na konkretnych procesorach
  - Dynamiczne szeregowanie (**Dynamic scheduling**) – ilość wątków w procesach może być zmieniana w trakcie wykonania

Planowanie wieloprocesorowe – Współdzielenie obciążenia

- Proste podejście, inspirowane w dużym stopniu środowiskami jednoprocesorowymi
- Zalety:
  - Obciążenie rozproszone równomiernie między procesory, gwarantuje, że żaden procesor nie pozostaje wolny, gdy jest praca do wykonania
  - Nie jest wymagany centralny planista, gdy procesor jest dostępny, procedura szeregująca systemu operacyjnego jest na nim wykonywana i dokonuje wyboru wątku
  - Globalna kolejka może być obsługiwana przy użyciu dowolnego algorytmu przedstawionego dla podejść jednoprocesorowych
- Wersje współdzielenia obciążenia
  - FCFS – kiedy zadanie przybywa, każdy z jego wątków jest kolejno umieszczany na końcu współdzielonej kolejki. Kiedy procesor jest bezczynny, wybiera on kolejny wątek, który wykonuje do chwili zakończenia lub zablokowania
  - Najpierw najmniejsza liczba wątków – kolejka gotowych wątków jest organizowana na zasadzie priorytetowej, najwyższy priorytet otrzymują wątki zadań o najmniejszej liczbie gotowych do wykonania wątków, zadania o jednakowym priorytecie są szeregowane według kolejności przybycia, przydzielony wątek jest wykonywany do chwili zakończenia lub zablokowania
  - Najpierw najmniejsza liczba wątków z wywłaszczeniem -- jak dla poprzedniego przypadku, ale przybywający proces z mniejszą ilością gotowych do wykonania wątków, niż aktualnie wykonywany, wywłaszcza wykonujące się wątki
    W [Stallings] stwierdzono, że przeprowadzone badania symulacyjne wykazały, że FCFS jest lepszy od pozostałych polityk, ale generalnie gorszy od szeregowania grupowego

Planowanie wieloprocesorowe – Współdzielenie obciążenia (2)

- Wady współdzielenia obciążenia:
  - Dostęp do centralnej kolejki należy uzyskiwać stosując wzajemne wykluczenie. Dla wielu procesorów (dziesiątek, setek) kolejka może się stać wąskim gardłem.
  - Wywłaszczane wątki rzadko wznawiają wykonanie na tym samym procesorze. Jeżeli każdy procesor jest wyposażony w lokalną pamięć podręczną, buforowanie staje się mniej efektywne
  - Jeżeli wszystkie wątki traktuje się jak wspólną pulę, mała jest szansa, że wszystkie wątki danego procesu uzyskają jednocześnie dostęp do procesorów, co przy silnych związkach koordynacyjnych między nimi może znacząco obniżyć wydajność
- Pomimo potencjalnych wad, współdzielenie obciążenia jest jedną z najczęściej stosowanych metod planowania we współczesnych systemach wieloprocesorowych

Planowanie wieloprocesorowe – Szeregowanie grupowe

- Jednoczesne szeregowanie wątków tworzących jeden proces
- Bardzo korzystne w przypadku aplikacji równoległych, których wydajność znacznie spada, gdy część aplikacji nie działa razem z innymi
- Przydatne także w przypadku aplikacji mniej wrażliwych na kwestie wydajnościowe
- Szeregowanie grupowe minimalizuje przełączenia procesów (wątki nie muszą oddawać procesora w oczekiwaniu na działania np. zwolnienia muteksów, wysłanie danych ze strony innych wątków)
- Sposoby przydziału procesorów:
  - Zakładamy, że mamy N procesorów i M aplikacji, z których każda składa się z co najwyżej N wątków
  - (1) Każdej aplikacji można przydzielić 1/M czasu na N procesorach
    Taka strategia może być nieefektywna, jeśli mamy różne ilości wątków w procesach, wtedy dobrze by było uruchomić na wolnych procesorach aplikacje o mniejszych ilościach wątków (np. jednowątkowe)
  - (2) Można dzielić czas używając szeregowania ważonego według liczby wątków np. proces i z M_i wątków może uzyskać $M_i / \sum_{j=1...M} M_j$

Planowanie wieloprocesorowe – Rezerwacja procesorów

- Skrajna forma szeregowania grupowego polegająca na zarezerwowaniu grupy procesorów na wyłączny użytek aplikacji przez czas jej trwania
- Gdy aplikacja zostaje skierowana do wykonania, każdemu z jej wątków przydziela się procesor, który pozostaje dla niego zarezerwowany do chwili ukończenia aplikacji
- Ocena
  - Wada:
    - Podejście wydaje się być bardzo rozrzutnym z punktu widzenia użytkowania procesorów – nie ma wieloprogramowania procesorów, wątek zablokowany na we/wy nadal zajmuje procesor
  - Argumenty „za”:
    - (1) W systemie z wysokim stopniem paralelizmu z dziesiątkami lub setkami procesów, z których każdy stanowi niewielki koszt systemu, wykorzystanie procesora nie jest tak kluczowe z punktu widzenia efektywności czy wydajności
    - (2) Całkowity brak przełączania procesów podczas życia danego procesu powinien doprowadzić do jego znacznie szybszego wykonania
      Przeprowadzone eksperymenty dowiodły słuszność stwierdzenia (2) → efektywną strategią jest ograniczenie liczby aktywnych wątków do liczby procesorów w systemie
- Analogie przydziału procesorów do przydziału stron. Określenie rozmiaru „zbioru roboczego aktywności” (activity working set) na oznaczenie minimalnej liczby wątków, których potrzebuje aplikacja, by czynić postępy

Planowanie wieloprocesorowe –

Dynamiczne szeregowanie (1)

- Często ilość wątków w procesie może się zmieniać
- System operacyjny mógłby regulować ilość wątków, a w konsekwencji obciążenie systemu, aby zoptymalizować stopień wykorzystania procesora
- Propozycja: zarówno system operacyjny jak i aplikacja uczestniczą w podejmowaniu decyzji o szeregowaniu
  - System operacyjny odpowiada za rozdział procesorów między procesy
  - Każdy proces używa przydzielonych mu procesorów, aby wykonać pewien zbiór czynności mapując je na wątki (decyzje, które wątki zawiesić, a które wykonać pozostają w gestii zależą od aplikacji, wspomaganej przez zbiory procedur bibliotecznych)

Planowanie wieloprocesorowe – Dynamiczne szeregowanie (2)

- Zadania systemu operacyjnego ograniczają się do przydziału procesorów procesom i postępowania jak powyżej:
  1. Jeżeli są bezczynne procesory – użyć ich do spełnienia zadania
  2. W przeciwnym razie, jeżeli żądanie zostało zgłoszone przez nowo przybyłe zadanie, przydzielić mu jeden procesor, zabierając go zadaniu, które obecnie dysponuje więcej niż jednym procesorem
  3. Jeśli nie można spełnić dowolnej części żądania, żądanie czeka, aż zwolni się dla niego procesor albo zadanie wycofa żądanie
  4. Po zwolnieniu jednego lub kilku procesorów (lub zakończeniu zadania) :
     - Przeskanować kolejkę niezrealizowanych żądań przydziału procesora. Przydzielić procesor każdemu zadaniu na liście, które obecnie nie ma procesora (tzn. wszystkim oczekującym, nowo przybyłym zadaniom),
     - Ponownie przeskanować listę przydzielając resztę procesorów na zasadzie – kto pierwszy ten lepszy)
- Przeprowadzone analizy sugerują zalety metody (lepsza niż szeregowanie zespołowe i rezerwacja procesorów) w przypadku aplikacji umożliwiających wykorzystanie szeregowania dynamicznego, koszty dodatkowe mogą jednak zanegować te korzyści

Planowanie w czasie rzeczywistym 1

- **Planowanie w rygorystycznych systemach czasu rzeczywistego**
  - proces jest dostarczany wraz z przepisem określającym wymaganą ilość czasu do jego zakończenia (lub wykonania operacji we/wy)
  - rezerwacja zasobów (ang. resource reservation) - planista akceptuje proces i zapewnia jego wykonanie na czas lub odrzuca zlecenie jako niewykonalne
  - planista musi mieć dokładne rozeznanie, ile czasu zużywają poszczególne funkcje systemu operacyjnego - dla każdej operacji systemowej musi być przypisany maksymalny czas jej wykonania (taka gwarancja - niemożliwa w systemach z pamięcią wirtualną lub pomocniczą)

Planowanie w czasie rzeczywistym 2

- **Planowanie w łagodnych systemach czasu rzeczywistego**
  - procesy o decydującym znaczeniu muszą mieć wyższy priorytet od innych procesów
  - dodanie do systemu takich uprzywilejowanych procesów może powodować niesprawiedliwy podział zasobów, większe opóźnienia innych procesów i ich głodzenie

- **Realizacja planisty:**
  - system musi mieć planowanie priorytetowe,
  - procesy działające w czasie rzeczywistym muszą mieć najwyższy priorytet
  - priorytety procesów czasu rzeczywistego nie mogą maleć z upływem czasu
  - opóźnienie ekspediowania procesów do procesora musi być małe - istotnym problemem jest to, że w większości systemów Unix przed przełączeniem kontekstu należy poczekać do zakończenia wykonania funkcji systemowej lub zablokowania procesu w oczekiwaniu na operację we/wy

Planowanie w czasie rzeczywistym 3

- zezwolenie na wywłaszczanie funkcji systemowych
- wstawienie do długotrwałych funkcji systemowych tzw. punktów wywłaszczeń (ang. preemption points), gdzie sprawdza się. czy nie ma wysokopriorytetowych procesów wymagających uaktywnienia
- punkty wywłaszczeń - tylko w "bezpiecznych miejscach" - gdzie nie są zmieniane struktury danych jądra
- wywłaszczanie całego jądra - struktury danych jądra muszą być chronione przez specjalne mechanizmy synchronizacyjne

- **odwracanie priorytetów** (ang. priority inversion)
  - wysokopriorytetowy proces, chcący odwołać się do danych jądra. które są używane przez proces niskopriorytetowy musiałby czekać na ich zwolnienie
  - stosowanie protokołu dziedziczenia priorytetów (ang. priority-inheritance protocol) - wszystkie procesy dziedziczą wysoki protokół do czasu zakończenia korzystania przez nie z zasobów na które czeka proces wysokopriorytetowy

Proste algorytmy szeregowania – zestawienie (1)

- FCFS – First Come First Served
- RR – Round Robin
- SPN – Shortest Process Next (SJF)
- SRT – Shortest Remaining Time (wywłaszczalna wersja SJF)
- HRRT – Highest Response Ratio Next
- Feedback - karanie procesów, które się długo wykonywały

Proste algorytmy szeregowania – zestawienie (2)

	FCFS	Round robin	SPN	SRT	HRRN	Feedback
Selection function	max[w]	constant	min[s]	min[s − e]	$\max\left(\frac{w+s}{s}\right)$	(see text)
Decision mode	Non-preemptive	Preemptive (at time quantum)	Non-preemptive	Preemptive (at arrival)	Non-preemptive	Preemptive (at time quantum)
Throughput	Not emphasized	May be low if quantum is too small	High	High	High	Not emphasized
Response time	May be high, especially if there is a large variance in process execution times	Provides good response time for short processes	Provides good response time for short processes	Provides good response time	Provides good response time	Not emphasized
Overhead	Minimum	Minimum	Can be high	Can be high	Can be high	Can be high
Effect on processes	Penalizes short processes; penalizes I/O bound processes	Fair treatment	Penalizes long processes	Penalizes long processes	Good balance	May favor I/O bound processes
Starvation	No	No	Possible	Possible	No	Possible

Funkcja wyboru procesu

- Określa, jaki proces jest wybrany do wykonania
- Jeśli jest oparta na charakterystykach wykonania, rozważane są następujące parametry:
  - ***w*** = dotychczasowy czas spędzony w systemie
  - ***e*** = dotychczasowy czas wykonania
  - ***s*** = całkowity wymagany czas wykonania, obejmujący *e*;

Ocena algorytmów planowania - wprowadzenie

- problem zdefiniowania kryterium wyboru algorytmu np.
  - maksymalizacja wykorzystania procesora przy założeniu, że maksymalny czas odpowiedzi wyniesie 1s,
  - maksymalizacja przepustowości tak, by czas cyklu przetwarzania był średnio liniowo proporcjonalny do ogólnego czasu wykonania

Ocena algorytmów planowania - modelowanie deterministyczne

- przyjmuje się konkretne obciążenie systemu i definiuje się zachowanie każdego algorytmu w warunkach tego obciążenia
- proste i szybkie
- daje wyniki pozwalające na porównanie algorytmów
- wymaga dokładnych liczb na wejściu, a odpowiedzi odnoszą się tylko do zadanych przypadków

Ocena algorytmów planowania - modele obsługi kolejek

- procesy w systemie zmieniają się
- badania dotyczą ustalenia rozkładów faz procesora i we/wy, ich mierzenia i szacowania, co w rezultacie może dać wzór opisujący prawdopodobieństwo wystąpienia poszczególnych faz procesora (zwykle - rozkład wykładniczy) (*)
- inne badania - ustalenie rozkładu czasów przybywania procesów do systemu (**)
- na podstawie (*) i (**) można obliczyć średnią przepustowość, wykorzystanie procesora, czas oczekiwania itd.
- system komputerowy: sieć usługodawców (procesorów i urządzeń we/wy), z których każdy posiada własną kolejkę czekających procesów (analiza obsługi kolejek w sieciach, ang. queuing-network analysis)

- **Wzór Little'a**

  $n = \lambda \times W$

  n - średnia długość kolejki

  λ **-** tempo przybywania nowych procesów do kolejki (ilość/s)

  W - średni czas oczekiwania w kolejce

  trudność matematycznego opracowania algorytmów

Ocena algorytmów planowania - symulacje

- symulacje w oparciu o implementacje modelu systemu komputerowego
- dane do sterowania symulacją można uzyskać poprzez:
- generator liczb losowych do tworzenia procesów, określania czasów trwania faz procesora itd.
- rozkłady można definiować matematycznie lub doświadczalnie (pomiary w badanym systemie)
- zastosowanie taśm śladów (ang. trace tapes), które powstają w wyniku nadzorowania rzeczywistego systemu i zapisywania rzeczywistej kolejności zdarzeń

Ocena algorytmów planowania - implementacja

- ostateczna weryfikacja - implementacja algorytmu, zastosowanie go w rzeczywistym systemie i analiza uzyskanych wyników

- trudność - koszt (zakodowanie algorytmu, włączenie do systemu, reakcja użytkowników), zmiana środowiska (nowe programy, zmiana zachowań użytkowników)