Systemy czasu rzeczywistego

Mariusz Żynel
mariusz@math.uwb.edu.pl
http://math.uwb.edu.pl/~mariusz/

Uniwersytet w Białymstoku

2024/2025

Parametry obciążenia w systemach czasu rzeczywistego

Założenia

- Zakładamy, że parametry twardych prac i zadań są znane w każdym momencie
- W przeciwnym razie nie można zagwarantować, że system spełni ograniczenia czasowe

Ilość zadań

- W twardych systemach (np. w systemach wbudowanych) liczba zadań jest stała
- Ilość zadań może ulec zmianie w przypadku zmiany trybu pracy
- Ilość zadań w nowym trybie pracy jest również ustalona
- W przykładzie kontrolera lotu helikoptera mamy 12 zadań:
  - dwa 30Hz zadania awioniki
  - trzy 30Hz obliczenia reguły kontroli
  - dwa 90hz obliczenia reguły kontroli
  - jedno 180Hz obliczenie reguły kontroli
  - jedno zadanie walidacji
  - jedno zadanie operacji wyjścia
  - jedno zadanie wbudowanych testów
- W trybie lądowania wszystkie obliczenia reguły kontroli są powielane trzykrotnie, mamy więc razem 24 zadania
- W systemach miękkich ilość zadań może ulegać zmianie, ale ilość zadań twardych jest stała
- W systemie kontroli ruchu powietrzenego zadanie monitorowania samolotu jest dodawane, gdy wchodzi on w obszar zasięgu i usuwane, gdy z niego wychodzi

Parametry pracy

czasowe	określają ograniczenia czasowe pracy i jej zachowanie w czasie
funkcjonalne	określają wewnętrzne właściwości pracy
zasobów	określają wymagania pracy co do zasobów
zależności	opisują jak dana praca zależy od innych oraz jak inne prace zależą od danej pracy

Parametry obciążenia w systemach czasu rzeczywistego

Parametry czasowe

Stałe, niestabilne i sporadyczne czasy uwolnienia

- W wielu systemach nie znamy dokładnego momentu r_i uwolnienia pracy J_i
- Określony jest wówczas przedział *wahania czasu uwolnienia* (release-time jitter) $[r_i^-, r_i^+]$ do którego należy r_i
- Gdy wahania te są nieznaczne w stosuku do pozostałym parametrów czasowych, to można przyjąć, że czas uwolnienia to najwcześniejszy r_i^- lub najpóźniejszy r_i^+ czas uwolnienia

- Prawie każdy system czasu rzeczywistego musi reagować na zdarzenia zewnętrzne, które występują w losowych momentach czasu
- Gdy takie zdarzenie wystąpi, system w odpowiedzi wykonuje pewną pracę
- Czas uwolnienia takiej pracy nie jest znany, dopóki nie nastąpi zdarzenie wyzwalające
- Takie prace nazywane są pracami *sporadycznymi* lub *aperiodycznymi*

Czas wykonania

- *Czas wykonania* (execution time) e_i pracy J_i to ilość czasu potrzebna do ukończenia tej pracy, gdy jest ona wykonywana osobno i posiada wszystkie wymagane zasoby
- Wartość tego parametru zależy od złożoności pracy i szybkości procesora, a nie od sposobu zaplanowania zadania
- Faktyczny czas wykonania może się wahać z wielu powodów (przewidywalność systemów)
- Na przykład: rzeczywisty czas transmisji klatki skompresowanego wideo MPEG zależy od ilości bitów użytych do zakodowania jej
- W wielu wypadkach można jedynie wyznaczyć *minimalny czas wykonania* e_i^- oraz *maksymalny czas wykonania* e_i^+ pracy J_i podając przedział $[e_i^-, e_i^+]$ zmienności e_i
- W modelach deterministycznych, podawany jest maksymalny czas wykonania
- W systemach czasu rzeczywistego, twarde zadania zwykle stanowią mniejszą część, w ten sposób zaalokowane, ale nie użyte, czas procesora oraz zasoby, można przeznaczyć na zadania miękkie i zwykłe

Model zadań okresowych

- Dobrze znany deterministyczny model obciążenia (workload)
- Model ten dokładnie charakteryzuje wiele tradycyjnych, twardych aplikacji czasu rzeczywistego:
  - sterowanie cyfrowe
  - monitorowanie w czasie rzeczywistym
  - transmisja audio/video o stałym bit-rate
- Algorytmy planowania (scheduling) oparte na tym modelu mają wysoką wydajność
- Istnieją metody i narzędzia wspierające projektowanie, analizę i walidację systemów czasu rzeczywistego opartych na tym modelu

Okres i czas wykonania zadań okresowych

- *Zadanie okresowe* (*periodic task*) to każde obliczenie lub transmisja danych, która jest wykonywana wielokrotnie w regularnych lub półregularnych odstępach czasu w celu zapewnienia ciągłej funkcjonalności systemu
- Zadanie okresowe T_i to ciąg prac
- *Okres* (*period*) p_i zadania okresowego T_i to minimalna długość wszystkich odstępów czasu między momentami uwolnienia (release time) następujących po sobie prac w T_i
- *Czas wykonania* (*execution time*) zadania okresowego T_i to maksimum czasów wykonania wszystkich prac w T_i
- Okres i czas wykonania każdego zadania okresowego w sytemie są zawsze znane
- Zadanie okresowe jest *prawdziwie okresowe* (*truly periodic*), gdy czas pomiędzy kolejnymi momentami uwolnienia (release time) jest równy okresowi
- Dokładność modelu zadań okresowych maleje wraz ze wzrostem wahań w momentach uwolnienia oraz zmianami w czasach wykonania
- Na przykład: model okresowy jest niewłaściwy przy transmisji wideo o zmiennym bit-rate z uwagi na zmienny czas wykonania, czyli czas transmisji poszczególnych klatek

Zadania aperiodyczne i sporadyczne

- System czasu rzeczywistego reaguje na zdarzenia zewnętrzne, a w odpowiedzi na nie wykonuje *prace aperiodyczne* (*aperiodic jobs*) lub *prace sporadyczne* (*sporadic jobs*), których czasy uwolnienia nie są znane
- Na przykład:
  - System radarowy musi działać nieprzerwanie cały czas, ale dodatkowo musi również reagować na polecenia operatora: na zmianę czułości radaru, progu wykrywalności itp.
  - Gdy pilot zmienia ustawienia autopilota z trybu przelotowego na tryb gotowości, system musi zareagować, rekonfigurując się, jednocześnie kontynuując wykonywanie zadań kontrolnych, które sterują samolotem
- *Zadanie aperiodyczne* (*aperiodic task*) to ciąg prac aperiodycznych
- *Zadanie sporadyczne* (*sporadic task*) to ciąg prac sporadycznych
- Prace w każdym zadaniu modelują robotę wykonywaną przez system w odpowiedzi na zdarzenia tego samego typu
- Zadanie jest aperiodyczne, jeżeli prace w nim zawarte mają albo miękkie, albo nie mają żadnych ograniczeń czasowych
- Zadanie jest sporadyczne, jeżeli prace w nim zawarte mają twarde ograniczenia czasowe

Parametry obciążenia w systemach czasu rzeczywistego

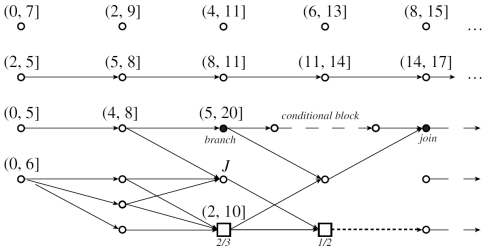
Parametry zależności

Zależności kolejności wykonania

- Prace są *zależne* (*have precedence constraints*), gdy muszą być wykonane w odpowiedniej kolejności, w przeciwnym razie prace są *niezależne* (*independent*)
- Na przykład: w systemie nadzoru radarowego zadanie przetwarzania sygnału produkuje rekordy śledzenia, podczas gdy zadanie śledzenia konsumuje je (*zagadnienie producent-konsument*). W szczególności każda praca śledzenia przetwarza rekordy śledzenia generowane przez pracę przetwarzania sygnału. Projektant systemu może zdecydować się na synchronizację zadań tak, aby wykonywanie k-tej pracy śledzenia nie rozpoczęło się, dopóki k-ta praca przetwarzania sygnału nie zostanie ukończona.

Zależności kolejności wykonania, graf poprzedzania

- Na zbiorze prac określamy relację *poprzedzania* (*precedence*), w ten sposób, że praca J_i poprzedza pracę J_k, co zapisujemy jako $J_i < J_k$, gdy wykonanie pracy J_k nie może się rozpocząć przed ukończeniem wykonania pracy J_i
- Zależności między pracami reprezentuje skierowany *graf poprzedzania* (*precedence graph*), którego wierzchołkami są prace i od pracy J_i biegnie skierowana krawędź do pracy J_k, gdy J_i jest bezpośrednim poprzednikiem J_k
- Zależności w systemie czasu rzeczywistego przedstawia się za pomocą *grafu zadań* (*task graph*), który jest rozszerzeniem grafu poprzedzania



Graf zadań

Kolejność dostępu do danych

- Zależności w dostępie do danych nie da się wyrazić za pomocą grafu poprzedzania
- W wielu systemach prace komunikują się za pośrednictwem współdzielonych obszarów pamięci. Projektant systemu nie musi synchronizować prac producenta i konsumenta. Zamiast tego producent umieszcza wygenerowane przez siebie dane we współdzielonej przestrzeni adresowej, aby mogły być używane przez konsumenta w dowolnym momencie. Graf poprzedzania w tym wypadku pokazuje, że producent i konsument są niezależni.
- Na przykład: w awionice samolotu praca nawigacyjna okresowo aktualizuje lokalizację samolotu i umieszcza odpowiednie dane w przestrzeni współdzielonej. Gdy praca kontroli lotu potrzebuje danych nawigacyjnych, odczytuje najnowsze dane z bufora. Kolejność prac nawigacji oraz kontroli lotu są tutaj niezależne.

Zależność czasowa

- Niektóre prace muszą być ukończone w określonym czasie względem siebie
- Różnicę pomiędzy momentami ukończenia dwóch prac nazywamy *odległością czasową* (*temporal distance*) między nimi
- Na przykład: rozważymy wyświetlanie klatek wideo i towarzyszącego im dźwięku, gdy wideo przedstawia osobę mówiącą. Aby uzyskać synchronizację ust, czas między wyświetlaniem każdej klatki a wygenerowaniem odpowiadającego jej segmentu audio nie może być dłuższy niż 160ms

Parametry obciążenia w systemach czasu rzeczywistego

Parametry funkcjonalne

Wywłaszczanie

- Planista (scheduler) może zawiesić wykonywanie mniej pilnej pracy i przekazać procesor bardziej pilnej pracy. Później, gdy bardziej pilna praca zostanie ukończone, planista przywraca procesor mniej pilnej pracy, aby mogła ona wznowić swoje wykonywanie. To przerwanie wykonywania pracy nazywa się *wywłaszczeniem* (*preemption*).
- Praca która jest *niewywłaszczalna* (*nonpreemptable*) musi być wykonywana od początku do końca bez przerw
- Na przykład:
  - Podczas pakietowej transmisji danych, jeśli przesyłanie zostanie wstrzymane w trakcie przekazywania pakietu, odbiorca odrzuci częściowo odebrany pakiet
  - Zachowanie stanu CPU, jego rejestrów, licznika instrukcji danego procesu w momencie wywłaszczania albo przerwania, tak aby można było ten proces wznowić, nstępnie przygotowanie środowiska do wykonania innego procesu to *przełączanie kontekstowe* (*context switching*). Ta praca CPU nie może być przerwana i dlatego nie jest wywłaszczalna.

Krytyczność

- W każdym systemie, różne prace nie są tak samo ważne
- *Krytyczność* (*criticality*) pracy to liczba dodatnia, która wskazuje, jak krytyczna (ważna) jest ta praca w stosunku do innych (priorytet)
- W przypadku przeciążenia, gdy nie jest możliwe wykonanie wszystkich prac tak, aby dotrzymać terminów, może mieć sens poświęcenie mniej krytycznych prac, aby prace bardziej krytyczne mogły dotrzymać terminów
- Na przykład: w systemie sterowania i zarządzania lotem praca, która kontroluje lot samolotu, jest ważniejsze niż praca nawigacyjna, która określa bieżącą pozycję względem wybranego kursu. Z kolei praca nawigacyjna jest ważniejsze niż praca, która dostosowuje kurs i prędkość przelotową w celu zminimalizowania zużycia paliwa. Prace związane z przepływem powietrza w kabinie i kontrolą temperatury są ważniejsze niż prace, które uruchamiają filmy w trakcie lotu, itd.

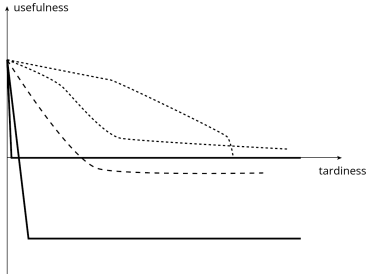
Opcjonalne wykonanie

- Często możliwe jest zaprojektowanie aplikacji tak, aby niektóre prace lub ich części były opcjonalne
- Jeśli opcjonalna praca lub opcjonalna część pracy zostanie ukończona z opóźnieniem lub w ogóle nie zostanie wykonana, wydajność systemu może się pogorszyć, ale mimo to będzie on działać zadowalająco
- Podczas przejściowego przeciążenia systemu, gdy nie jest możliwe ukończenie wszystkich prac na czas, system może zdecydować się na odrzucenie prac opcjonalnych
- System idzie na kompromis pomiędzy jakością wyników, które generuje i usług, które dostarcza, na rzecz terminowości swoich wyników i usług
- Na przykład: w systemie unikania kolizji możemy uznać pracę obliczania poprawnej akcji unikowej i informowania operatora o tej akcji za opcjonalne. Zwykle chcemy, aby system pomagał operatorowi, wybierając poprawną akcję. Jednak w przypadku gdy system działa w trybie zdegradowanym, nie jest możliwe ukończenie tego obliczenia na czas. System unikania kolizji może nadal działać zadowalająco, gdy pominie to obliczenie, o ile wygeneruje ostrzeżenie i wyświetli kurs obiektu, z którym mamy się zderzyć.

Laxity

- Parametr *laxity* wskazuje, czy ograniczenia czasowe są miękkie czy twarde
- Czasem ten parametr uzupełniany jest przez funkcję *użyteczności* (*usefulness*), która określa użyteczność wyniku wygenerowanego przez pracę jako funkcję jego opóźnienia

- Linie ciągłe: twarde prace. Użyteczność wyniku staje się zerowa lub ujemna jak tylko praca jest opóźniona. Lepiej nie wykonać pracy niż wykonać ją i zakończyć z opóźnieniem (better never than late).
- Linie kropkowane: miękkie prace, użyteczność spada łagodnie. Na przykład: sprawdzenie stanu konta w punkcie sprzedaży. W miarę upływu czasu i braku wyniku klient oraz sprzedawca stają się coraz brdziej niecierpliwi.
- Linia przerywana: miękka praca, użyteczność spada gwałtownie. Na przykład: aktualizacja ceny akcji na giełdzie. Lekko spóźniony wynik może być do przyjęcia, jeśli jednak aktualizacja zakończy się tak późno, że nowa cena znacznie różni się od poprzedniej, wynik będzie mylący.



Funkcje użyteczności

Parametry obciążenia w systemach czasu rzeczywistego

Parametry zasobów

Parametry zasobów przypisane do pracy

- Każda praca, aby być wykonana wymaga procesora, a często również zasobów
- Parametry zasobów przypisane do pracy określają:
  - typ procesora
  - typy zasobów wymaganych przez pracę
  - przedziały czasowe podczas jej wykonywania, gdy zasoby będą wymagane
- Powyższe parametry konieczne są do podejmowania decyzji w zarządzaniu zasobami

Wywłaszczanie zasobów

- Parametry zasobów charakteryzują procesory i zasoby niezależnie od aplikacji
- Gdy jednostka zasobu zostanie przydzielona do pewnej pracy, a inne prace potrzebujące tej jednostki muszą czekać, aż praca zakończy użycie tego zasobu, to taki zasób jest *niewywłaszczalny* (*nonpreemptable*)
- W przeciwnym razie, jeśli prace mogą używać każdej jednostki zasobu w sposób przeplatany, zasób jest *wywłaszczalny* (*preemptable*)

- Zasoby wywłaszczalne:
  - CPU (ogólnie procesor)
  - pamięć
- Zasoby niewywłaszczalne:
  - drukarka
  - blokada (lock)

- W przykładzie pakietowej transmisji danych, modelowanie pracy przesyłania pakietu danych jako niewywłaszczalnej jest niepoprawne. Lepszą alternatywą jest modelowanie łącza danych jako zasobu niewywłaszczalnego i pozostawianie transmisji pakietu jako wywłaszczalnej. Fakt, że przerwana transmisja pakietu musi zostać ponowiona, jest konsekwencją sposobu działania protokołu dostępu do łącza danych.

```
> FIN < ACK < FIN > ACK

init 5
```