Systemy czasu rzeczywistego

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

Uniwersytet w Białymstoku

2024/2025

Przewidywalność systemów komputerowych

Przewidywalność systemów komputerowych

- Najważniejszą cechą systemu czasu rzeczywistego jest *przewidywalność*
- System powinien być w stanie:
  - przewidywać jak zadania będą ewoluować
  - z góry zagwarantować spełnialność wszystkich ograniczeń czasowych
- Spełnialność ograniczeń czasowych zależy od wielu czynników sprzętowych i programowych
- Wewnętrzne cechy CPU jak: *instruction prefetch*, *pipelining*, *cache memory* poprawiają wydajność, ale powodują *niedeterminizm* systemu

Przewidywalność: Direct Memory Access - DMA

- Urządzenia peryferyjne przesyłają dane bezpośrednio do pamięci głównej RAM
- Odciążenie CPU od kontroli transferu wejścia/wyjścia
- CPU i urządzenie DMA współdzielą szynę danych (*cycle stealing*)
- Transfer wejścia/wyjścia oraz program na CPU wykonywane są równolegle
- Gdy urządzenie DMA i CPU sięgają do pamięci w tym samym czasie, szyna danych przypisywana jest do urządzenia DMA, a CPU oczekuje do następnego cyklu
- Nie jesteśmy w stanie przewidzieć jak długo CPU będzie czekać na urządzenie DMA
- Nie jesteśmy w stanie precyzyjnie określić czasu reakcji zadania (response time)
- Możliwym rozwiązaniem jest zastosowanie techniki *time-slice*, w której jeden cykl dostępu do pamięci dzielony jest na dwa sąsiednie przedziały czasowe: jeden dla CPU, drugi dla urządzenia DMA

Przewidywalność: pamięć podręczna (cache)

- Szybka pamięć wstawiana jako bufor pomiędzy CPU i pamięć RAM
- Pamięć podręczna nie jest widziana z poziomu programu użytkownia
- Po ustaleniu adresu lokalizacji fizycznej pamięci sprawdzane jest, czy żądane informacje są przechowywane w pamięci podręcznej: jeśli tak, dane są odczytywane z pamięci podręcznej, w przeciwnym razie informacje pobierane są z pamięci RAM i kopiowane do pamięci podręcznej wraz z sąsiednimi regionami
- Satystycznie przy 1MB RAM i 8kB cache, dane żądane przez program w 80 przypadkach na 100 znajdują się w pamięci podręcznej (*program locality*, *hit ratio*)
- Obserwacje statystyczne pozwalają jedynie na oszacowanie przeciętnego zachowania aplikacji, ale nie mogą służyć do wyznaczania czasu reakcji (response time)

Przewidywalność: przerwania (interrupts)

- W systemie operacyjnym nadejście sygnału przerwania powoduje wykonanie procedury sterownika (*driver*) dedykowanej do zarządzania powiązanym z nim urządzeniem
- Na ogół przerwania są obsługiwane na podstawie ustalonego schematu priorytetów: sterowniki mają statyczny priorytet, wyższy niż priorytety zwykłych procesów, gdyż sterowniki mają ograniczenia czasu rzeczywistego, podczas gdy aplikacje ich nie mają
- Przerwania generowane przez urządzenia peryferyjne, jeśli nie są odpowiednio obsłużone, mogą wprowadzać nieograniczone opóźnienia w trakcie wykonywania programu
- Ilość przerwań oraz opóźnienie wykonania zadania nimi spowodowane są nieprzewidywalne
- Możliwe rozwiązania:
  - Wszystkie zewnętrzne przerwania, poza zagarem (*timer*), są wyłączone, urządzenia obsługiwane są przez aplikację, transfer danych przez odpytywanie (*polling*) (busy waiting)
  - Wszystkie zewnętrzne przerwania, poza zagarem (*timer*), są wyłączone, urządzenia obsługiwane są przez dedykowane moduły jądra okresowo aktywowane zegarem
  - Zewnętrzne przerwania są włączone, sterowniki ograniczają się do aktywowania odpowiednich zadań znajdujących się pod kontrolą systemu operacyjnego dzięki czemu priorytety tych zadań mogą być ustawiane w zależności od wymagań aplikacji

Przewidywalność: wywołania systemowe (system calls)

- Aby precyzyjnie określić czas wykonania zadania w najgorszym przypadku, powinien być znany czas realizacji każdego wywołania systemowego (*system call*)
- Wywołanie systemowe, którego nie można wywłaszczyć (przejąć), może opóźnić aktywację lub wykonanie krytycznych zadań
- Wszystkie wywołania systemowe powinny być wywłaszczalne (*preemptable*)

Przewidywalność: semafory

- Typowy machanizm semaforów nie nadaje się do użycia w implementacji aplikacji czasu rzeczywistego z uwagi na możliwość wystąpienia *odwrócenia priorytetów* (*priority inversion*)
- Odwrócenie priorytetów polega na tym, że zadanie o wysokim priorytecie czeka na zadanie o niskim priorytecie
- W wyniku odwrócenia priorytetów powstają niedeterministyczne opóźnienia wykonania zadań, co czyni system nieprzewidywalnym
- Rozwiązaniem jest stosowanie protokołów, które modyfikują priorytet zadania w zależności od bieżącego użycia zasobów, kontrolują dysponowanie zasobami za pomocą sprawdzeń wykonywanych przed każdym wejściem do sekcji krytycznych (*critical sections*) i ograniczają maksymalny czas blokowania zadań, które współdzielą te sekcje krytyczne

Przewidywalność: zarządzanie pamięcią

- Procesy widzą *pamięć wirtualną* (*virtual memory*), którą zarządza system operacyjny
- Pamięć wirtualna składa się z pamięci fizycznej RAM oraz przestrzeni (*swap*) na nośniku
- Aby szybciej odnajdować dane w pamięci wirtualnej stosuje się *stronicowanie* (*pages*)
- *Page fault* powoduje wczytanie odpowiedniej *ramki* (*page frame*) z nośnika do pamięci RAM — i dla lepszego wykorzystania zasobów — wykonywane jest w tym czasie *przełączanie kontekstowe* (*context switching*)
- Zarządzanie pamięcią jest niedeterministyczne z uwagi na nieprzewidywalne opóźnienia spowodowane przez page faults

Przewidywalność: język programowania

- Uniwersalne języki programowania:
  - Niedeterministyczne konstrukcje językowe – szacowanie czasu reakcji
  - Brak protokołów dostępu do współdzielonych zasobów – odwrócenie priorytetów
- Real-Time Euclid – tworzenie statycznych aplikacji czasu rzeczywistego gwarantujących zachowanie ograniczeń czasowych w czasie kompilacji
  - Brak dynamicznych struktur danych
  - Brak rekursji
  - Pętle ograniczone czasowo (time-bounded loops) – programista musi określić maksymalną ilość iteracji wykonywanych w pętli
  - Wyjątki przekroczenia czasu (timeout exceptions) – programista musi określić granice czasowe dla pętli oraz instrukcji dostępu do urządzeń
  - Klasyfikowanie procesów jako okresowych i nieokresowych
- Real-Time Concurrent C – tworzenie dynamicznych aplikacji czasu rzeczywistego
  - Konstrukcja językowe pozwalająca określać termin ostateczny procedury i wykonania procedury alternatywnej, gdy wymagania czasowe nie mogą być dotrzymane:

    `within deadline (d)` *`statement-1`* `else` *`statement-2`*

```
> FIN < ACK < FIN > ACK

init 5
```