

Algorytmy Równoległe i Rozproszone

Część IV - Model PRAM

Łukasz Kuszner
pokój 209, WETI

<http://www.sphere.pl/~kuszner/>
kuszner@sphere.pl

Oficjalna strona wykładu

<http://www.sphere.pl/~kuszner/ARiR/>

2005/06

Motywacja

Model - wstęp

PRAM - założenia

Twierdzenie Brent'a

Przetwarzanie danych

Kilka przykładów

Obliczenia w drzewach

Pointer jumping

Strona główna

Strona tytułowa



Strona 1 z 24

Powrót

Full Screen

Zamknij

Koniec

1. Powody rozwoju systemów równoległych i rozproszonych

- wymiana informacji,
- współdzielenie zasobów,
- zwiększanie niezawodności poprzez powielanie,
- zwiększanie wydajności poprzez zrównoleglanie,
- upraszczanie konstrukcji poprzez specjalizację.

Motywacja

Model - wstęp

PRAM - założenia

Twierdzenie Brent'a

Przetwarzanie danych

Kilka przykładów

Obliczenia w drzewach

Pointer jumping

Strona główna

Strona tytułowa



Strona 2 z 24

Powrót

Full Screen

Zamknij

Koniec

1.1. Wymiana informacji

Konieczność wymiany informacji pociąga za sobą konieczność budowy systemów, które taką wymianę umożliwiają. W powszechnym użyciu jest wiele takich systemów jak choćby globalna sieć Internet, sieci telefonii stacjonarnej i komórkowej, wewnętrzne sieci komputerowe dużych korporacji przemysłowych, wojskowe i cywilne systemy wczesnego ostrzegania, systemy nawigacji satelitarnej i inne.

[Strona główna](#)[Strona tytułowa](#)[◀](#) [▶](#)[◀](#) [▶](#)[Strona 3 z 24](#)[Powrót](#)[Full Screen](#)[Zamknij](#)[Koniec](#)

1.2. Współdzielenie zasobów

Wytworzenie, a później utrzymanie niektórych zasobów może być bardzo kosztowne. Konieczne staje się więc współdzielenie ich przez wielu użytkowników. W takim wypadku dla ułatwienia dostępu i uregulowania zasad korzystania z zasobów powstają często rozproszone systemy dostępu i kontroli. Jako przykłady można tu podać np.: teleskop kosmiczny Hubble'a, a w mniejszej skali ploter lub drukarkę współdzieloną przez kilku użytkowników. Zasobami mogą być też dane, lub moc obliczeniowa superkomputerów.

[Strona główna](#)[Strona tytułowa](#)[◀◀](#) [▶▶](#)[◀](#) [▶](#)[Strona 4 z 24](#)[Powrót](#)[Full Screen](#)[Zamknij](#)[Koniec](#)

1.3. Zwiększanie niezawodności poprzez powielanie

Systemy rozproszone mogą być potencjalnie bardziej niezawodne. O ile awaria lub zniszczenie samodzielnie działającego komputera uniemożliwia pracę całego systemu, o tyle awaria systemu rozproszonego może być zneutralizowana poprzez zastąpienie niesprawnego elementu poprzez inne działające równolegle.

Motywacja

Model - wstęp

PRAM - założenia

Twierdzenie Brent'a

Przetwarzanie danych

Kilka przykładów

Obliczenia w drzewach

Pointer jumping

Strona główna

Strona tytułowa

◀▶

◀▶

Strona 5 z 24

Powrót

Full Screen

Zamknij

Koniec

1.4. Zwiększanie wydajności poprzez zrównoleglenie

W systemach masowej obsługi powielenie jednostek wykonujących to samo zadanie powoduje wzrost wydajności. Przykładem tutaj mogą być komputery odpowiadające na zapytania kierowane do baz danych. Również w innego rodzaju systemach jeśli tylko realizowane zadania mogą być dzielone na mniejsze części, to możemy przyspieszyć obsługę poprzez zrównoleglenie pewnych operacji.

Motywacja

Model - wstęp

PRAM - założenia

Twierdzenie Brent'a

Przetwarzanie danych

Kilka przykładów

Obliczenia w drzewach

Pointer jumping

Strona główna

Strona tytułowa

◀◀ ▶▶

◀ ▶

Strona 6 z 24

Powrót

Full Screen

Zamknij

Koniec

Motywacja

Model - wstęp

PRAM - założenia

Twierdzenie Brent'a

Przetwarzanie danych

Kilka przykładów

Obliczenia w drzewach

Pointer jumping

Strona główna

Strona tytułowa

◀◀ ▶▶

◀ ▶

Strona 7 z 24

Powrót

Full Screen

Zamknij

Koniec

1.5. Upraszczenie konstrukcji poprzez specjalizację

Konstrukcja systemów komputerowych może być bardzo złożona. Podobnie jak klasyczna modularyzacja podział systemu na kooperujące części może zaowocować zmniejszeniem złożoności pojedynczych elementów i zarazem uproszczeniem konstrukcji całego systemu.

2. Model - wstęp

W modelu przetwarzania sekwencyjnego kluczową rolę pełni model maszyny RAM (random access machine). Każda taka maszyna składa się z ustalonego programu, jednostki obliczeniowej, taśmy (tylko do odczytu) z danymi wejściowymi, taśmy (tylko do zapisu) na wynik działania programu oraz nieograniczonej pamięci o dostępie swobodnym. Ponadto każda komórka pamięci jest w stanie zapamiętać liczbę całkowitą o nieograniczonym zakresie. Jednostka obliczeniowa nie jest skomplikowana – pozwala na wykonywanie najprostszych instrukcji takich jak: kopiowanie komórek pamięci, porównania i skoki warunkowe, podstawowe operacje arytmetyczne itp. Ustalony program użytkownika składa się z ciągu takich instrukcji.

Motywacja

Model - wstęp

PRAM - założenia

Twierdzenie Brent'a

Przetwarzanie danych

Kilka przykładów

Obliczenia w drzewach

Pointer jumping

Strona główna

Strona tytułowa

◀◀ ▶▶

◀ ▶

Strona 8 z 24

Powrót

Full Screen

Zamknij

Koniec

Miarą złożoności programów dla maszyny RAM są typowo czas działania mierzony liczbą wykonanych instrukcji i zużycie pamięci mierzone liczbą wykorzystywanych komórek. Żeby uchronić ten model przed zniekształceniami zabronione jest generowanie bardzo dużych liczb w krótkim czasie. Np. zabrania się generowania liczb o niewielomianowej długości zapisu w wielomianowym czasie. Można to osiągnąć albo przez uważny dobór zestawu instrukcji, albo przerzucając odpowiedzialność na „twórców algorytmów” dla danego modelu.

W ten sposób otrzymujemy gamę równoważnych modeli dla obliczeń sekwencyjnych.

Motywacja

Model - wstęp

PRAM - założenia

Twierdzenie Brent'a

Przetwarzanie danych

Kilka przykładów

Obliczenia w drzewach

Pointer jumping

Strona główna

Strona tytułowa

◀ ▶

◀ ▶

Strona 9 z 24

Powrót

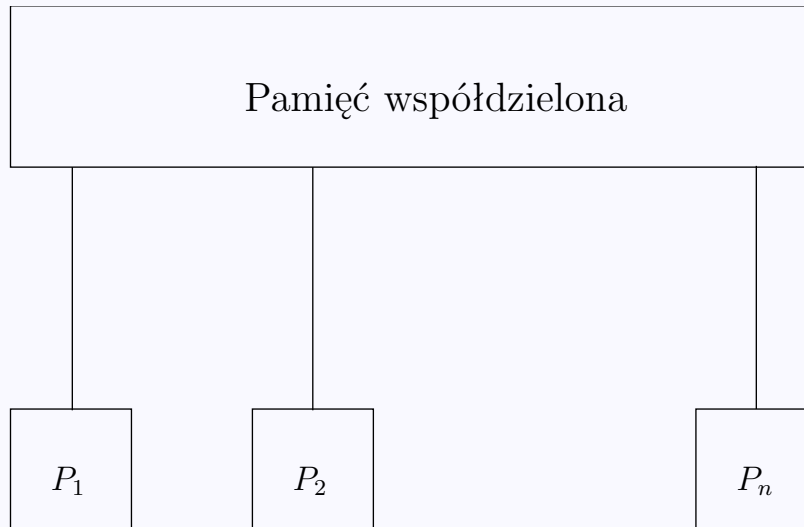
Full Screen

Zamknij

Koniec

Naturalnym uogólnieniem modelu RAM jest dodanie większej liczby jednostek obliczeniowych.

Ideę maszyny PRAM może ilustrować poniższy schemat:



Motywacja

Model - wstęp

PRAM - założenia

Twierdzenie Brent'a

Przetwarzanie danych

Kilka przykładów

Obliczenia w drzewach

Pointer jumping

Strona główna

Strona tytułowa

◀ ▶

◀ ▶

Strona 10 z 24

Powrót

Full Screen

Zamknij

Koniec

3. PRAM - założenia

- Pamięć jest wspólna dla wszystkich procesorów.
- Każdy procesor jest maszyną typu RAM.
- Wszystkie procesory działają synchronicznie.
- Czas działania mierzymy liczbą dostępów do pamięci współdzielonej.
- Zużycie pamięci liczymy liczbą użytych komórek.
- Dodatkowym parametrem jest liczba użytych procesorów. Tu zakładamy, że w wielomianowym czasie można użyć „tylko” wielomianowej liczby procesorów.

Motywacja

Model - wstęp

PRAM - założenia

Twierdzenie Brent'a

Przetwarzanie danych

Kilka przykładów

Obliczenia w drzewach

Pointer jumping

Strona główna

Strona tytułowa



Strona 11 z 24

Powrót

Full Screen

Zamknij

Koniec

3.1. Uwagi do założeń

- Ostatni punkt założeń można rozwiązać np. w taki sposób, że procesor P_1 oblicza potrzebną liczbę procesorów, a następnie włącza je wpisując liczbę do odpowiedniego rejestru.
- Liczenie dostępu do pamięci ma taki sens praktyczny, że zwykle wszelkie operacje typu komunikacyjnego zabierają znacznie więcej czasu niż obliczenia lokalne.
- Wadą założenia o jednostkowym czasie dostępu jest, występowanie w rzeczywistych systemach równoległych mechanizmów komunikacji o bardzo zróżnicowanej wydajności.

3.2. Dostęp do pamięci

Istnieje kilka sposobów modelowania równoległego dostępu do pamięci współdzielonej. We wszystkich modelach zakładamy oddzielenie operacji zapisu i odczytu. Przyjmujemy, że maszyna PRAM działa w cyklu składającym się z:

- (jeśli potrzeba) czytaj z pamięci współdzielonej,
- (jeśli potrzeba) wykonaj obliczenia lokalne,
- (jeśli potrzeba) pisz do pamięci współdzielonej.

W ten sposób zakładamy, że nie ma konfliktów typu: jednoczesny zapis/odczyt.

[Strona główna](#)[Strona tytułowa](#)[◀◀](#) [▶▶](#)[◀](#) [▶](#)

Strona 13 z 24

[Powrót](#)[Full Screen](#)[Zamknij](#)[Koniec](#)

Pozostają jednak konflikty typu: jednoczesny zapis/zapis i odczyt/odczyt. Generalnie możliwości są następujące:

- maszyna **EREW**-PRAM: nie dopuszcza się konfliktów żadnego rodzaju,
- maszyna **CREW**-PRAM: dopuszcza się konflikty typu jednoczesny odczyt,
- maszyna **ERCW**-PRAM: dopuszcza się konflikty typu jednoczesny zapis,
- maszyna **CRCW**-PRAM: dopuszcza się zarówno konflikty typu jednoczesny odczyt jak i jednoczesny zapis.

Przy czym w przypadku dopuszczenia jednoczesnego odczytu (CREW, CRCW) zakładamy, że wszystkie procesory przeczytają żadaną komórkę pamięci. W przypadku dopuszczenia jednoczesnego zapisu sytuacja jest bardziej złożona.

Motywacja

Model - wstęp

PRAM - założenia

Twierdzenie Brent'a

Przetwarzanie danych

Kilka przykładów

Obliczenia w drzewach

Pointer jumping

Strona główna

Strona tytułowa

◀▶

◀▶

Strona 14 z 24

Powrót

Full Screen

Zamknij

Koniec

3.2.1. Rozwiązywanie konfliktów typu jednoczesny zapis

- ECR (equality conflict resolution) - jednoczesny zapis się powiedzie, jeśli wszystkie procesory próbują zapisać to samo.
- PCR (priority conflict resolution) - zapis udaje się tylko procesorowi o najwyższym priorytecie.
- ACR (arbitrary conflict resolution) - jednemu z procesorów zapis się powiedzie.

Strona główna

Strona tytułowa

◀◀ ▶▶

◀ ▶

Strona 15 z 24

Powrót

Full Screen

Zamknij

Koniec

Motywacja
Model - wstęp
PRAM - założenia
Twierdzenie Brent'a
Przetwarzanie danych
Kilka przykładów
Obliczenia w drzewach
Pointer jumping

4. Twierdzenie Brent'a

Twierdzenie 1 *Każdy układ kombinacyjny o rozmiarze n , głębokości d i stopniu wejściowym bramek ograniczonym przez stałą da się symulować na n p -procesorowej maszynie CREW-PRAM w czasie $O(n/p + d)$.*

Ćwiczenie 1

Uzasadnij twierdzenie Brent'a (zob. Cormen str 793).

Strona główna

Strona tytułowa

◀◀ ▶▶

◀ ▶

Strona 16 z 24

Powrót

Full Screen

Zamknij

Koniec

Motywacja

Model - wstęp

PRAM - założenia

Twierdzenie Brent'a

Przetwarzanie danych

Kilka przykładów

Obliczenia w drzewach

Pointer jumping

Strona główna

Strona tytułowa



Strona 17 z 24

Powrót

Full Screen

Zamknij

Koniec

5. Modele przetwarzania danych

- SISD
- SIMD
- MISD
- MIMD

6. Kilka przykładów

Algorytm 1: Iloczyn skalarny

```
1: for  $i = 1$  to  $n$  in parallel do  
2:    $c_i = a_i * b_i$   
3: end for  
4:  $p = n/2$   
5: while  $p > 0$  do  
6:   for  $i = 1$  to  $p$  in parallel do  
7:      $c_i = c_i + c_{i+p}$   
8:   end for  
9:    $p = p/2$   
10: end while
```

We: Tablice współrzędnych $a[1 : n]$ i $b[1 : n]$

Wy: Liczba będąca iloczynem skalarnym wektorów a i b .

Model: EREW PRAM.

Czas $O(\lg n)$ i $O(n)$ procesorów.

Motywacja

Model - wstęp

PRAM - założenia

Twierdzenie Brent'a

Przetwarzanie danych

Kilka przykładów

Obliczenia w drzewach

Pointer jumping

Strona główna

Strona tytułowa

◀ ▶

◀ ▶

Strona 18 z 24

Powrót

Full Screen

Zamknij

Koniec

Motywacja

Model - wstęp

PRAM - założenia

Twierdzenie Brent'a

Przetwarzanie danych

Kilka przykładów

Obliczenia w drzewach

Pointer jumping

Strona główna

Strona tytułowa

◀ ▶

◀ ▶

Strona 19 z 24

Powrót

Full Screen

Zamknij

Koniec

Algorytm 2: Koniunkcja logiczna 1

```
1: result=TRUE
2: for  $i = 1$  to  $n$  in parallel do
3:   if  $A[i] == \text{FALSE}$  then
4:     result=FALSE
5:   end if
6: end for
```

We: Tablica wartości logicznych $A[1 : n]$.

Wy: result

Model: ERCW PRAM.

Czas $O(1)$ i $O(n)$ procesorów.

- Motywacja
- Model - wstęp
- PRAM - założenia
- Twierdzenie Brent'a
- Przetwarzanie danych
- Kilka przykładów
- Obliczenia w drzewach**
- Pointer jumping

Algorytm 3: Koniunkcja logiczna 2

```

1: result=FALSE
2: for  $i = 1$  to  $n$  in parallel do
3:   if  $A[i] == \text{FALSE}$  then
4:     result= $A[i]$ 
5:   end if
6: end for

```

We: Tablica wartości logicznych $A[1 : n]$.

Wy: result

Model: EREW-ECR PRAM.

Czas $O(1)$ i $O(n)$ procesorów.

Ćwiczenie 2

Uzasadnij poprawność powyższych algorytmów.

Strona główna

Strona tytułowa

◀ ▶

◀ ▶

Strona 20 z 24

Powrót

Full Screen

Zamknij

Koniec

7. Obliczenia w drzewie binarnym

Algorytm 4: Koniunkcja logiczna 3

```
1:  $p = n/2$ 
2: while  $p > 0$  do
3:   for  $i = 1$  to  $p$  in parallel do
4:      $A[i] = A[2i - 1]A[2i]$ 
5:   end for
6:    $p = p/2$ 
7: end while
```

We: Tablica wartości logicznych $A[1 : n]$.

Wy: result

Model: EREW PRAM.

Czas $O(\lg n)$ i $O(n)$ procesorów.

Motywacja

Model - wstęp

PRAM - założenia

Twierdzenie Brent'a

Przetwarzanie danych

Kilka przykładów

Obliczenia w drzewach

Pointer jumping

Strona główna

Strona tytułowa

◀◀ ▶▶

◀ ▶

Strona 21 z 24

Powrót

Full Screen

Zamknij

Koniec

8. Pointer jumping

Pointer jumping (przeskakiwanie) pozwala na tworzenie równoległych algorytmów dla list.

8.1. Przykład

Problem *list-ranking* – obliczanie odległości obiektu od końca listy. Niech A będzie tablicą obiektów, a $Link[i] = j$ oznacza, że element j następuje w liście po elemencie i . Jeśli $Link[i] = 0$, to nie ma kolejnego elementu, i jest elementem ostatnim. Przez *Head* oznaczymy pierwszy element na liście.

Algorytm 5: List Ranking

```
1: for  $i = 1$  to  $n$  in parallel do
2:   Rank[i]=1
3:   Next[i]=Link[i]
4: end for
5: for  $i = 1$  to  $\lceil \lg n \rceil$  do
6:   for  $i = 1$  to  $n$  in parallel do
7:     if  $Next[i] \neq 0$  then
8:       Rank[i] += Rank[Next[i]]
9:       Next[i] = Next[Next[i]]
10:    end if
11:   end for
12: end for
```

We: Tablice $A[1 : n]$, $Link[1 : n]$.

Wy: Rank[1:n]

Model: EREW PRAM.

Czas $O(\lg n)$ i $O(n)$ procesorów.

Motywacja

Model - wstęp

PRAM - założenia

Twierdzenie Brent'a

Przetwarzanie danych

Kilka przykładów

Obliczenia w drzewach

Pointer jumping

Strona główna

Strona tytułowa

◀ ▶

◀ ▶

Strona 23 z 24

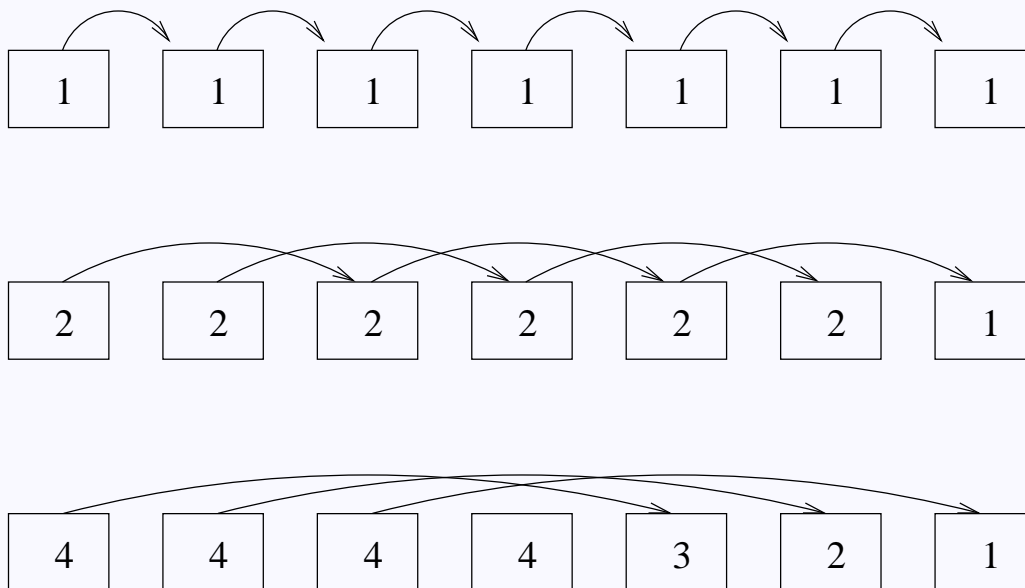
Powrót

Full Screen

Zamknij

Koniec

Na rysunku znajdują się procesory oznaczone prostokątami w kolejności wskazywanej przez *Link*. Strzałki obrazują wartość *Next*, a liczby wpisane w każdy prostokąt wartości *Rank*.



Motywacja

Model - wstęp

PRAM - założenia

Twierdzenie Brent'a

Przetwarzanie danych

Kilka przykładów

Obliczenia w drzewach

Pointer jumping

Strona główna

Strona tytułowa



Strona 24 z 24

Powrót

Full Screen

Zamknij

Koniec