Zaawansowane algorytmy

Wstęp 2/2

dr Janusz Dybizbański

4. Projektowanie algorytmów ze względu na liczbę procesorów

Algorytm 2. Suma wszystkich wartości w tablicy

wejście:

- tablica zmiennych liczbowych $A[1...n]$ umieszczona w pamięci globalnej, $n = 2^k$, $k \in \mathbb{N}$

Algorytm 2. Suma wszystkich wartości w tablicy

wejście:

- tablica zmiennych liczbowych $A[1...n]$ umieszczona w pamięci globalnej, $n = 2^k$, $k \in \mathbb{N}$

wyjście:

- zmienna liczbowa C umieszczona w pamięci globalnej taka, że $C = \sum_{i=1}^{n} A[i]$

a) $p = 1$

a) $p = 1$

```
1 C := A[1]
2 for i:=2 to n do
3       C := C + A[i]
```

a) $p = 1$

```
1 C := A[1]
2 for i:=2 to n do
3       C := C + A[i]
```

W jaki sposób przyśpieszyć działanie powyższego algorytmu, gdy mamy więcej procesorów?

b) $p \geq \frac{n}{2}$

b) $p \geq \frac{n}{2}$

Program dla wszystkich procesorów:

```
1 for h := 1 to log n do
2     for 1 <= i <= n/2^h pardo
3         A[i] := A[2i-1] + A[2i]
4 if i = 1 then
5     C := A[1]
```

b) $p \geq \frac{n}{2}$

Program dla wszystkich procesorów:

```
1 for h := 1 to log n do
2     for 1 <= i <= n/2^h pardo
3         A[i] := A[2i-1] + A[2i]
4 if i = 1 then
5     C := A[1]
```

W jaki sposób napisać powyższy algorytm z perspektywy jednego procesora?

c) $1 < p < \frac{n}{2}$, $p = 2^q$.

c) $1 < p < \frac{n}{2}$, $p = 2^q$.

Program dla procesora o identyfikatorze i:

```
1 for h := 1 to log n do
2     if k-h-q > 0 then
3         for j := 1 to 2^(k-h-q) do
4             l := i + (j-1)p
5             A[l] := A[2l-1] + A[2l]
6     else if i <= 2^(k-h) then
7         A[i] := A[2i-1] + A[2i]
8 if i=1 then
9     C := A[1]
```

c) $1 < p < \frac{n}{2}$, $p = 2^q$.

Program dla procesora o identyfikatorze i:

```
1 for h := 1 to log n do
2     if k-h-q > 0 then
3         for j := 1 to 2^(k-h-q) do
4             l := i + (j-1)p
5             A[l] := A[2l-1] + A[2l]
6     else if i <= 2^(k-h) then
7         A[i] := A[2i-1] + A[2i]
8 if i=1 then
9     C := A[1]
```

Czasem niemożliwym jest zrównoleglenie algorytmu zaprojektowanego na małą liczbę procesorów. Łatwiej jest symulować algorytm zaprojektowany na większą liczbę procesorów.

c) $1 < p < \frac{n}{2}$, $p = 2^q$.

Program dla procesora o identyfikatorze i:

```
1 for h := 1 to log n do
2     if k-h-q > 0 then
3         for j := 1 to 2^(k-h-q) do
4             l := i + (j-1)p
5             A[l] := A[2l-1] + A[2l]
6     else if i <= 2^(k-h) then
7         A[i] := A[2i-1] + A[2i]
8 if i=1 then
9     C := A[1]
```

Czasem niemożliwym jest zrównoleglenie algorytmu zaprojektowanego na małą liczbę procesorów. Łatwiej jest symulować algorytm zaprojektowany na większą liczbę procesorów.

W przyszłości będziemy zakładać, że mamy dowolnie dużo procesorów (n, n^2, n^3).

5. Parametry algorytmów równoległych

5. Parametry algorytmów równoległych

n - rozmiar danych wejściowych

- $T(n)$ - **złożoność czasowa**, liczba kroków wykonywanych przez algorytm równoległy dla danych wejściowych rozmiaru n
- $W(n)$ - **praca**, liczba operacji wykonywanych przez wszystkie procesory w trakcie działania całego algorytmu,
- $P(n)$ - **liczba używanych procesorów**.

Twierdzenie

Praca algorytmu równoległego nie może być mniejsza niż czas działania najlepszego algorytmu sekwencyjnego rozwiązującego ten sam problem.

Twierdzenie

Praca algorytmu równoległego nie może być mniejsza niż czas działania najlepszego algorytmu sekwencyjnego rozwiązującego ten sam problem.

Definicja

Algorytm równoległy nazywamy **optymalnym** (prace optymalną), gdy jego praca jest asymptotycznie równa złożoności czasowej najlepszego algorytmu sekwencyjnego.