AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

Wybrane metody automatycznej generacji kodu na procesory wielordzeniowe

Student realizujący: Piotr Listkiewicz
Opiekun pracy: dr inż. Marcin Pietroń

Cel pracy

- Analiza metod automatycznej detekcji kodu który można zrównoleglić
- Wytworzenie narzędzia umożliwiającego automatyczną generację kodu w wersji równoległej
- Obecnym celem jest implementacja narzędzia dla języka C , które przekształca go do postaci OpenMP
- Jeżeli pierwsza faza projektu poszłaby pomyślnie, to w drugiej implementacja narzędzia przekształcającego bytecody javy do postaci równoległej przy pomocy wątków

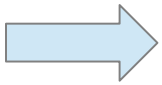
Przekształcenia pętli

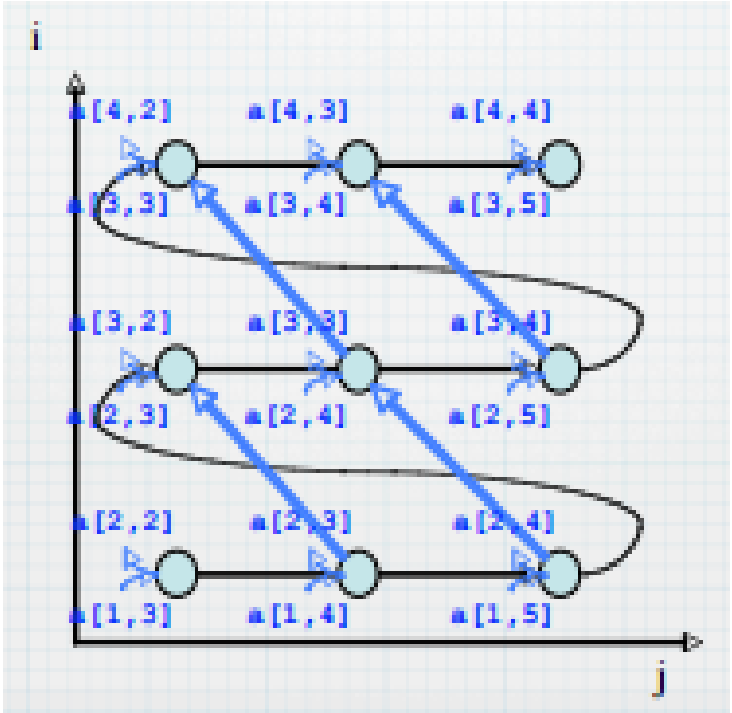
- **Loop Vectorization – przekształcenia skalarów na wektory/macierze**
- **Loop Interchanging – podmiana kolejności pętli**
- **Loop concurrentization – przypisanie iteracji do wykonania na różne procesory**
- **Loop scalarization – zmiana wektorów/macierzy na skalary**
- **Loop fusion – scalenie zagnieżdżonych pętli**

Testowanie zależności

- Wykrywanie zależności występujących w pętli pomiędzy iteracjami i instrukcjami
- Jest kilka możliwych rodzajów zależności pomiędzy instrukcjami
- Na ich podstawie buduje się graf zależności

```
  for i = 2 to 4
    for j = 2 to 4
1:    a[i,j] = a[i-1,j+1];
```





Istota badania zależności

```
      DO i1 = L1, U1
         DO i2 = L2, U2
            ...
            DO in = Ln, Un
S1             A(f(i1,...,in)) = ...
S2             ... = A(g(i1,...,in))
             ENDDO
             ...
         ENDDO
      ENDDO
```

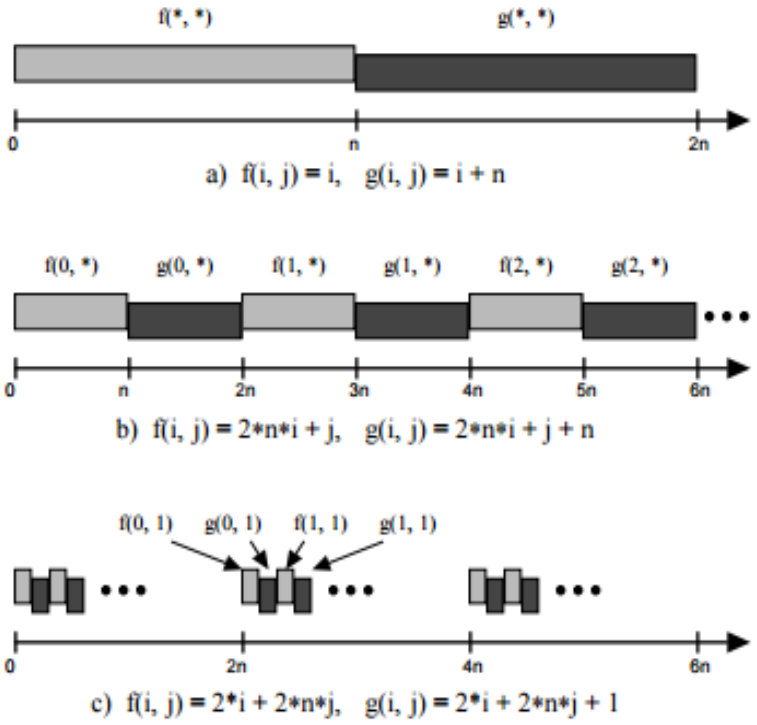
$$f(x_1, x_2, \ldots, x_n) = g(y_1, y_2, \ldots, y_n)$$

$$L_i \le x_i, y_i \le U_i, \ \forall i,\ 1 \le i \le n$$

- **Problem jest w gruncie rzeczy taki sam jak problem programowania całkowitoliczbowego**
- **Zbadanie zależności trochę utrudnia fakt ,że to problem NP-zupełny**

Range Test

- **Metoda na testowanie zależności**
- **Wykrywanie czy pomiędzy iteracjami nastąpiło odwołanie do tych samych elementów tablicy**
- **Wykorzystuje własności funkcji odwołującej się do tablicy**



GCD test

$$f(x_1,x_2,\ldots,x_n) = g(y_1,y_2,\ldots,y_n)$$

$$f(x_1,x_2,\ldots,x_n) = a_0 + a_1x_1 + \ldots + a_nx_n$$

$$g(y_1,y_2,\ldots,y_n) = b_0 + b_1y_1 + \ldots + b_ny_n$$

$$a_0 - b_0 + a_1x_1 - b_1y_1 + \ldots + a_nx_n - b_ny_n = 0$$

-> równanie diofantyczne ma rozwiązanie wtw gdy:

$$\gcd(a_1,\ldots,a_n,b_1,\ldots,b_n) \text{ divides } b_0 - a_0.$$

Ohmega Test

- **Metoda Integer Programming**
- **Podobna do GCD test**
- **Wykonujemy przekształceń zmniejszających współczynniki by w końcu któryś z nich miał wartość 1 i mógł zostać wyeliminowany**

$$
\begin{aligned}
&7x + 12y + 31z = 17\\
&3x + 5y + 14z = 7\\
&1 \le x \le 40\\
&-50 \le y \le 50
\end{aligned}
$$

$$0 \le \sigma \le 2$$

- **Poza jednym współczynnkiem pozostałe zmniejszyły się maksymalnie o 2/3**

$$-|a_k|\sigma + \sum_{i \in V - \{k\}} ((\lfloor a_i/m + \frac{1}{2} \rfloor + (a_i \widehat{\bmod} m))x_i = 0$$

Przykłady zastosowania testowania zależności

- **Jeżeli pomiędzy iteracjami pętli nie ma zależności mogą być wykonane jednocześnie !!**
- **Jeżeli kierunki wektorów zależności są w porządku leksykograficznym po zmianie kolejności pętli to była ona poprawna**
-

Obecny stan prac

- Generalny research literatury
- Przekształcenie kodu w C do postaci drzewa rozbioru
- Przekstałcenie drzewa rozbioru do AST
- Rozpoczęcie prac nad generacją drzewa zależności
- Przygotowanie systemu do podpięcia istniejących algorytmów testowania zależności

Pytania?