Analiza efektywności mnożenia macierzy w systemach z pamięcią współdzieloną

WYKŁAD Z PRZETWARZANIA RÓWNOLEGŁEGO KWIECIEŃ 2020

Organizacja dostępów do pamięci a efektywność jej wykorzystania -
na przykładzie mnożenie tablic

- Pozytywny efekt zastosowania pamięci podręcznej procesora wynika z:
  - wielokrotnego wykorzystania danych z pp (szybki dostęp) sprowadzonych jednokrotnie - jako linia ppp (wolny dostęp)

Cechami programów, które zapewniają efektywność dostępu do maczierzy są:

- czasowa lokalność odwołań (ang. temporal locality of reference) – dane raz sprowadzone do pamięci zostaną użyte wielokrotnie zanim zostaną z pamięci usunięte lub unieważnione, brak ***clo*** powoduje niski stosunek trafień do pp i spowalnia przetwarzanie.
- przestrzenna lokalnością odwołań - (ang. spatial locality of memory access) korzystanie w kodzie z danych zajmujących sąsiednie lokacje w pamięci – brak ***plo*** powoduje niski stosunek trafień do pp i do bufora translacji adresów.

(np. jeżeli tablica jest zapisywana wierszami w pamięci to kolejne **dostępy do tablicy** powinny też, jeśli to możliwe, być realizowane wierszami).

Mnożenie macierzy – dostęp do pamięci podręcznej [język C, kolejność - j,i,k][1]

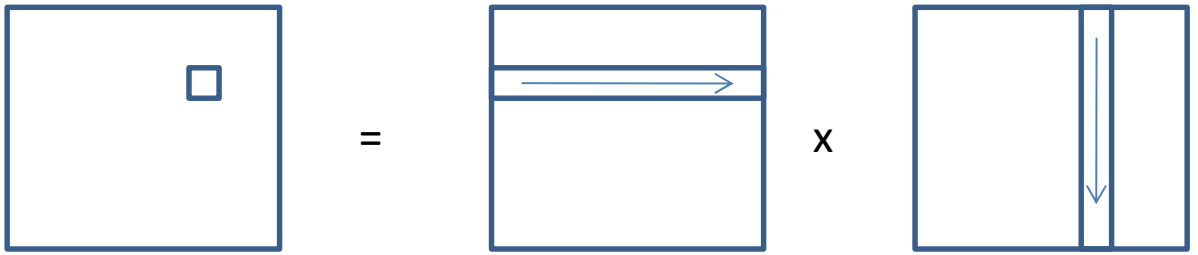
A,B,C są tablicami nxn

```
for (int j = 0 ; j < n ; j++)
    for ( int i = 0 ; i < n ; i++)
        for (int k = 0 ; k < n ; k++)
            C[i][j] + = A[i][k] * B[k][j] ;
```

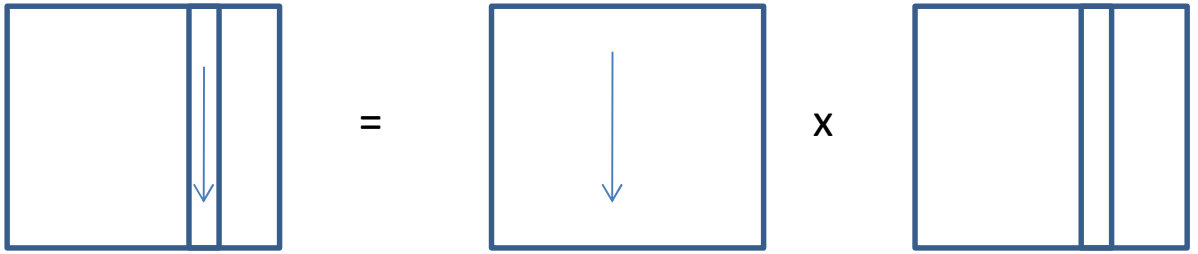
A[i][*] lokalność przestrzenna danych – różne elementy z linii pp wykorzystane w kolejnych iteracjach

B[*][j] możliwy brak wielokrotnego pobrania linii (kiedy?)

C[i][j] czasowa lokalność odwołań – ten sam element dla każdej iteracji pętli wewnętrznej



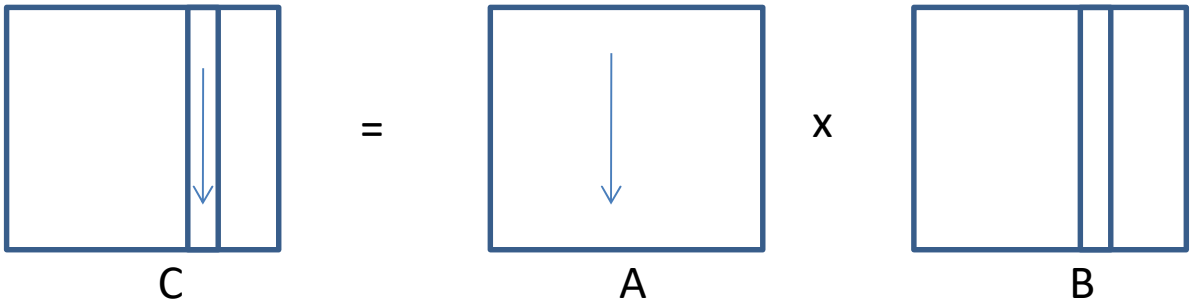
Ilość danych wykorzystywanych w pętli wewnetrznej



Ilość danych wykorzystywanych w 2 pętlach wewnętrznych A współbieżnie ?

C A B

Mnożenie macierzy – lokalność czasowa dostępu- pamięć podręczna [j,i,k]



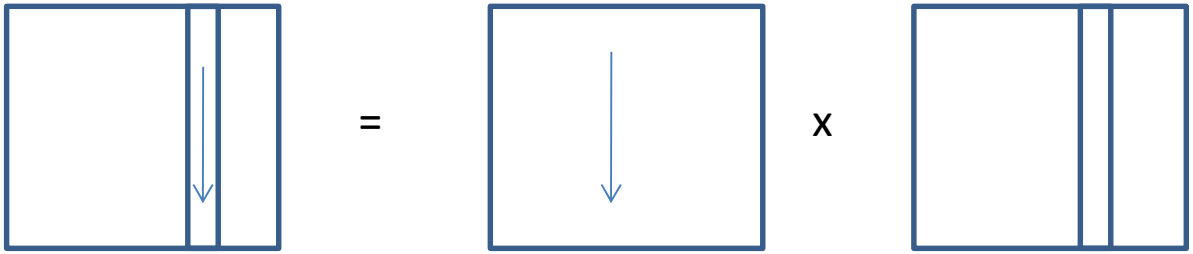
Rysunek dla i

- C[i][j] + = A[i][k] * B[k][j]
- Warunek na czasową lokalność dostępów do danych ?
- Jaka jest wielkość maksymalnego zbioru danych (w pp) używanego cyklicznie?

Warunek na lokalność czasową odwołań do danych (jednokrotne pobieranie danych) = pamięć podręczna mieści:

- tablicę A (używana n razy)
- linie pp zawierające jedną kolumnę tablicy B (ta sama kolumna używana wielokrotnie) i
- linie pp zawierające jedną kolumnę tablicy C (kolejna do wyznaczenia kolumna tablicy korzysta z tych samych linii pp)

Mnożenie macierzy – lokalność przestrzenna dostępu - DTLB [C, j,i,k]



- C[i][j] + = A[i][k] * B[k][j]
- Warunek na przestrzenną lokalność dostępów do danych ?
- Jaki jest wielkość maksymalnego zbioru stron wirtualnych używanego cyklicznie?

Warunek na lokalność przestrzenną odwołań do danych =

Bufor translacji mieści:

- adresy stron zawierających tablicę A (używana n razy),
- adresy stron zawierających cyklicznie wykorzystywane strony pw dla tablicy B (przy przechodzeniu wielokrotnie kolumną) i
- adresy stron zawierających cyklicznie wykorzystywane strony pw dla tablicy C (przy przechodzeniu przez tablicę kolejnymi kolumnami).

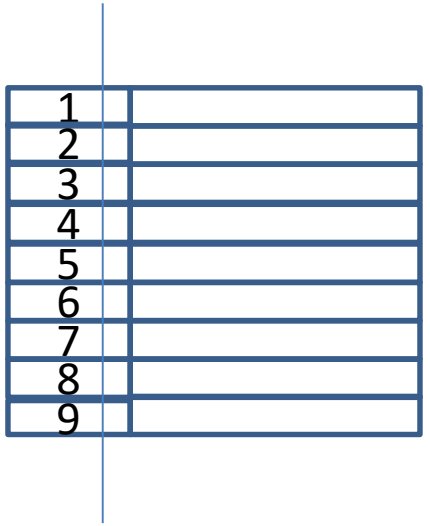
Zawartość pamięci podręcznej przy jednokrotnym dostępie do tablicy wierszami

W tym przypadku:

- przeglądamy dane po kolei i nie wracamy do wcześniej zapisanych informacji,
- ze względu na dostęp wierszami kolejno czytane dane pochodzą z tej samej linii pamięci podręcznej procesora (ppp) lub kolejnej linii ppp
- Wielość pamięci zapewniająca czasową lokalność dostępu do danych (jednokrotne pobieranie danych z pamięci podręcznej) wynosi 64 bajty – jedna linia ppp.

Zawartość pamięci podręcznej przy jednokrotnym dostępie do tablicy kolumnami

- Przeglądamy dane z odstępem, kolejne słowa oddalone są o długość wiersza macierzy, przy macierzach o wierszach dłuższych niż długość linii ppp (64 B) kolejne słowa w tym wzorze dostępu będą leżały w innej linii ppp
- w jednej linii pamięci podręcznej procesora (przy wyrównaniu słów do początku linii) znajduje się 16 słów 4 bajtowych,
- dane z jednej linii pamięci podręcznej procesora należą do 16 kolejnych kolumn macierzy,
- raz pobrana linia pamięci podręcznej może dostarczać danych z kolejnych 15 kolumn macierzy,
- aby dane z kolejnych kolumn (już pobrane do ppp) nie musiały być pobierane ponownie konieczne jest aby linie ppp wykorzystane dla jednej kolumny zostały zachowane do końca przechodzenia przez tę kolumnę.
- Pamięć podręczna pozwalająca na lokalność czasową przy tym wzorcu dostępu ma rozmiar N* 64B.
- Na rysunku ponumerowano kolejnymi liczbami naturalnymi linie ppp kolejno sprowadzane do ppp przy kolejnych **dostępach** do tablicy.



Aby nie pobierać linii ppp ponownie PP> 9X 64 Bajtów

Zawartość pamięci podręcznej przy wielokrotnym dostępie do tablicy wierszami lub kolumnami

- Przeglądamy dane po kolei z obszaru tablicy wielokrotnie,
- przy każdym przejściu przez tablicę korzystamy z wszystkich linii, na których zawarte są dane macierzy,
- wielkość pamięci zapewniająca czasową lokalność dostępu do danych (jednokrotne pobieranie danych z pamięci podręcznej) jest równa liczbie linii ppp składających się na ciągły obszar tablicy - dla tablicy nxn elementów typu float (4 bajty) sufit(4xnxn/64)*64 bajtów.

Zawartość pamięci podręcznej przy wielokrotnym dostępie do tablicy wierszami lub kolumnami

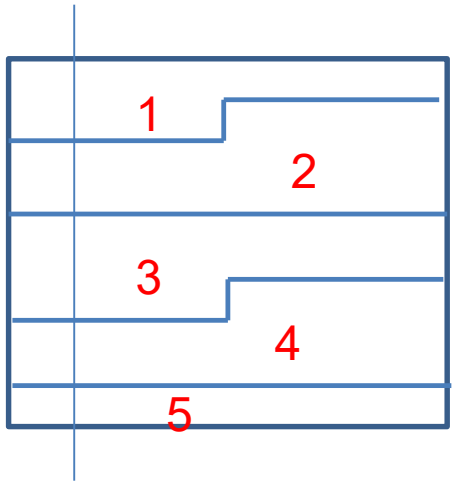
- Brak wystarczającej wielkości pamięci podręcznej skutkuje brakiem trafienia do pamięci przy dostępie do kolejnej linii pamięci podręcznej, która ze względu na brak miejsca została z pp usunięta.
- Brak trafienia występuje wtedy, gdy **ilość danych pobieranych pomiędzy dostępami** do tej samej linii ppp nie mieści się w ppp.
- Na efektywny dostęp do danych mają zatem wpływ:
  - wielkość ppp i
  - wielkość danych wykorzystywanych cyklicznie, wynikająca z kolejności odwołań do danych wykorzystywanych wielokrotnie.

Zawartość bufora translacji przy jednokrotnym dostępie tablicy wierszami

- Przeglądamy słowa z pamięci po kolei i nie wracamy do wcześniej używanych informacji,
- ze względu na dostęp wierszami kolejno czytane dane pochodzą z jednej strony pamięci wirtualnej - korzystamy z jednego wpisu w DTLB związanego z tą stroną,
- przechodząc na kolejną stronę nie będziemy potrzebować już w przyszłości dotychczasowego adresu zatem maksymalny **zbiór cyklicznie wykorzystywanych par adresów** danych z tej tablicy ze względu na brak powrotu na tę samą stronę ma wielkość = 1 para adresów (adres wirtualny strony i adres rzeczywisty – adres ramki).

Zawartość bufora translacji przy jednokrotnym dostępie do tablicy wzdłuż kolumn

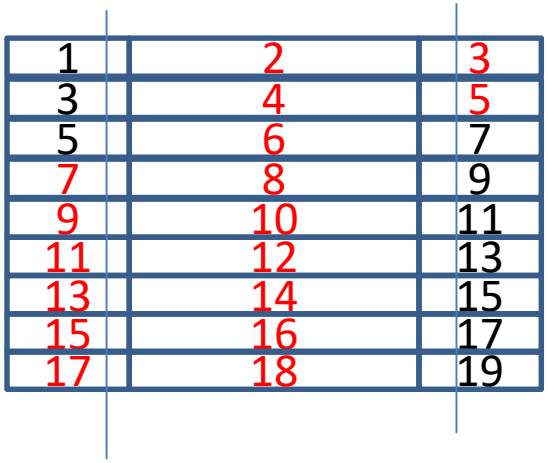
- Liczba wszystkich potrzebnych par adresów jest równa liczbie stron pamięć wirtualnej zajmowanych przez tę macierz.
- Liczba **cyklicznie wykorzystywanych stron danych** dla macierzy zapewniająca minimalizację kosztu sprowadzania adresów do DTBL jest zależna od wielkości macierzy.
- Dla małych macierzy, gdy długość wiersza macierzy jest mniejsza od wielkości strony pw i strona zawiera kilka wierszy macierzy może się zdarzyć że będziemy przechodzić cyklicznie przez **wszystkie** strony na których macierz się znajduje, wariant rysunek A i liczba stron odwiedzanych w cyklu wyniesie sufit(sufit(4xnxn/64)*64/4/1024).



Obszary są stronami pw, a prostokąty tablicami danych. Liczba cyklicznie używanych stron pw 5

Zawartość bufora translacji przy jednokrotnym dostępie do tablicy wzdłuż kolumn

- Liczba wszystkich potrzebnych par adresów jest równa liczbie stron pamięć wirtualnej zajmowanych przez tę macierz.
- Liczba **cyklicznie wykorzystywanych stron danych** dla macierzy zapewniająca minimalizację kosztu sprowadzania adresów do DTBL jest zależna od wielkości macierzy.



Dla dużych macierzy, gdy długość wiersza macierzy jest równa N lub większa od wielkości strony pw kolejne słowa z kolumny macierzy będą znajdować się na różnych stronach pw. Zatem przechodząc kolumnę macierzy o N wierszach odwiedzamy N stron pw. Część tych stron, które zawierają częściowo wiersz poprzedni i kolejny macierzy będzie wykorzystywana ponownie. Takich stron jest co najwyżej N-1.

Obszary są stronami, a prostokąty tablicami danych. Liczba cyklicznie używanych stron pw 17 = N+(N-1)

Zawartość bufora translacji przy wielokrotnym czytaniu tablicy wierszami lub kolumnami

W tym przypadku:

- przeglądamy dane z obszaru tablicy wielokrotnie,
- przy każdym przejściu przez tablicę korzystamy z wszystkich stron pamięci wirtualnej na których tablica się znajduje,
- maksymalny **zbiór cyklicznie wykorzystywanych par adresów** danych z tej tablicy ze względu na powrót (wielokrotny) na tę samą stronę ma wielkość równą liczbie stron, na których ta tablica się znajduje.
- Tablica nxn elementów typu float (4 bajty)
  - ciągły obszar danych o rozmiarze w pamięci w liniach pamięci podręcznej procesora to sufit(4xnxn/64)
  - ciągły obszar danych o rozmiarze w pamięci w bajtach to sufit(4xnxn/64)*64
- Liczba stron tablicy i par wpisów w buforze translacji dla tablicy nxn elementów typu float (4 bajty) sufit(sufit(4xnxn/64)*64/4/1024)
  - Dla typowej strony pamięci wirtualnej o wielkości 4KiB

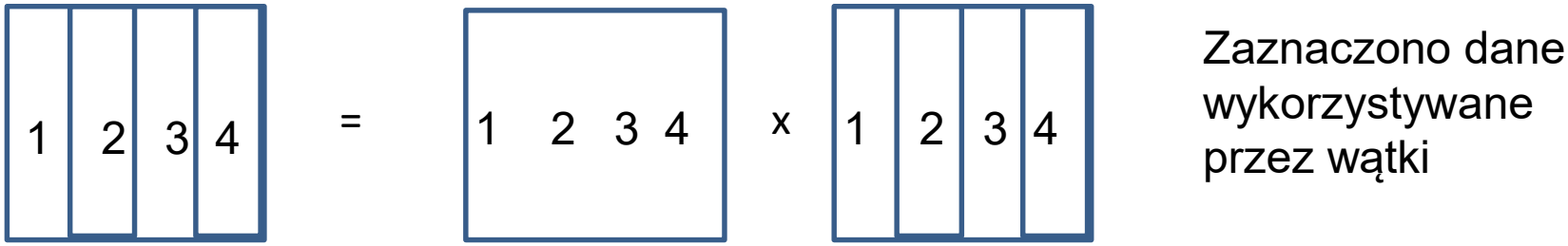
Zawartość bufora translacji przy wielokrotnym czytaniu tablicy wierszami lub kolumnami

- Brak wystarczającej wielkości bufora translacji skutkuje brakiem trafienia przy dostępie do danych spod nieznanego w DTLB adresu.
- Brak trafienia występuje wtedy, gdy **liczba różnych stron pw używanych pomiędzy dostępami** do tej samej strony jest większa od rozmiaru DTLB rdzenia.
- Na efektywny dostęp do danych ze względu na DTLB mają zatem wpływ:
  - wielkość DTLB i
  - wielkość zbioru cyklicznie wykorzystywanych stron wynikająca z kolejności odwołań do danych mieszczących się na wykorzystywanych wielokrotnie stronach, ten zbiór jest tym większy im bardziej dane są rozproszone w pamięci na różnych stronach wirtualnych.

Mnożenie macierzy równolegle – kolejność dostępu j,i,k

A,B,C są tablicami nxn

```
#pragma omp parallel for  //4 wątki podział  statyczny blokowy
 for (int j = 0 ; j < n ; j++)
      for ( int i = 0 ; i < n ; i++)
            for (int k = 0 ; k < n ; k++)
                  C[i][j] + = A[i][k] * B[k][j] ;
```



Zaznaczono dane wykorzystywane przez wątki

C A B

Możliwe zależności w dostępie do pamięci- PP a DTBL

Przykładowy procesor – parametry :

- bufor translacji adresów dla danych - DTLB ok. 500 par wpisów adresów dla rdzenia procesora
- Pamieć podręczna procesora L3 = 6MB

Sytuacja 1:

- Pamięć pomieści 2MB dla jednego rdzenia wynikające z ciągłego obszaru 500 stron wirtualnych o wielkości 4 kB
- 4 rdzenie z własnymi DTLB mogą posiadać adresy dla 4*2MB pamięci wirtualnej.
- Niektóre dane, których adres jest znany mogą nie mieścić się w PP L3 – 4* 2 MB > 6MB pp L3
- Zatem możliwe są jednocześnie: trafienie do DTLB oraz nietrafienie do pp

Sytuacja 2:

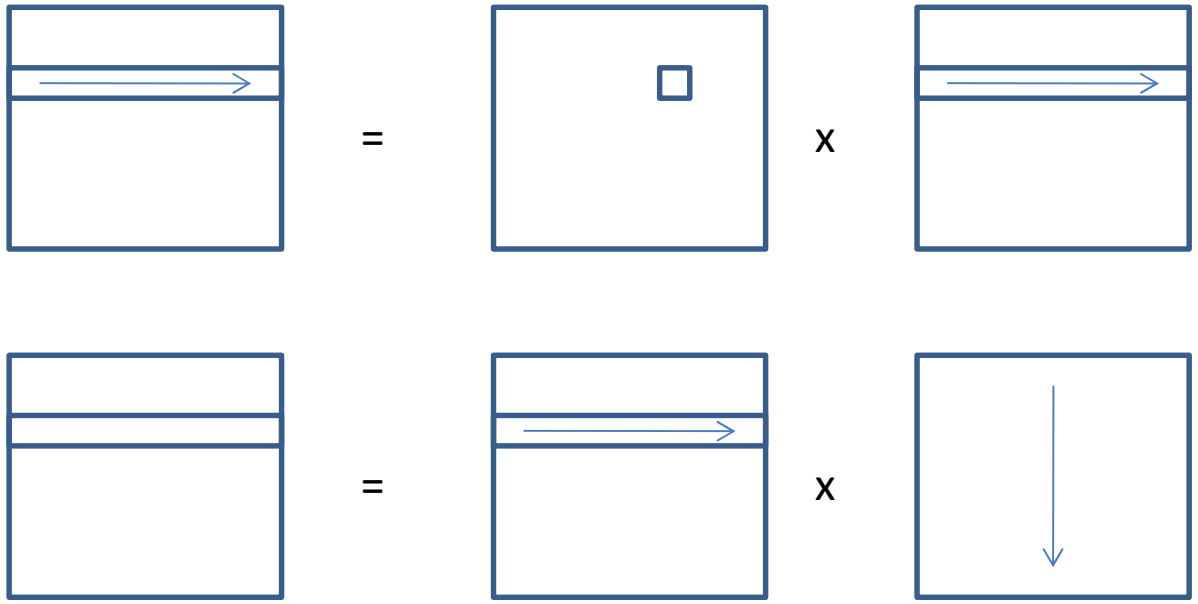
- Minimalny rozmiar danych dostępny przez 4 bufory DTLB gdy bufory mają jednakową zawartość i każdy adres wskazuje tylko na jedyną linię ze strony pw (dostęp z odstępem 4 KiB)
- Wielkość dostępnych natych adresach danych to tylko 500 x64B czyli 32 KiB<<PPP wtedy danych w ppp jest znacznie więcej niż adresów do nich możemy łątwo znaleźć.
- Zatem możliwe również nietrafienie do DTLB oraz trafienie ppp.

Mnożenie macierzy – pamięć podręczna[C, i,k,j][1]

A,B,C są tablicami nxn

```
for ( int i = 0 ; i < n ; i++)
    for (int k = 0 ; k < n ; k++)
        for (int j = 0 ; j < n ; j++)
            C[i][j] + = A[i][k] * B[k][j] ;
```

A[i][k] – lokalność czasowa odwołań
B[k][*], C[i][*] – lokalność przestrzenna odwołań (sąsiednie słowa w kolejnych instr.)



Najbardziej zagnieżdżona pętla j zapewnia lokalność przestrzenną dostępu do danych, odpowiednia wielkość pamięci podręcznej zapewni lokalność czasową dostępu do danych.

Mnożenie macierzy [i,k,j]– pamięć podręczna

Rysunek dla iteracji k

C[i][j] + = A[i][k] * B[k][j] (powyżej dane w 2 iteracji)

Warunek na czasową lokalność dostępów do danych ?

Jaki jest wielkość maksymalnego zbioru danych (w pp) używanego cyklicznie?

Warunek na lokalność czasową odwołań do danych =
pamięć podręczna mieści:

- tablicę B (używana n razy)
- linia pp zawierające jedno słowo tablicy A (jedno powolne przejście tablicy) i
- linie pp zawierające jeden wiersz tablicy C (wiersz przechodzony wielokrotnie)

Co się dzieje jeśli dane się nie mieszczą ? Tablica B pobierana do ppp N razy !
Rozwiązanie problemu – podział na etapy przetwarzania.

Mnożenie macierzy [i,k,j] loklaność przestrzenna dostepu - bufor DTLB

C[i][j] + = A[i][k] * B[k][j] (powyżej dane w 2 iteracji)

Warunek na przestrzenną lokalność dostępów do danych ?

Jaki jest wielkość maksymalnego zbioru stron pamięci - używanego cyklicznie?

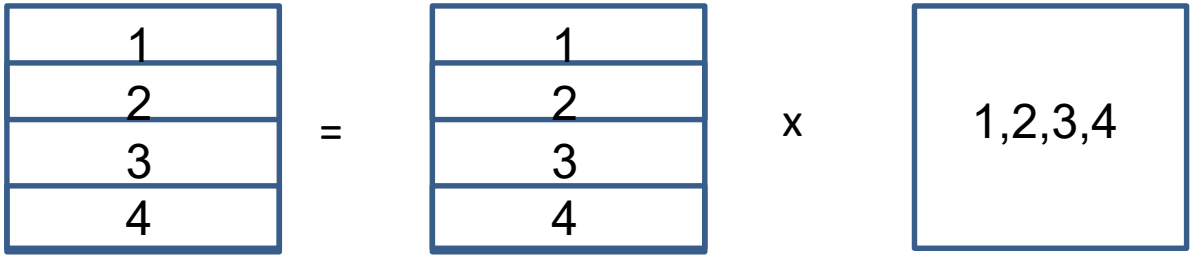
Warunek na lokalność przestrzenną odwołań do danych =
bufor translacji mieści:

- adresy stron zawierające tablicę B (używana n razy)
- adres strony zawierającej jedno słowo tablicy A (jedno powolne przejście tablicy) i
- adres(y) strony mieszczącej jeden wiersz tablicy C (wiersz przechodzony wielokrotnie)

Mnożenie macierzy [i,k,j]

A,B,C są tablicami nxn

```
#pragma omp parallel for // 4 wątki
 for ( int i = 0 ; i < n ; i++)
      for (int k = 0 ; k < n ; k++)
           for (int j = 0 ; j < n ; j++)
                C[i][j] + = A[i][k] * B[k][j] ;
```

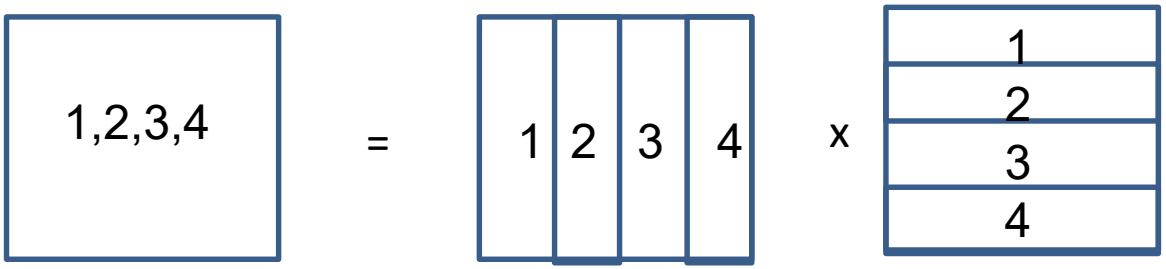


Dane wykorzystywane przez wątki

Mnożenie macierzy [i,k,j]

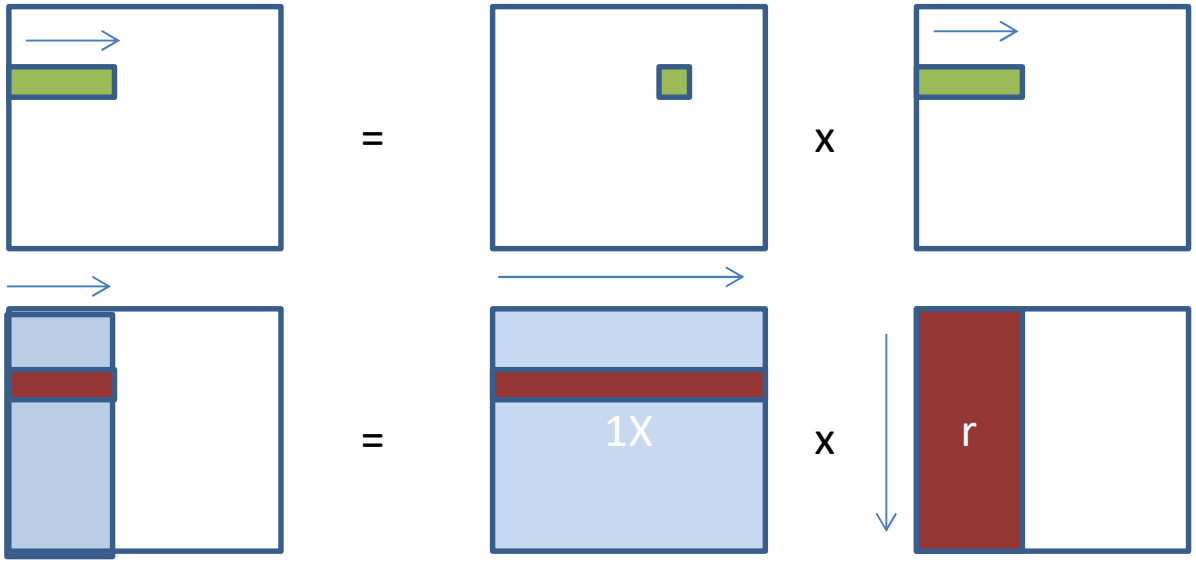
A,B,C są tablicami nxn

```
#pragma omp parallel // 4 wątki
  for ( int i = 0 ; i < n ; i++)
#pragma omp for
        for (int k = 0 ; k < n ; k++)
              for (int j = 0 ; j < n ; j++)
                    C[i][j] + = A[i][k] * B[k][j] ;
```



Dane wykorzystywane przez wątki
Wyścig w dostępie do danych – kod niepoprawny

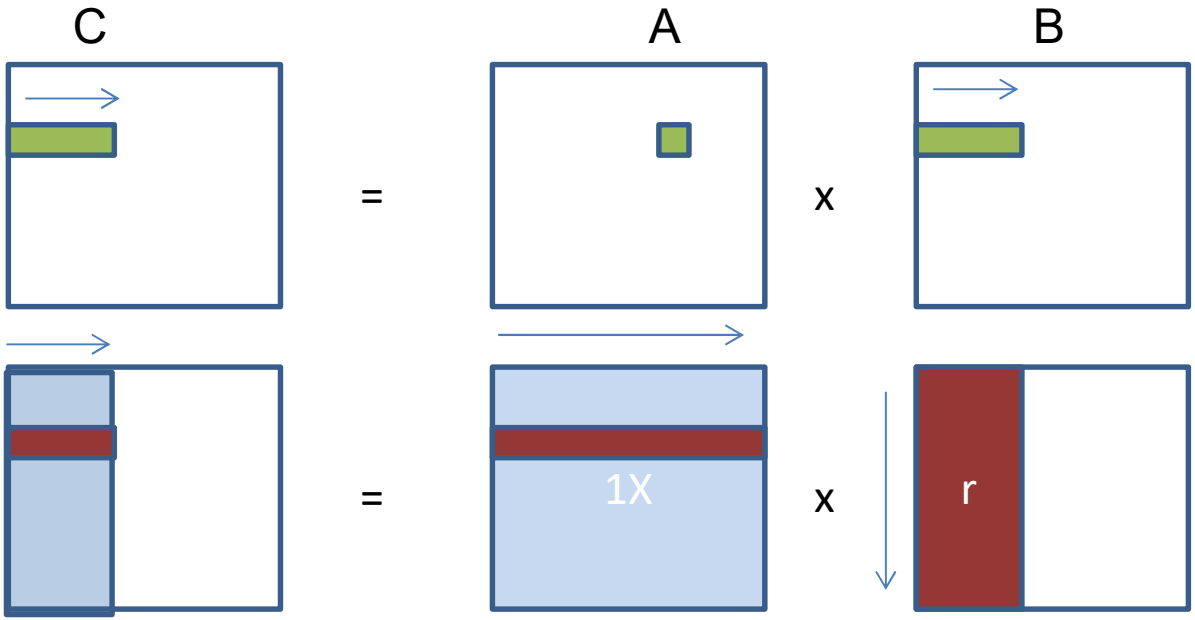
Mnożenie macierzy [i,k,j*]
wiele faz obliczeń - zmniejszenie zakresu pętli wewnętrznej



Obliczamy fragmenty kolejnych wierszy macierzy wynikowej (nie cały wiersz)

```
for ( int j = 0 ; j < n ; j+=r) // iteracje po pasach wyniku
    for ( int i = 0 ; i < n ; i++)  // wyznaczenie niebieskiej części wyniku
        for (int k = 0 ; k < n ; k++)  // wyznaczenie brązowej części wyniku
            for (int jj = j ; jj < j+r-1 ; jj++)
                C[i][jj] + = A[i][k] * B[k][jj] ;
```

Wiele faz obliczeń - zmniejszenie zakresu pętli wewnętrznej – łatwiejsza lokalność czasowa



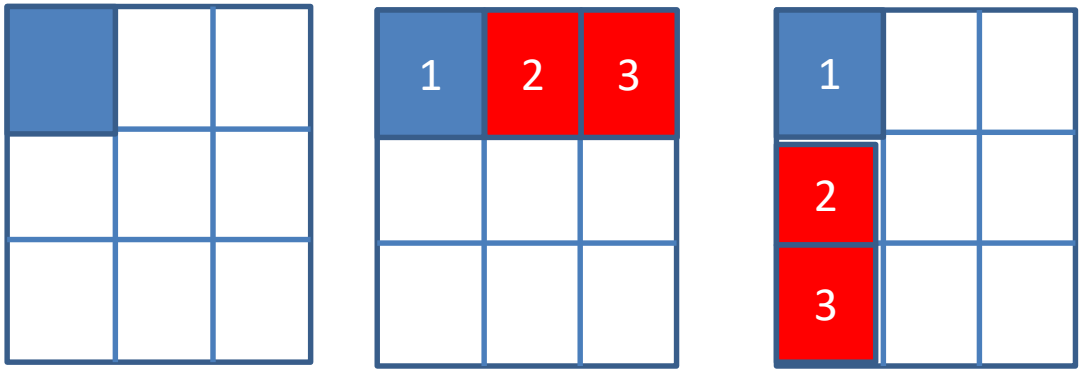
- Przy odpowiedniej wielkości r możliwa lokalność czasowa odwołań do danych w tablicy B - tablica B ograniczona B[*][jj:jj+r-1]
- Zmniejszenie wielkości fragmentów tablic, na podstawie których realizowane są obliczenia (w fazie przetwarzania) prowadzi do większej lokalności odwołań.
- Konieczne ponowne pobrania macierzy A – ile razy ?
- Macierz A użyta jednokrotnie w 3 wewnętrznych pętlach.
- Konieczność pobrania A w każdym etapie pętli zewnętrznej (n/r razy)

Wiele faz obliczeń -

zmniejszenie zakresu dwóch pętli wewnętrznych

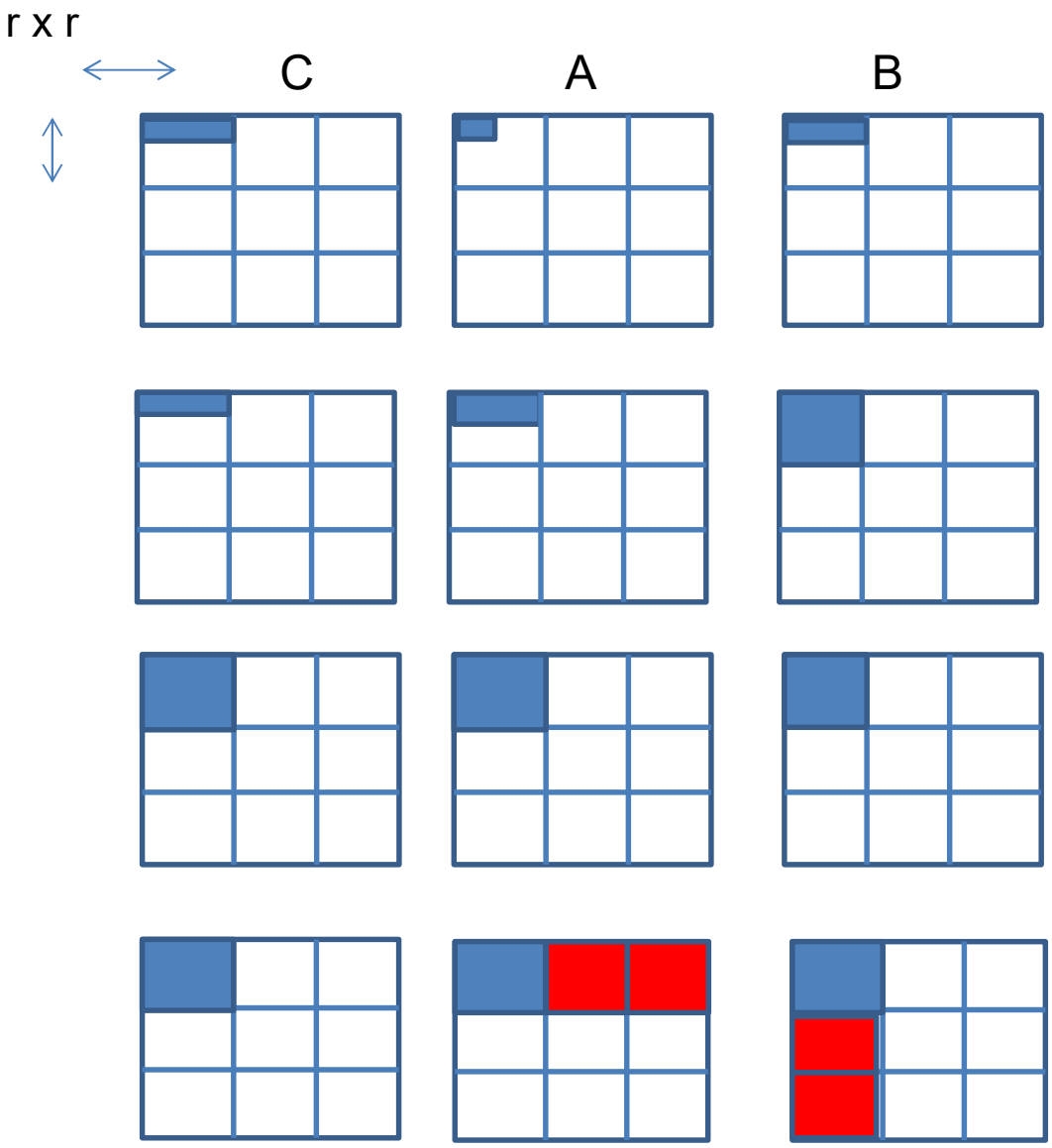
– łatwiejsza lokalność czasowa

- Dwuwymiarowy podział pracy - generacja zadań pozwalających na minimalizację ilości danych używanych przez proces w każdym etapie przetwarzania.



Dla k=3 Podział macierzy na k * k części i wynik kolejno dla każdej części powstaje w k=3 etapach z ilością danych etapu = 3* r * r gdzie r=N/k wielkość bloku.

Mnożenie macierzy – podział pracy dwuwymiarowy




```
for (int jj = j ; jj < j+r ; jj++)
    C[ii][jj] + = A[ii][kk] * B[kk][jj];
```

```
for (int kk = k ; kk < k+r ; kk++)
    for (int jj = j ; jj < j+r ; jj++)
        C[ii][jj] + = A[ii][kk] * B[kk][jj];
```

```
for ( int ii = i ; ii < i+r; ii++)
 for (int kk = k ; kk < k+r ; kk++)
   for (int jj = j ; jj < j+r ; jj++)
        C[ii][jj] + = A[ii][kk] * B[kk][jj];
```

```
for (int k = 0 ; k < n ; k+=r)
   for ( int ii = i ; ii < i+r; ii++)
      for (int kk = k ; kk < k+r ; kk++)
         for (int jj = j ; jj < j+r ; jj++)
            C[ii][jj] + = A[ii][kk] * B[kk][jj];
```

Mnożenie macierzy – podział pracy dwuwymiarowy metoda 6 pętlowa

```
for ( int i = 0 ; i < n ; i+=r) //wszystkie wiersze bloków
  for ( int j = 0 ; j < n ; j+=r) //kolejny wiersz bloków
    for (int k = 0 ; k < n ; k+=r) // wynik bloku RxR
      for ( int ii = i ; ii < i+r; ii++)//wynik częściowy blok
        for (int kk = k ; kk < k+r ; kk++)
          for (int jj = j ; jj < j+r ; jj++)
            C[ii][jj] + = A[ii][kk] * B[kk][jj];
```

Dla C[ii][jj], A[ii][kk], B[kk][jj] lokalność czasowa dostępów do danych przy założeniu, że wszystkie podmacierze A,B i C (A[i:i+r-1][k:k+r-1],B[k:k+r-1][j:j+r-1], C[i:i+r-1][j:j+r-1]) mieszczą się w pamięci podręcznej .

1. 3 pętle wewnętrzne służą do wyznaczenia wyniku częściowego dla fragmentu tablicy wynikowej (sum iloczynów elementów wierszy i kolumn fragmentów macierzy wejściowych),
2. czwarta pętla (po k) służy do uzupełnienia wyniku o pozostałe iloczyny wynikające z uwzględnienia kolejnych (branych po r) elementów wierszy i kolumn fragmentów macierzy wejściowych,
3. pętle piąta i szósta służą do wyznaczenia kolejnych kwadratowych (r) obszarów macierzy wynikowej.

Metoda 6 pętlowa
analiza lokalności czasowej

W ramach 4 wewnętrznych pętli:

- korzysta się z wielu tablic o wielkości r*r. Jedna tablica (część wyniku) jest potrzebna na każdym z etapów, tablice wejściowe (części tablicy A i B) jednocześnie są potrzebne w 2 egzemplarzach po jednej dla A i B w kolejnych etapach nowe części po r*r elementów.
- jednocześnie potrzebne są dla każdego z wątków przetwarzania 3 tablice o rozmiarze r*r. W pamięci podręcznej o rozmiarze M używanej przez jeden wątek zmieszczą się 3 tablice o rozmiarze
- $r <= (M/ts/3)^{1/2}$
- (ts rozmiar typu zmiennej, 3 - liczba tablic używanych przez jeden wątek).

- Dla przyjętej kolejności zagnieżdżenia podtablica B jest czytana wielokrotnie i musi być podobnie jak tablica C cały czas dostępna, podtablica A jest czytana tylko raz (wierszami) i faktycznie mogłaby (w aktualnie potrzebnym zakresie – bez potrzeby ponownego pobierania do pamięci podręcznej) zajmować jedną linię pamięci podręcznej przy efektywnym zarządzaniu pamięcią. Jednakże dla zapewnienia ciągłej obecności w pamięci wielokrotnie używanych fragmentów tablic C i B bezpieczniej również dla podtablicy A zarezerwować obszar w pamięci podręcznej równy r*r (wg wzoru powyżej).
- Iteracje zewnętrznych pętli to realizacja obliczeń powyższego typu dla innych wyników w oparciu o te same lub inne dane (tablice A i B). W całości przetwarzania każdy blok tablic A,B,C o rozmiarze r*r jest używany wielokrotnie w n/r etapach i tyle razy pobierany do pamięci podręcznej w najgorszym razie przy spełnieniu powyższej zależności na r.

Metoda 6 pętlowa równoległa -

analiza lokalności czasowej

W sytuacji, gdy przetwarzanie jest **realizowane równolegle** w zależności od sposobu podziału pracy mamy różne zależności:

- Wersja A - podział pracy przed pierwszą pętlą
  - Każdy z wątków wykonuje prace na podzbiorze bloków położonych obok siebie w poziomych pasach – liczba zadań do podziału wynosi N/R i powinna być dobrana dla zapewnienia zrównoważenia pracy systemu (jest to dodatkowy warunek dla określenia R – poza lokalnością czasową obliczaną dla wielu tablic R*R obliczanych równocześnie przez różne wątki)
- Wersja B - podział pracy przed pętlą czwartą
  - Każdy z wątków dzieli pracę w ramach wyliczania sum częściowych każdego wynikowego bloku c [R,R], każdy wątek liczy inną cześć wyników tablicy R*R w oparciu o wszystkie dane wejściowe niezbędne dla tego celu. Liczba zadań do podziału wynosi R. Podział pracy wprowadza wielokrotną synchronizację wątków.
- Wersja C - podział pracy przed pętlą trzecią
  - może powodować wyścig w dostępie do danych – możliwe jednoczesne uaktualnienia tych samych elementów zmiennych mogą doprowadzić do błędnych wyników ze względu na brak atomowości uaktualnienia zmiennej. Zapewnienie atomowości uaktualnienia wprowadza dalszą synchronizację wątków.

Metoda 6 pętlowa równoległa - analiza lokalności przestrzennej

Lokalność przestrzenna dostępu do danych wynika:

- z kolejności najbardziej wewnętrznych pętli programu,
- Z odległości w pamięci elementów tablicy położonych w sąsiednich wierszach (zależy od rozmiaru wiersza),
- Z liczby wierszy przetwarzanych na danym etapie obliczeń (wielokrotnie) wielkość parametru R.