Zadanie laboratoryjne 2

Zapewnienie lokalności dostępu do danych
na podstawie problemu znajdowania liczb pierwszych

Znajdowanie liczb pierwszych w zakresie liczb

- Metody:
  - **dzielenie** każdej badanej liczby przez mniejsze liczby pierwsze i badanie reszty z dzielenia
  - usuwanie wielokrotności - usuwanie ze zbioru badanych liczb - liczb będących wielokrotnością liczb pierwszych

Znajdowanie liczb pierwszych - dzielenie

- Jakie liczby pierwsze uwzględniać dla badanej liczby (bądź górnego zakresu badanego przedziału) n?
- Wystarczy znaleźć dla każdej liczby złożonej minimalny podzielnik: dla 35 – 5, dla 77 – 7, dla 121 – 11.
- Czy istnieje warunek ograniczający maksymalną wartość najmniejszego podzielnika liczby n (jak wyznaczyć obszar przeszukiwania dzielników)? Tak !

  Jest to maksymalna wartość najmniejszego podzielnika liczby złożonej n - wynosi ona $\mathbf{n^{1/2}}$.
- Aby znaleźć zatem liczby pierwsze $x_i \in <k,l>$ należy:
  - odrzucić liczby dzielące się bez reszty przez liczby pierwsze brane z przedziału $<2, x_i^{1/2}>$ **lub**
  - usunąć zbadanego przedziału liczby będące wielokrotnością liczb pierwszych z przedziału $<2, l^{1/2}>$

Znajdowanie liczb pierwszych – dodawanie - wykreślanie [wielokrotności] z tablicy

- Prezentacje procesu sekwencyjnego wykreślania

można zaobserwować korzystając ze strony:

http://www.hbmeyer.de/eratosiv.htm

Wykreślanie z **tablicy** – algorytm równoległy dla systemów z pamięcią współdzieloną

Z badanego zbioru (tablicy) usuwamy wielokrotności (jakie?) liczb pierwszych z przedziału <2,zakres górny$^{1/2}$>

Przykład dla zakresu : <2,65> liczba_pierwsza: jej wielokrotności

2 : 4,6,8,...64 3 : 9,15,21,27,33,39,45,51,57,63

5 : 25,35,55,65 7 : 49

- nie jest konieczna do rozpoczęcia obliczeń znajomość wszystkich liczb-pierwszych z przedziału <2,zakres górny$^{1/2}$> - wynik poprawny, możliwa dodatkowa praca

- kolejno pojawiające się liczby pierwsze mogą być wykorzystane dopiero później, gdyż wyznaczanie wielokrotności mniejszych liczb pierwszych odbywa się dla całego badanego przedziału i zajmuje stosunkowo dużo czasu.

Możliwe podejście z rozwiązaniem rekurencyjnym :

- aby rozwiązać problem dla <2,n>,
- rozwiązujemy problem dla <2, n$^{1/2}$ > i tak rekurencyjnie aż int (zakres górny$^{1/2}$)=2

Wykreślanie z tablicy
sekwencyjne i współbieżne

Podejścia równoległe do tego algorytmu polegają na podziale pracy:

- **Podejście domenowe:** procesy otrzymują fragment tablicy wykreśleń i całą tablicę liczb pierwszych (do pierwiastka z maksimum zakresu) lub

- **Podejście funkcyjne:** procesy otrzymują całą tablicę wykreśleń i fragment zbioru liczb pierwszych, których wielokrotności są usuwane

Problem stanowi udostępnienie wejściowego zbioru liczb pierwszych potrzebnego do generacji wielokrotności – przygotowanie tego zbioru sekwencyjne lub równolegle, rekurencyjne tą samą procedurą, synchronizacja etapów ?.

Problem stanowi określenie w jaki sposób dobrać wielkości i zakresy podzbiorów używanych w:

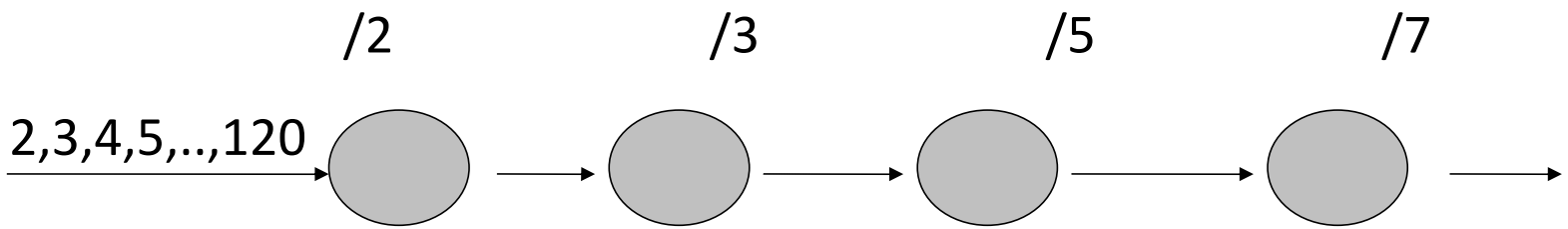
- podziale pracy dla wątków dla zapewnienia zrównoważenia wykorzystania systemu

- podziale pracy na etapy dla zapewnienia lokalności dostępu do wątków.

Jakie są efektywne sposoby podziału pracy: statyczne, dynamiczne, czy istnieją ogólne zasady efektywnego podziału pracy dla dowolnego systemu i dowolnej liczby procesorów?

Sito Eratostenesa – algorytm dla systemów z pamięciami prywatnymi

koncepcja podejścia funkcyjnego do podziału w architekturze z przekazywaniem komunikatów

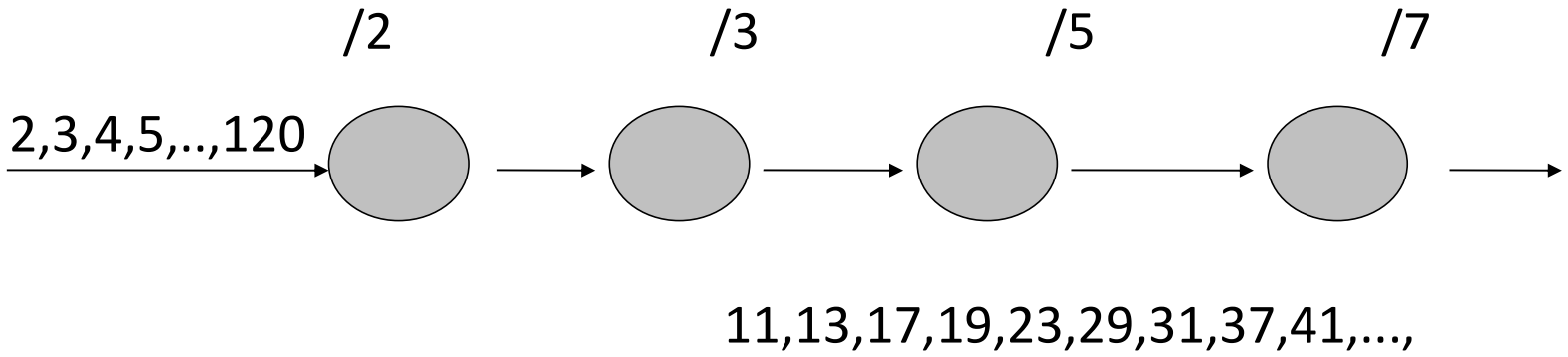


Okręgi oznaczają procesy z pamięcią prywatną (np. procesy uruchomione na różnych komputerach). Strzałki to kanały komunikacyjne umożliwiające wymianę informacji poprzez przesyłanie komunikatów. /2 oznacza proces dzielnika liczb przez 2.

Pierwsza liczba odebrana przez każdy z procesów jest traktowana jako liczba pierwsza i jako dzielnik następnych liczb . Liczby (oprócz pierwszej) dzielące się z resztą są przesyłane dalej. Wynik przetwarzania to liczby pierwsze rezydujące w procesach jako dzielniki (należy je przesłać na wyjście jeśli są rozwiązaniem problemu) oraz liczby pojawiające się na wyjściu systemu (wyjście ostatniego procesora).

Sito Eratostenesa – algorytm dla systemów z pamięciami prywatnymi

koncepcja podejścia funkcyjnego do podziału w architekturze z przekazywaniem komunikatów



- Liczba procesów niezbędnych dla badanego zakresu <n,k> jest równa liczbie liczb pierwszych w zakresie od <2, $k^{1/2}$> około $2k^{1/2}/\ln k$
- Aby zbadać liczby pierwsze w zakresie <n,k> należy na wejście potoku przetwarzania podać liczby <2,.., $k^{1/2}$> i <n,k> .

Uwaga: znaczenie podejścia równoległego jest tylko koncepcyjne – niska efektywność ze względu na dużą liczbę komunikacji (i synchronizacji).

Kody przykładowe do poszukiwania liczb pierwszych

Liczby pierwsze wyznaczane sekwencyjnie przez dzielenie w zakresie <m,n> [k1]

```
bool* result = (bool*)malloc((n - m + 1) * sizeof(bool));
memset(result, true, (n - m + 1) * sizeof(bool));

bool* primeArray = (bool*)malloc((sqrt(n) + 1) * sizeof(bool));
memset(primeArray, true, (sqrt(n) + 1) * sizeof(bool));

for (int i = 2; i * i <= n; i++) {
        for (int j = 2; j * j <= i; j++) {
                if (primeArray[j] == true && i % j == 0) { primeArray[i] = false; break;
                                                }}}

for (int i = m; i <= n; i++){
        for (int j = 2; j * j <= i; j++){
                if (primeArray[j] == true && i % j == 0) {
                                                result[i - m] = false;break;
                                }}}
```

Liczby pierwsze wyznaczane równolegle przez dzielenie w zakresie <m,n> [k2]

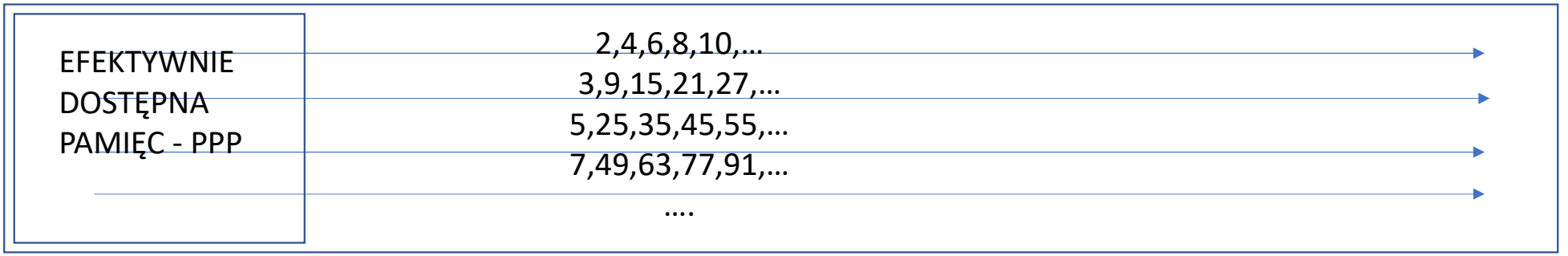
```
bool* result = (bool*)malloc((n - m + 1) * sizeof(bool));
memset(result, true, (n - m + 1) * sizeof(bool));

bool* primeArray = (bool*)malloc((sqrt(n) + 1) * sizeof(bool));
memset(primeArray, true, (sqrt(n) + 1) * sizeof(bool));

for (int i = 2; i * i <= n; i++) {
        for (int j = 2; j * j <= i; j++) {
                if (primeArray[j] == true && i % j == 0) { primeArray[i] = false; break;
                                }}}

#pragma omp parallel
{
#pragma omp for ? Jaki podział pracy ?
for (int i = m; i <= n; i++){
        for (int j = 2; j * j <= i; j++){
                if (primeArray[j] == true && i % j == 0) { result[i - m] = false;break;
}}}}
```

Sito sekwencyjne bez lokalności dostępu do danych [k3]

```
for (int i = 2; i*i*i*i <= n; i++) {
        if (primeArray[i] == true) {
                for (int j = i * i; j*j <= n; j+=i) {primeArray[j] = false;}
}}
for (int i = 2; i*i <= n; i++){
        if (primeArray[i]) {
                int firstMultiple = (m / i);
                if (firstMultiple <= 1) {firstMultiple = i + i;}
                else
                if (m % i) { firstMultiple = (firstMultiple * i) + i;}
                        else {firstMultiple = (firstMultiple * i);}
                for (int j = firstMultiple; j <= n; j+=i) { result[j-m] = false;}
}}
```

Sito sekwencyjne bez lokalności dostępu do danych

```
for (int i = 2; i*i*i*i <= n; i++) {
        if (primeArray[i] == true) {
                for (int j = i*i; j*j <= n; j+=i) {primeArray[j] = false;}
}}
for (int i = 2; i*i <= n; i++){
        if (primeArray[i]) {
                …
                for (int j = firstMultiple; j <= n; j+=i) { result[j-m] = false;}
}}
```



Sito równoległe funkcyjne bez lokalności dostępu do danych [k4]

```
for (int i = 2; i*i*i*i <= n; i++) {
        if (primeArray[i] == true) {
                for (int j = i*i; j*j <= n; j+=i) {primeArray[j] = false;}
}}
#pragma omp parallel for ? Jaki podział pracy ?
for (int i = 2; i*i <= n; i++){
        if (primeArray[i]) {
                int firstMultiple = (m / i);
                if (firstMultiple <= 1) {firstMultiple = i + i;}
                else
                if (m % i) { firstMultiple = (firstMultiple * i) + i;}
                        else {firstMultiple = (firstMultiple * i);}
                for (int j = firstMultiple; j <= n; j+=i) { result[j-m] = false;}
                                }}
```

Sito równoległe funkcyjne bez lokalności dostępu do danych [k4a]

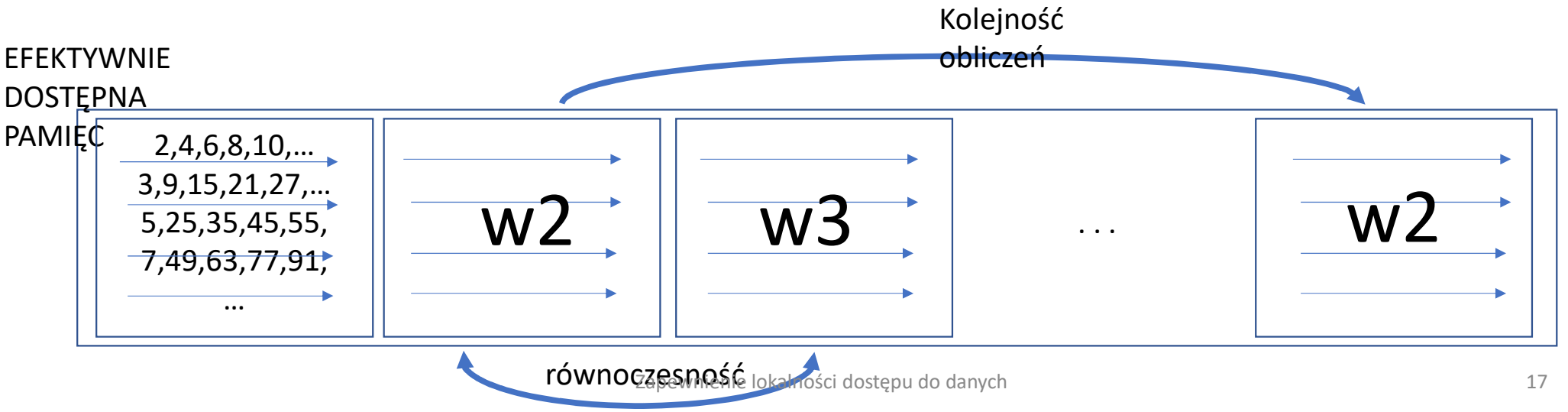
```
for (int i = 2; i*i*i*i <= n; i++) {
       if (primeArray[i] == true) {
              for (int j = i*i; j*j <= n; j+=i) {primeArray[j] = false;}
}}
#pragma omp parallel for ? Jaki podział pracy ?
for (int i = 2; i*i <= n; i++){
       if (primeArray[i]) {
              …
              for (int j = firstMultiple; j <= n; j+=i) {
                     if (result[j-m]) result[j-m] = false;} FS!
                            }}
```

Sito równoległe domenowe z potencjalną lokalnością dostępu do danych [k5]

```
int blockSize = ?????;  int numberOfBlocks = (n - m) / blockSize;
if ((n - m) % blockSize != 0) { numberOfBlocks++;}
#pragma omp parallel for ?
for (int i = 0; i < numberOfBlocks; i++) {
        int low = m + i * blockSize; int high = m + i * blockSize + blockSize;
        if (high > n) {high = n;}
        for (int j = 2; j * j<= high; j++) {
                if (primeArray[j]) {
                        int firstMultiple = (low / j);
                        if (firstMultiple <= 1) {firstMultiple = j + j;}
                        else
                                if (low % j) {firstMultiple = (firstMultiple * j) + j;}
                                else {firstMultiple = (firstMultiple * j);}
                        for (int k = firstMultiple; k <= high; k += j) {result[k - m] = false;}
}}}
```

Sito równoległe domenowe z potencjalną lokalnością dostępu do danych

```
int blockSize = ?????;  int numberOfBlocks = (n - m) / blockSize;
if ((n - m) % blockSize != 0) { numberOfBlocks++;}
#pragma omp parallel for ?????
for (int i = 0; i < numberOfBlocks; i++) {
        … (low,high)
        for (int j = 2; j * j<= high; j++) {
                if (primeArray[j]) {
                        … (firstMultiple)
                        for (int k = firstMultiple; k <= high; k += j) {result[k - m] = false;}
}}}
```



Zadania projektowe i eksperymentalne (część 1)

1. Przeanalizować kody k1-k5 znajdowania liczb pierwszych w zakresie <m,n>
2. Sprawdzić poprawność obliczeń dla wariantów kodów poprzez wykonanie obliczeń i porównanie wyników osiągniętych różnymi metodami.
3. Zbadać prędkość obliczeń (Mliczb/sek) dla poszczególnych wariantów kodu k1-k5 dla zakresów <2,max> <max/2,max> <2,max/2> dla max = 10 exp 8
4. Wybrać na podstawie znajomości problemu i charakterystyki algorytmu wskazać interesujące rodzaje podziału pracy dla wątków w kodzie [k2] i zbadać prędkość przetwarzania w zależności od wybranego sposobu szeregowania. Uzasadnić różnice w prędkościach przetwarzania.
5. Przygotować sito sekwencyjne z lokalnością dostępu do danych [k3a] w oparciu o koncepcję lokalnego sita domenowego [k7].
6. Określić na podstawie parametrów procesora potencjalnie efektywne rozmiary bloku danych dla sita sekwencyjnego z lokalnością dostępu do danych.
7. Dla wybranych w punkcie 6 wielkości bloków danych porównać prędkości przetwarzania dla przygotowanego kodu lokalnego sita sekwencyjnego i kodu sita [k3] dla zakresów danych z pkt 3.

Zadanie projektowe i eksperymentalne (część 2)

8. Wykonać eksperyment dla kodów sita funkcyjnego wersje kodów k4, k4a badając wpływ zastosowanego rodzaju szeregowania pętli na prędkość przetwarzania.
9. Określić na podstawie parametrów procesora potencjalnie efektywne rozmiary bloku danych dla równoległego sita domenowego.
10. Dla ustalonych w punkcie 3 i 9 parametrów wyznaczyć prędkości przetwarzania dla równoległego sita domenowego w funkcji wybranych charakterystycznych rodzajów szeregowania pętli.
11. W oparciu o wyniki eksperymentów wykonanych punktach 3,4,7,8,10 przygotować tabelaryczne zestawienie prędkości przetwarzania i uzyskanego przyspieszenia w zależności od: zastosowanej metody przetwarzania: k1,k2,k3,k3a,k4,k4a,k5 oraz sposobu szeregowania pętli i wielkości bloku danych wykorzystanych w etapie przetwarzania.
12. Jako przyspieszenie przetwarzania równoległego określić iloraz czasu przetwarzania równoległego i najszybszego przetwarzania sekwencyjnego (kod k3a)
13. Korzystając z programu Vtune określić stopień efektywności wykorzystania zasobów systemu obliczeniowego. Realizując eksperymenty obliczeniowe analizujące przetwarzanie kodu wskazać na występujące podczas poszczególnych uruchomień ograniczenia wykorzystania zasobów. W opracowaniu wyników eksperymentu określić prędkość przetwarzania, stopień zrównoważenia przetwarzania, wielkości poszczególnych typów ograniczeń wykorzystania zasobów procesora. We wnioskach omawiających wyniki eksperymentu obliczeniowego należy powiązać je z użytym algorytmem, sposobem podziału pracy i stopniem lokalności dostępu do danych
14. Wszystkie eksperymenty należy zrealizować przy użyciu jednakowego sposobu kompilacji kodu (efektywnego – O3) oraz zapewnić (jeśli będzie to konieczne) wystarczający dla uzyskania wiarygodności wyników czas zbierania metryk jakości przetwarzania poprzez wielokrotne przeprowadzenie obliczeń podczas jednego eksperymentu w Vtune.
15. Sprawozdanie z realizacji projektu należy wykonać zgodnie z wymaganiami zawartymi w oddzielnym pliku (sprawozdanie_liczby_lierwsze24).

Sprawozdanie z realizacji zadania - zawartość