

Obliczenia równoległe – możliwości i ograniczenia

Przemysław Stpiczyński

Zakład Informatyki UMCS, Lublin

Dlaczego potrzebujemy obliczeń równoległych ?

Odpowiedź: istnieje **rzeczywista potrzeba rozwiązywania problemów obliczeniowych dużej skali** pojawiających się w różnych dziedzinach nauki i techniki:

- prognozowanie zjawisk meteorologicznych i geologicznych,
- testowanie broni nuklearnej (program ASCI),
- chemia kwantowa,
- biologia obliczeniowa,
- astronomia i astrofizyka,
- kryptografia,
- algorytmy aproksymacyjne dla problemów \mathcal{NP} -zupełnych,
- poszukiwanie cywilizacji pozaziemskich (projekt SETI),
- ...

Czas obliczeń dla problemów o różnej złożoności: procesor Pentium IV 3GHz

funkcja złożoności	$n = 10$	$n = 60$
n	0.00000001 sec.	0.00000006 sec.
n^3	0.000001 sec.	0.000216 sec.
n^5	0.0001 sec.	0.7776 sec.
2^n	0.000001 sec.	336 lat
3^n	0.000059 sec.	1.3×10^{10} stuleci

Wpływ wzrostu szybkości obliczeń na rozmiar problemu rozwiązywanego w ciągu godziny

funkcja złożoności	Pentium IV	100 razy	1000 razy
n	N_1	$100N_1$	$1000N_1$
n^3	N_2	$4.64N_2$	$10N_2$
n^5	N_3	$2.5N_3$	$3.98N_3$
2^n	N_4	$N_4 + 6.64$	$N_4 + 9.97$
3^n	N_5	$N_5 + 4.19$	$N_5 + 6.29$

Idea obliczeń równoległych

- 1624: Henry Briggs publikuje dzieło *Arithmetica Logarithmica* zawierające logarytmy 30000 liczb naturalnych od 1 do 20000 oraz od 90000 do 100000 obliczone do 14 miejsc. Proponuje, aby brakujące logarytmy były wyznaczone przez zespół rachmistrzów wykonujących **równolegle** swoje *zadania obliczeniowe*.
- 1842: Generał L. F. Menabrea pisze artykuł *Sketch of the Analytical Engine* opisujący możliwości maszyny zaprojektowanej przez Charlesa Babbage. Proponuje:

When a long series of identical computations is to be performed, such as those required for the formation of numerical tables, the machine can be brought into play so as to give several results at the same time, which will greatly abridge the whole amount of the processes.
- 1974: D. L. Slotnick buduje komputer Illiac IV zawierający 64 procesory obliczeniowe oraz jeden procesor sterujący o łącznej rzeczywistej mocy obliczeniowej 15 Mflops.
- 2002: Powstaje *Earth Simulator* - największy komputer wyposażony w 5120 procesorów o łącznej rzeczywistej mocy obliczeniowej 35.86Tflops.

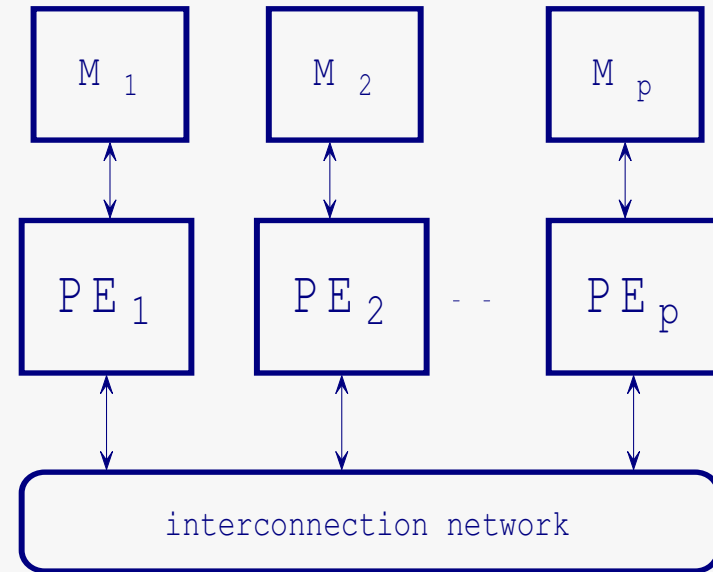
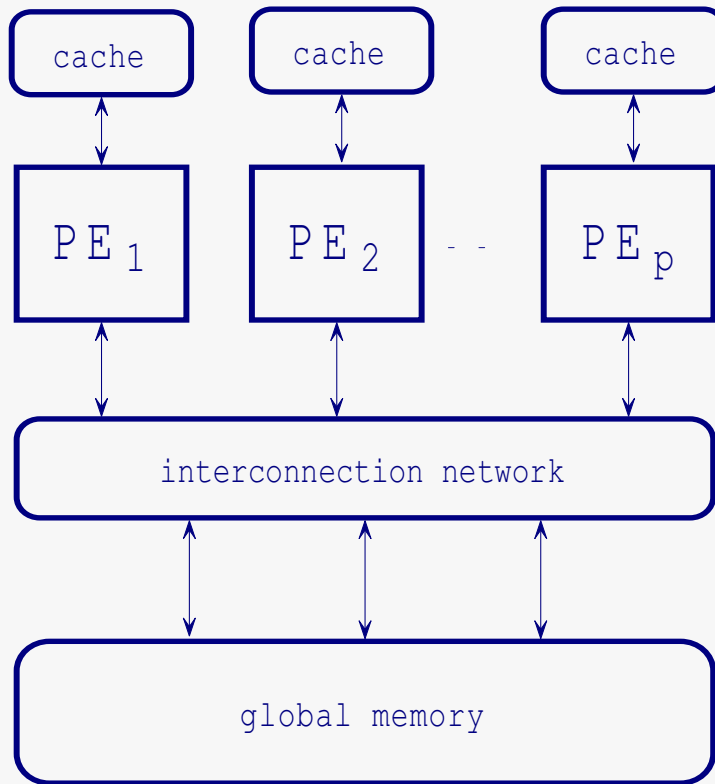
Klasyfikacja komputerów (Flynn, 1966)

- **SISD** – single instruction stream/single data stream: komputery działające w oparciu o model von Neumanna
- **MISD** – multiple instruction streams/single data stream: nie powstały komputery należące do tej klasy
- **SIMD** – single instruction stream/multiple data streams: tablice procesorów, procesory wektorowe
- **MIMD** – multiple instruction streams/multiple data streams: większość współczesnych komputerów równoległych

Klasa MIMD może być podzielona na dwie podklasy:

- **shared memory** – komputery równoległe z *pamięcią wspólną*,
- **distributed memory** – komputery równoległe z *pamięcią rozproszoną* oraz klastry (gridy) obliczeniowe.

Komputery klasy MIMD



Równoległość wewnątrz pojedynczego procesora:

potokowość i wektorowość

Stage	Cycle					
	1	2	3	4	5	6
fetch operands	A[1], B[1]	A[2], B[2]	A[3], B[3]	A[4], B[4]	A[5], B[5]	A[6], B[6]
adjust exponents		A[1], B[1]	A[2], B[2]	A[3], B[3]	A[4], B[4]	A[5], B[5]
execute multiplication			A[1]*B[1]	A[2]*B[2]	A[3]*B[3]	A[4]*B[4]
normalize result				A[1]*B[1]	A[2]*B[2]	A[3]*B[3]
store result					A[1]*B[1]	A[2]*B[2]

Ograniczenia mocy obliczeniowej

- **Pamięć jest wolniejsza niż procesor.** Rozwiązania: podział pamięci na banki oraz stosowanie pamięci podręcznej.
- **Efektywność dostępu wielu procesorów do wspólnej pamięci spada wraz ze wzrostem liczby procesorów.** Rozwiązanie: budowa specjalnych przełączników organizujących „sprzętowo” dostęp do wspólnej pamięci oraz ograniczanie liczby procesorów.
- W przypadku komputerów z pamięcią rozproszoną szybkość obliczeń zależy od **wydajności sieci połączeń między procesorami**

Metody programowania komputerów równoległych

- użycie kompilatorów optymalizujących
- zastosowanie odpowiednich podprogramów z bibliotek
 - Linpack, Eispack, BLAS Level 1
 - LAPACK, BLAS Level 2 & 3
 - PLAPACK (równoległa wersja biblioteki LAPACK)
 - BLACS, PBLAS, ScaLAPACK (obliczenia na komputerach z pamięcią rozproszoną i klastrach)
- tworzenie programów przy wykorzystaniu OpenMP, MPI (Message Passing Interface) or PVM (Parallel Virtual Machine) poprzez zrównoleglanie istniejących algorytmów
- projektowanie nowych algorytmów wykorzystujących pamięć podręczną, które mogą być zrównoleglane.

Analiza wykonania programu

Szybkość wykonania programów na komputerach jest podawana w milionach operacji zmiennopozycyjnych na sekundę (Mflops)

$$r = \frac{N}{t} \text{ Mflops,}$$

gdzie N oznacza liczbę operacji zmiennopozycyjnych wykonanych w czasie t mikrosekund.

Czas wykonania programu spełnia zależność

$$t = \frac{N}{r} \text{ microsec.}$$

Dodatkowo wprowadza się jednostki: Gflops (gigaflops) = 10^9 , Tflops (teraflops) = 10^{12} oraz Pflops (petaflops) = 10^{15} operacji zmiennopozycyjnych na sekundę.

Prawo Amdahla

Niech f będzie częścią programu (N operacji) wykonywaną z szybkością V , zaś część $1 - f$ z szybkością S ($V \gg S$). Wówczas otrzymujemy czas wykonywania obliczeń

$$t = f \frac{N}{V} + (1 - f) \frac{N}{S} = N \left(\frac{f}{V} + \frac{1 - f}{S} \right)$$

szybkość obliczeń

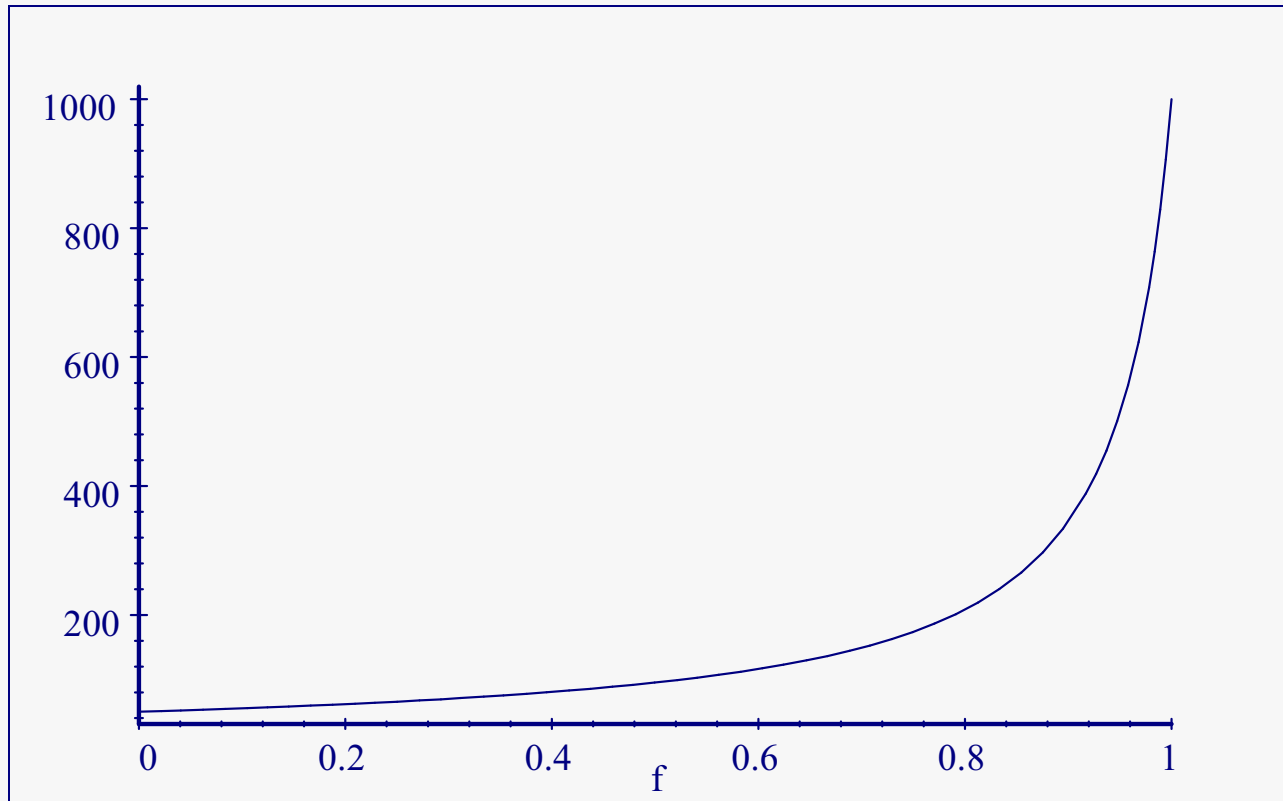
$$r = \frac{N}{t} = \frac{1}{\frac{f}{V} + \frac{(1-f)}{S}}$$

ograniczenie na przyspieszenie

$$\frac{t_s}{t_v} < \frac{N}{S} \cdot \frac{S}{N(1 - f)} = \frac{1}{1 - f}.$$

Prawo Amdahla (przykład)

Dla $V = 1000$ Mflops oraz $S = 50$ Mflops, szybkość wykonania programu (Mflops) w zależności od wartości f :



Prawo Amdahla (obliczenia równoległe)

Przyspieszenie s_p względem algorytmu sekwencyjnego uzyskane na p procesorach wyraża się wzorem

$$s_p = \frac{t_1}{t_p}$$

gdzie t_j oznacza czas wykonania algorytmu na j procesorach. Załóżmy, że część f algorytmu może być podzielona idealnie między p procesorów, zaś pozostałe $1 - f$ będzie wykonywane na jednym procesorze. Czas wykonania algorytmu na p procesorach wyraża się wzorem:

$$t_p = f \frac{t_1}{p} + (1 - f)t_1 = \frac{t_1(f + (1 - f)p)}{p},$$

zaś przyspieszenie

$$s_p = \frac{p}{f + (1 - f)p}.$$

Z uwagi na $f < 1$, otrzymujemy

$$s_p < \frac{1}{1 - f}$$

Linpack, Eispack & Basic Linear Algebra Subprograms (BLAS Level 1)

BLAS Level 1:

- $y \leftarrow \alpha x + y$, $x \leftarrow \alpha x$, $y \leftarrow x$, $y \leftrightarrow x$, $dot \leftarrow x^T y$, $nrm2 \leftarrow \|x\|_2$, $asum \leftarrow \|re(x)\|_1 + \|im(x)\|_1$.

Biblioteki podprogramów rozwiązujących zagadnienia numeryczne algebry liniowej:

- Linpack - podprogramy do rozwiązywania układów równań liniowych
- Eispack - podprogramy do rozwiązywania zagadnienia własnego

Basic Linear Algebra Subprograms (BLAS Level 2 & 3)

Level 2:

- matrix-vector products: $y \leftarrow \alpha Ax + \beta y$, $y \leftarrow \alpha A^T x + \beta y$
- rank-1 update of a general matrix: $A \leftarrow \alpha xy^T + A$
- rank-1 and rank-2 update of a symmetric matrix: $A \leftarrow \alpha xx^T + A$, $A \leftarrow \alpha xy^T + \alpha yx^T + A$,
- multiplication by a triangular matrix: $x \leftarrow Tx$, $x \leftarrow T^T x$,
- solving a triangular system of equations: $x \leftarrow T^{-1}x$, $x \leftarrow T^{-T}x$.

Level 3:

- matrix-matrix products: $C \leftarrow \alpha AB + \beta C$, $C \leftarrow \alpha A^T B + \beta C$, $C \leftarrow \alpha AB^T + \beta C$,
 $C \leftarrow \alpha A^T B^T + \beta C$
- rank- k and rank- $2k$ update of a symmetric matrix: $C \leftarrow \alpha AA^T + \beta C$, $C \leftarrow \alpha A^T A + \beta C$,
 $C \leftarrow \alpha A^T B + \alpha B^T A + \beta C$, $C \leftarrow \alpha AB^T + \alpha BA^T + \beta C$,
- multiplication by a triangular matrix: $B \leftarrow \alpha TB$, $B \leftarrow \alpha T^T B$, $B \leftarrow \alpha BT$, $B \leftarrow \alpha BT^T$,
- solving a triangular system of equations: $B \leftarrow \alpha T^{-1}B$, $B \leftarrow \alpha T^{-T}B$, $B \leftarrow \alpha BT^{-1}$,
 $B \leftarrow \alpha BT^{-T}$.

Zalety użycia wyższych poziomów BLAS-u

BLAS	memory	flops	ratio
$y \leftarrow \alpha x + y$	$3n$	$2n$	3 : 2
$y \leftarrow \alpha Ax + \beta y$	$mn + n + 2m$	$2m + 2mn$	1 : 2
$C \leftarrow \alpha AB + \beta C$	$2mn + mk + kn$	$2mkn + 2mn$	2 : n

Algorytmy mnożenia macierzy

BLAS 3:

$$C \leftarrow \alpha AB + \beta C$$

BLAS 2:

$$C_{*j} \leftarrow \alpha AB_{*j} + \beta C_{*j}, \quad \text{dla } j = 1, \dots, n$$

BLAS 1:

for $j = 1, \dots, n$

$$C_{*j} \leftarrow \beta C_{*j}$$

$$C_{*j} \leftarrow C_{*j} + (\alpha b_{ij}) A_{*i} \quad \text{for } i = 1, \dots, k$$

Simple:

for $j = 1, \dots, n$

for $i = 1, \dots, m$

$$t \leftarrow 0$$

$$t \leftarrow t + a_{il} b_{lj} \quad \text{for } l = 1, \dots, k$$

$$c_{ij} \leftarrow \beta c_{ij} + \alpha t$$

Algorytmy mnożenia macierzy (wyniki eksperymentów)

- Pentium III 866MHz & Pentium IV 3GHz

	Mflops	sec.
simple	93.98	21.28
BLAS 1	94.65	21.13
BLAS 2	342.46	5.83
BLAS 3	1398.60	1.43

	Mflops	sec.
simple	282.49	7.08
BLAS 1	1162.79	1.72
BLAS 2	1418.43	1.40
BLAS 3	7692.30	0.26

- Cray X1 (1-msp)

	Mflops	sec.
simple	7542.29	0.27
BLAS 1	587.41	3.40
BLAS 2	7259.48	0.28
BLAS 3	16369.89	0.12

Blokowy algorytm mnożenia macierzy

Operacja $C \leftarrow \alpha AB + \beta C$ może być zapisana w postaci blokowej

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \alpha \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} + \beta \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$$

Stąd

$$C_{11} \leftarrow \alpha A_{11} B_{11} + \beta C_{11} \quad /1/$$

$$C_{11} \leftarrow \alpha A_{12} B_{21} + C_{11} \quad /2/$$

$$C_{12} \leftarrow \alpha A_{11} B_{12} + \beta C_{12} \quad /3/$$

$$C_{12} \leftarrow \alpha A_{12} B_{22} + C_{11} \quad /4/$$

$$C_{21} \leftarrow \alpha A_{22} B_{21} + \beta C_{21} \quad /5/$$

$$C_{21} \leftarrow \alpha A_{21} B_{11} + C_{21} \quad /6/$$

$$C_{22} \leftarrow \alpha A_{22} B_{22} + \beta C_{22} \quad /7/$$

$$C_{22} \leftarrow \alpha A_{21} B_{12} + C_{22} \quad /8/$$

Blokowy rozkład Choleskiego

$$A = \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix} = \begin{pmatrix} L_{11} & & \\ L_{21} & L_{22} & \\ L_{31} & L_{32} & L_{33} \end{pmatrix} \begin{pmatrix} L_{11}^T & L_{21}^T & L_{31}^T \\ & L_{22}^T & L_{32}^T \\ & & L_{33}^T \end{pmatrix} = LL^T$$

$$A_{11} = L_{11}L_{11}^T$$

$$A_{21} = L_{21}L_{11}^T$$

$$A_{31} = L_{31}L_{11}^T$$

$$A_{12} = L_{11}L_{21}^T$$

$$A_{22} = L_{21}L_{21}^T + L_{22}L_{22}^T$$

$$A_{32} = L_{31}L_{21}^T + L_{32}L_{22}^T$$

$$A_{13} = L_{11}L_{31}^T$$

$$A_{23} = L_{21}L_{31}^T + L_{22}L_{32}^T$$

$$A_{33} = L_{31}L_{31}^T + L_{32}L_{32}^T + L_{33}L_{33}^T$$

Przykład

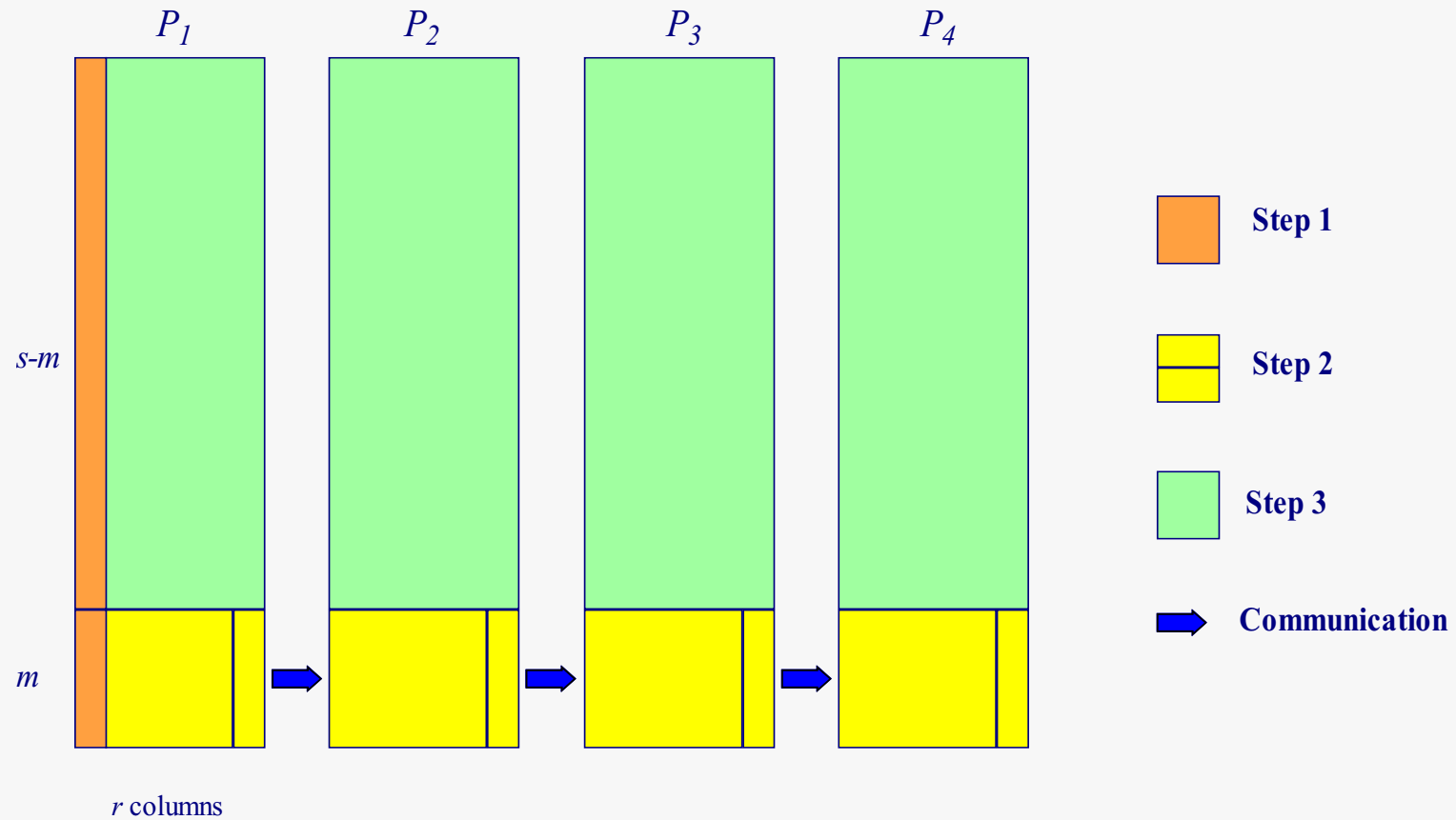
Liniowe równanie rekurencyjne rzędu m o stałych współczynnikach

$$x_k = \begin{cases} 0 & \text{dla } k \leq 0 \\ f_k + \sum_{j=1}^m a_j x_{k-j} & \text{dla } 1 \leq k \leq n \end{cases}$$

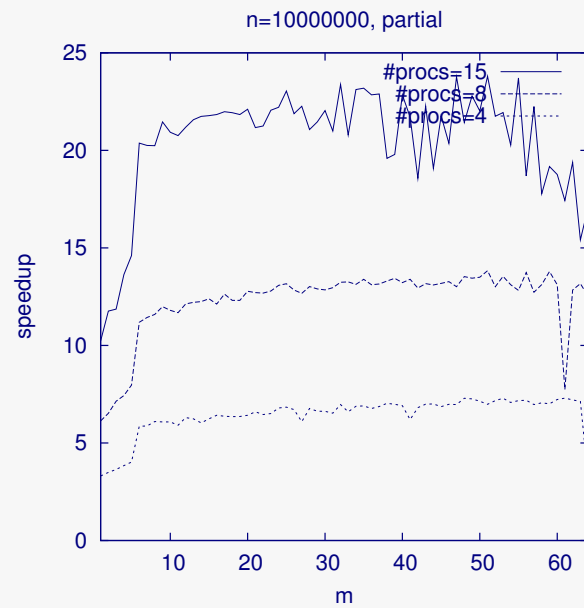
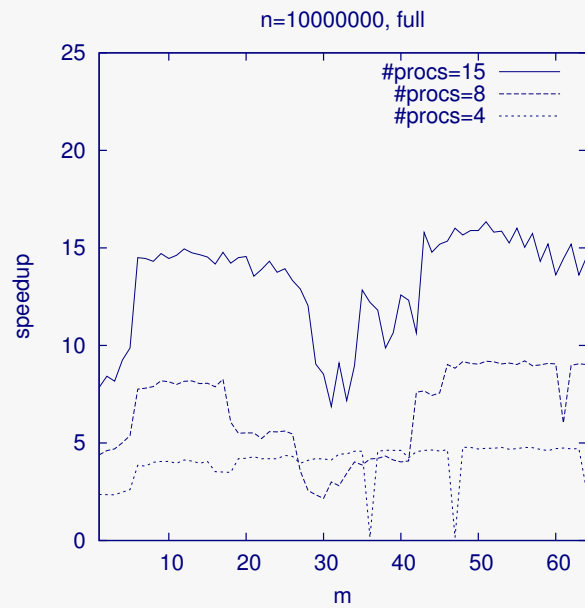
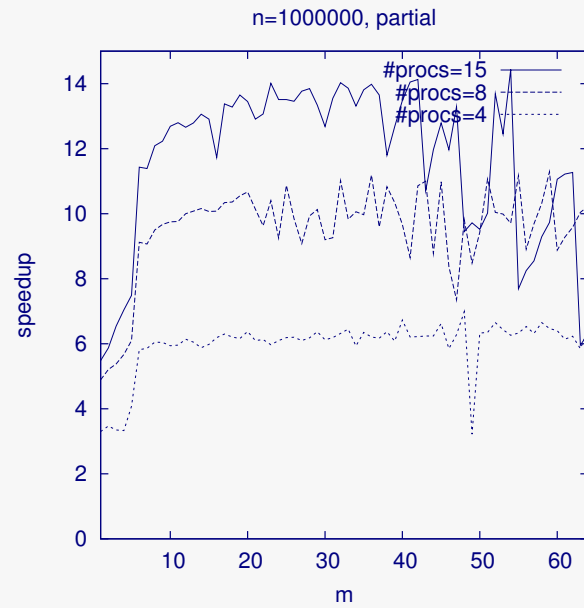
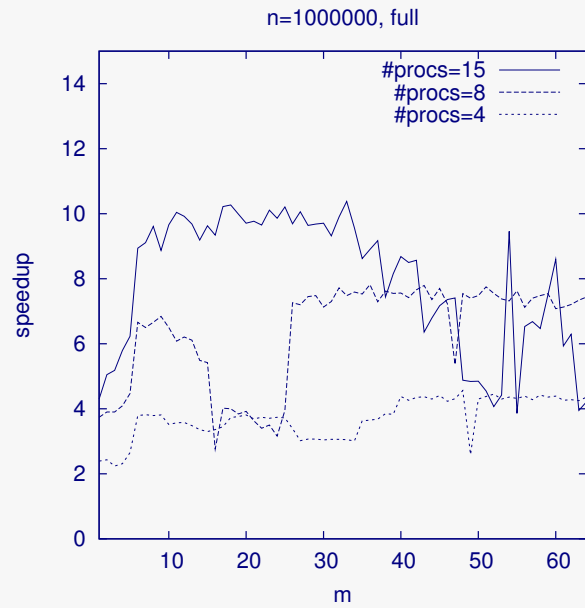
Podprogram wyznaczający rozwiązanie

```
subroutine recurt(a,f,x,n,m)
real f(*),x(*),a(*)
integer n,m,j,k
do k=1,n
  x(k)=f(k)
  do j=1,min(m,k-1)
    x(k)=x(k)+x(k-j)*a(j)
  end do
end do
end
```

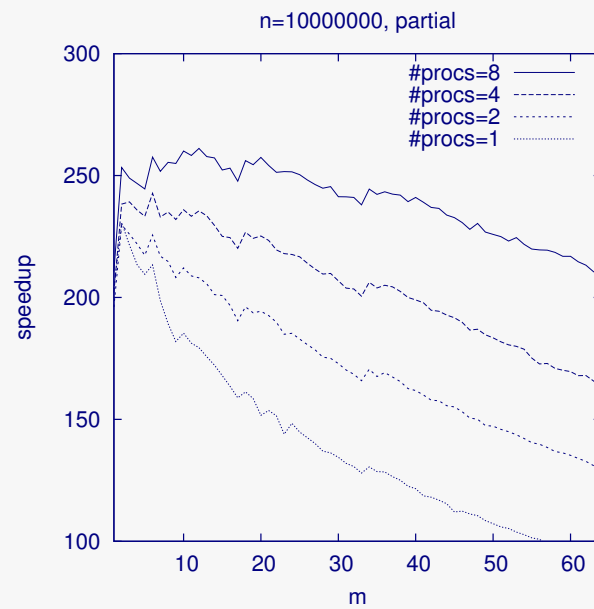
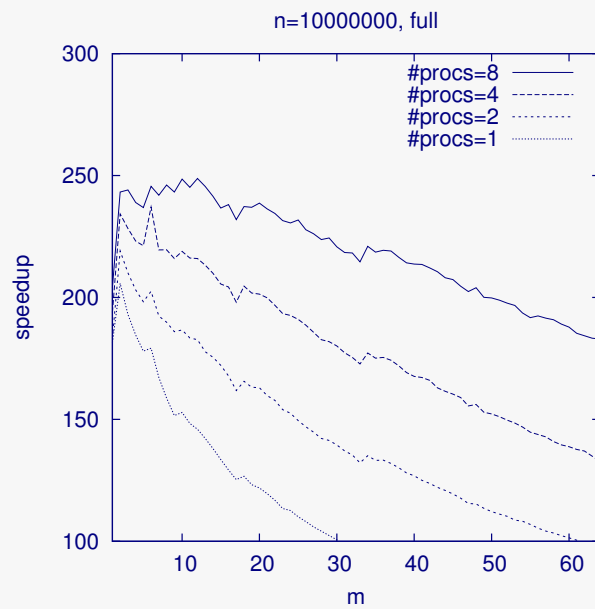
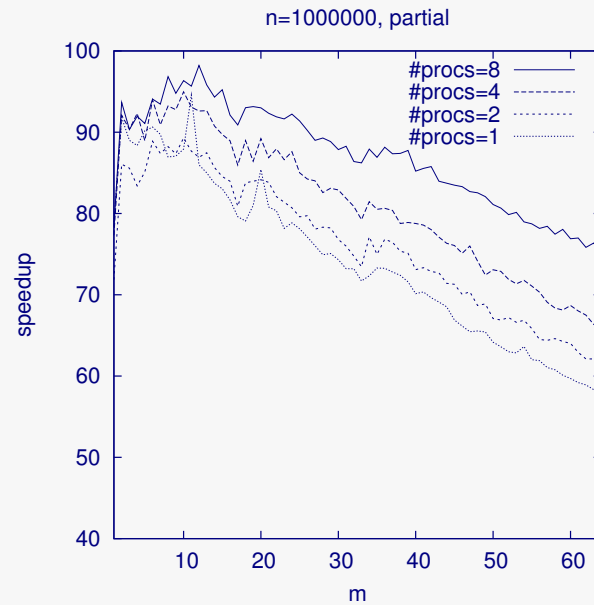
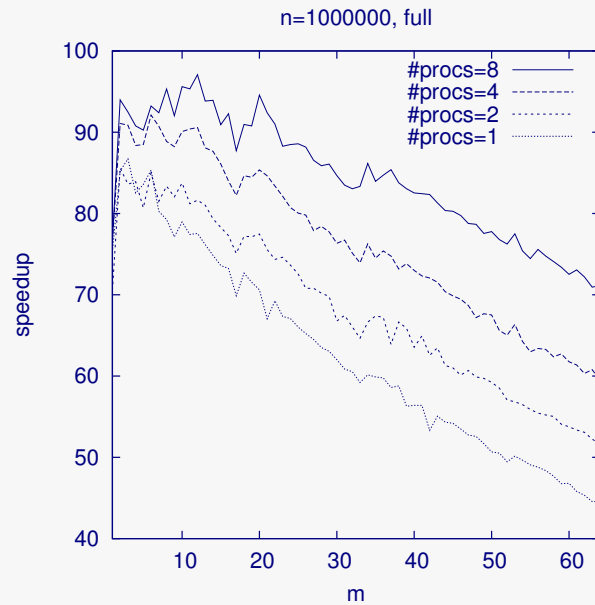
Algorytm rozwiązania wykorzystujący BLAS poziomu 2 & 3



Linux cluster



Cray X1



Równoległość obliczeń a złożoność obliczeniowa

Kilka najważniejszych faktów:

- Podstawowy model obliczeń równoległych PRAM zakłada istnienie nieograniczonej liczby procesorów (stanowi uogólnienie modelu RAM). Obliczenia rozpoczynają się od uaktywnienia pierwszego procesora. W każdym kolejnym kroku procesor może wykonać standardową operację lub uaktywnić kolejny procesor.
- Klasa problemów rozwiązywalnych przez model PRAM w czasie wielomianowym jest równa klasie **PSPACE** - klasie problemów rozwiązywalnych przez DTM w przestrzeni wielomianowej (do klasy **PSPACE** należą między innymi problemy NP-zupełne).
- Wiele problemów z klasy **P** można rozwiązać na maszynie PRAM w czasie $(\log n)^{O(1)}$, czyli polylogarytmicznym.