Analiza efektywności przetwarzania współbieżnego

Wykład: Przetwarzanie Równoległe

Politechnika Poznańska

Rafał Walkowiak

Grudzień 2015

Źródła kosztów przetwarzania współbieżnego

- interakcje między procesami – komunikacja synchronizacja;
- bezczynność – przyczyny: brak zrównoważenia obciążenia (brak możliwości oceny czasu przetwarzania zadań), obecność części sekwencyjnej przetwarzania, konieczność synchronizacji;
- dodatkowe obliczenia – algorytm równoległy dla swej efektywności wymaga innej konstrukcji od optymalnego algorytmu sekwencyjnego, np. powielenie obliczeń zamiast kosztownej komunikacji;

Miary efektywności systemów współbieżnych

- ***System współbieżny*** - połączenie algorytmu (programu) i architektury równoległej, w której jest on implementowany
- Czas przetwarzania (ang. runtime)
  - T_p – czas przetwarzania równoległego – od momentu rozpoczęcia przetwarzania równoległego do momentu zakończenia przetwarzania przez ostatnią jednostkę przetwarzającą
  - T_s – czas przetwarzania sekwencyjnego
- Koszt zrównoleglenia (ang. parallel overhead) – czas wspólnie spędzony przez jednostki współbieżne nad rozwiązywaniem problemu ponad czas niezbędny do rozwiązania tego samego problemu przez najlepszy algorytm sekwencyjny przy użyciu jednej jednostki przetwarzającej tego samego typu – $T_o = pT_p - T_s$
- Przyspieszenie – miara zysku wynikającego ze zrównoleglenia przetwarzania (realizowanego na p identycznych jednostkach przetwarzających) w stosunku do sekwencyjnego przetwarzania zrealizowanego przy użyciu najlepszego algorytmu sekwencyjnego – $S=T_s/T_p$ – jednostki przetwarzające systemu równoległego są identyczne do wykorzystywanych w przetwarzaniu sekwencyjnym.

Przyspieszenie

- Wartość przyspieszenia zazwyczaj nie przekracza liczby jednostek przetwarzających.
  - Jeżeli czas przetwarzania współbieżnego na *p* procesorach wynikający z równego podziału przetwarzania sekwencyjnego trwającego T_s wynosi mniej niż T_s/p (wtedy byłoby $S>p$) to emulując sekwencyjnie pracę *p* procesorów na jednej jednostce przetwarzającej uzyskalibyśmy czas emulacji < T_s co jest sprzeczne z założeniem, że najszybsze przetwarzanie sekwencyjne trwa T_s.
- W praktyce przyspieszenie większe od liczby procesorów występuje – wynika ono z:
  - struktury i sposobu wykorzystania zasobów systemu przetwarzającego - w systemie z identycznymi p procesorami więcej pamięci podręcznej – wielkość pamięci przypadająca na wielkość wykorzystywanych danych decyduje o efektywności jej wykorzystania. Przykład:
    - w systemie 2 procesorowym ilość danych wykorzystywanych przez procesor spadnie o połowę w stosunku do przetwarzania sekwencyjnego
    - efekt: wzrost stosunku trafień do pamięci podręcznej gdyż więcej stosunkowo danych w tej pamięci
    - efekt: obliczenia ograniczone prędkością dostępu do pamięci będą realizowane szybciej ze względu na szybszy średni dostęp do pamięci (zależne także od wielkości danych i prędkości dostępu do danych nielokalnych)
  - innej kolejności przetwarzania w przypadku problemów przeszukiwania – wielkość uzyskiwanego przyspieszenia wynika z lokalizacji poszukiwanego rozwiązania, sposobu podziału problemu i przydziału zadań do procesorów.

Efektywność i koszt

- Efektywność określa tę część czasu przetwarzania w jakiej procesory są efektywnie wykorzystane $E=S/p$
- W idealnym systemie równoległym $E=1$
- Koszt (praca) określa ilość czasu wykorzystywania do rozwiązywania problemu procesorów systemu równoległego $C=T_p*p$.
- System współbieżny jest *kosztowo optymalny* jeżeli koszt w funkcji rozmiaru problemu rośnie (asymptotycznie) tak samo szybko (Θ) jak funkcja złożoności najlepszego algorytmu sekwencyjnego.
- $rT_s(n) <= C(p,n) <= t\,T_s(n)$ dla stałych r,t : 1<=r<=t
- Zależność oznaczamy $C= \Theta(T_s(n))$

Wpływ granulacji obliczeń na efektywność

Skalowanie w dół systemu równoległego oznacza używanie mniejszej niż wymagana liczby procesorów (symulują one pracę większej liczby jednostek przetwarzających).

W systemach optymalnych kosztowo skalowanie w dół nie powinno powodować spadku efektywności (powoduje wzrost ziarna przetwarzania). Systemy nieoptymalne kosztowo mogą takimi pozostać. Może też powstać system optymalny kosztowo dzięki skalowaniu w dół – np. zaleznie od zastosowanego algorytmu.

Skalowanie w dół systemu równoległego (metoda 1)

sumowanie 16 liczb na 4 węzłach

zamiast 16 procesorów – 4

Metoda: prześlij i dodaj (nieefektywna)

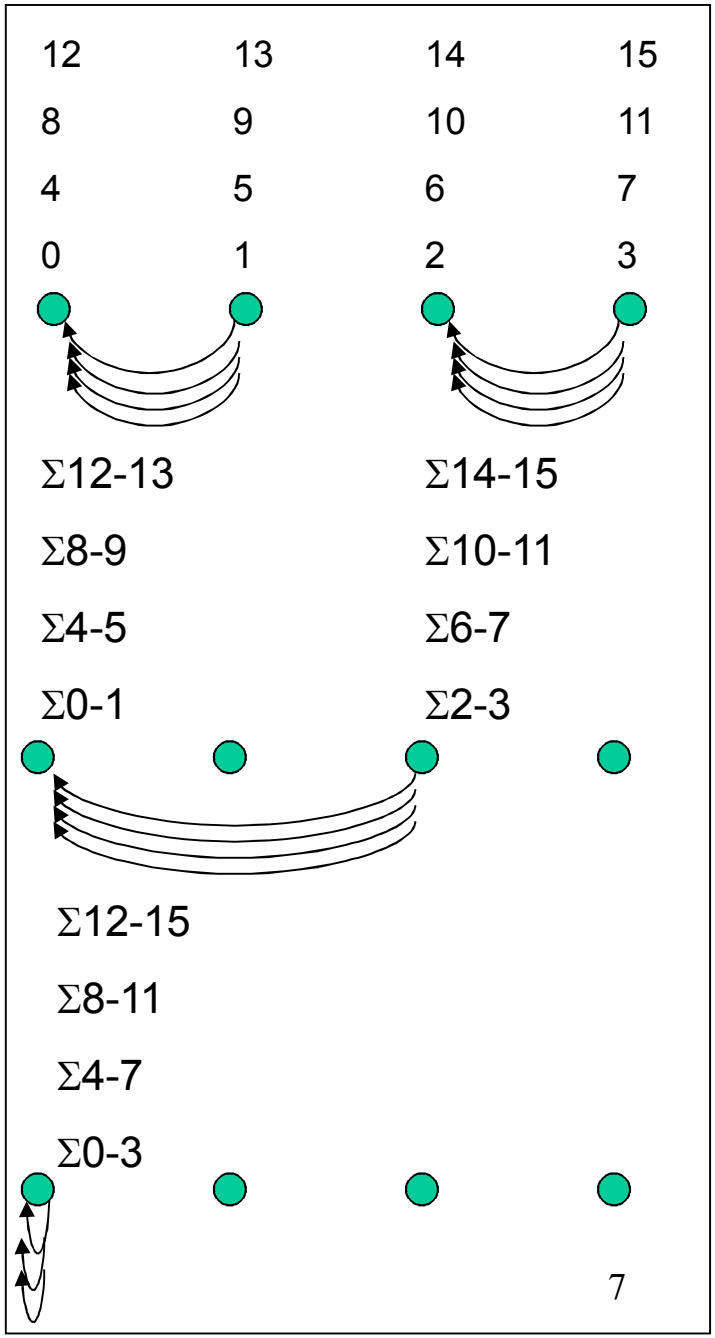
$T_{p2} = \Theta(n/p \log p + n/p)$

- *Komunikacja + sumowanie wykonywane n/p log p razy – n/p przesłań i sumowań realizowane log p razy.*

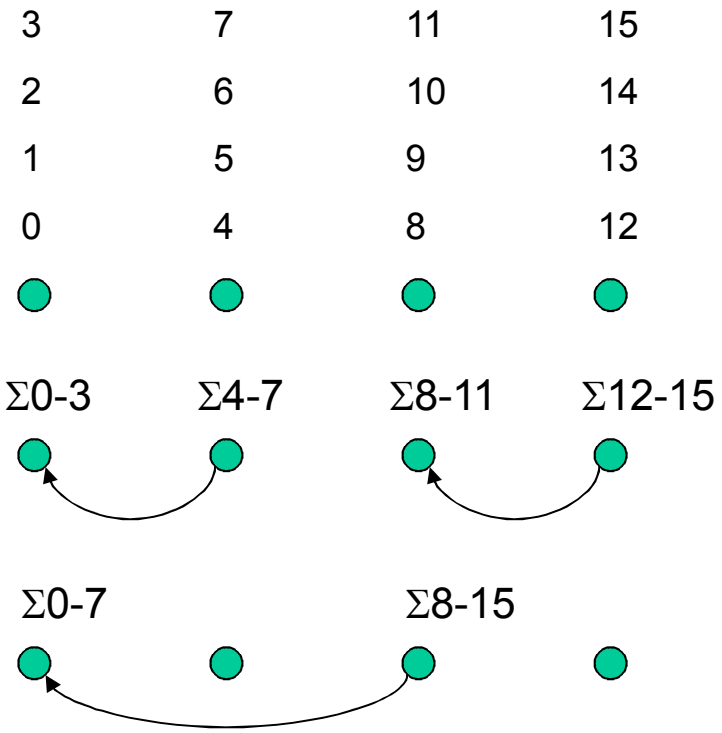
- *Sumowanie w docelowym węźle to n/p*

$C_2 = T_{p2} * p = \Theta(n \log p)$

$C > \Theta(n)$ - system jest nieoptymalny kosztowo



Skalowanie w dół systemu równoległego (metoda 2)



Takie skalowanie systemu równoległego pozwala uzyskać system optymalny kosztowo.

1. $T_p = \Theta(n/p + \log p);\ C = \Theta(n + p \log p)$
2. jeżeli n „asymptotycznie rośnie tak samo szybko jak” plog p

to $C = \Theta(n)$, a system jest optymalny kosztowo

Skalowalność systemów współbieżnych

System skalowalny umożliwia zachowanie stałej wielkości efektywności przy wzroście liczby procesorów dzięki wzrostowi rozmiaru przetwarzanego problemu.

$$E = T_s/(pT_p)$$

$$T_o = pT_p - T_s$$

$$E = 1/(1+T_o/T_s)$$

koszt zrównoleglenia $T_o = f(p)$ $np.T_o > t_{serial}(p-1)$, gdyż część sekwencyjna realizowana tylko przez jeden procesor, p-1 pozostałych czeka;

Skalowalność systemów współbieżnych

Poprzedni przykład „system optymalny kosztowo dodawania liczb” (zakładamy czas komunikacji =czas sumowania - stąd 2 we wzorze):

$T_p = n/p + 2\log p$ ***- n/p sumowań lokalnych oraz log p sumowań i przesłań***

$T_1 = n$

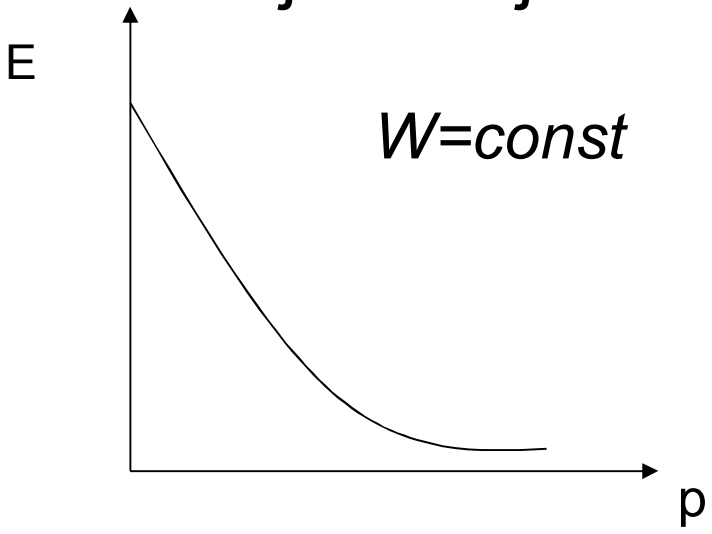
$E = T_1 / (pT_p) = n/(n+2p\log p) = 1/(1+2p/n * \log p)$

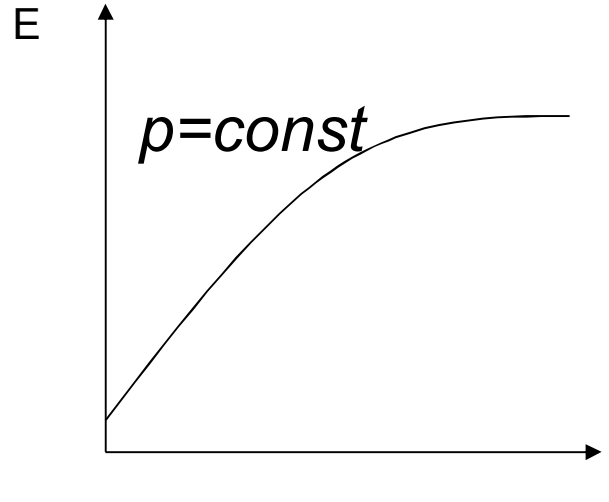
Przykład: jeśli E=0,8 to wtedy 2p/n * logp = ¼;

zatem gdy p=4 to n=64, a gdy zwiększymy liczbę procesorów do 8 to n powinno wzrosnąć do 192

dla uzyskania stałej wielkości efektywności.

Funkcja stałej efektywności - miara skalowalności





Typowe przebiegi

E=f(p),
E=f(W)

$T_p=(W+T_o(W,p))/p$

zatem jeśli $E = W/(pT_p)$ *to* $E= 1/(1+T_o(W,p)/W)$ *a stąd*

W= E/(1-E) $T_o(W,p)$ czyli W= K $T_o(W,p)$

Funkcja stałej efektywności W=f(p) określa:

- **jak łatwo** system równoległy zachowa stałą efektywność i osiągnie przyspieszenie odpowiednie do wzrostu liczby procesorów.
- jak powinna rosnąć ilość pracy do wykonania w systemie równoległym, aby zapewnić, że przy wzroście liczby procesorów stałą zachowana zostanie wielkość przyspieszenia.

Prawo Amdahla (1)

Założenia:

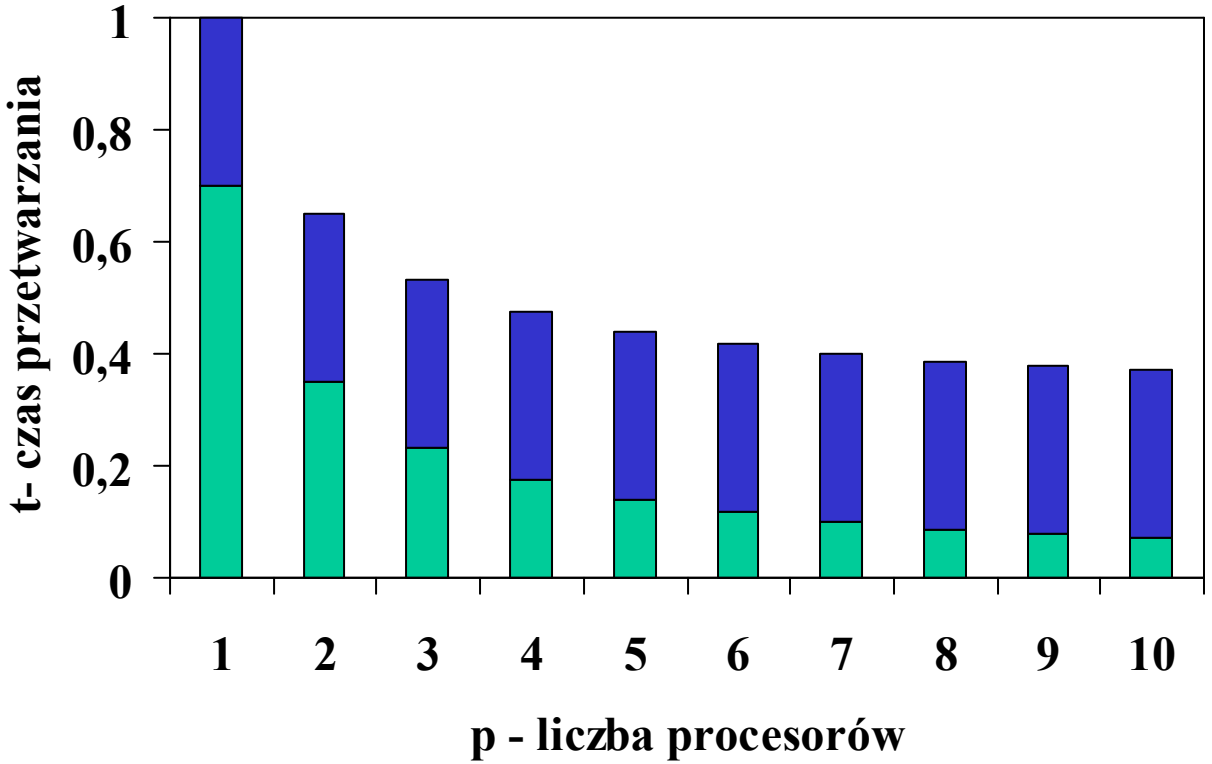
czas wykonania programu dla rozwiązania instancji problemu w sposób sekwencyjny ma dwa składniki:

- *s* – możliwy do wykonania w sposób tylko sekwencyjny i
- *r* – możliwy do wykonania w sposób równoległy. Daje się on zrównoleglić w dowolny sposób - czas przetwarzania maleje odwrotnie proporcjonalnie do użytej liczby procesorów.
- $s+r=1$

Rozpatrujemy przetwarzanie instancji problemu o stałym rozmiarze.

Prawo Amdahla (2)

czas przetwarzania w funkcji liczby procesorów s=30%



Prawo Amdahla (3)

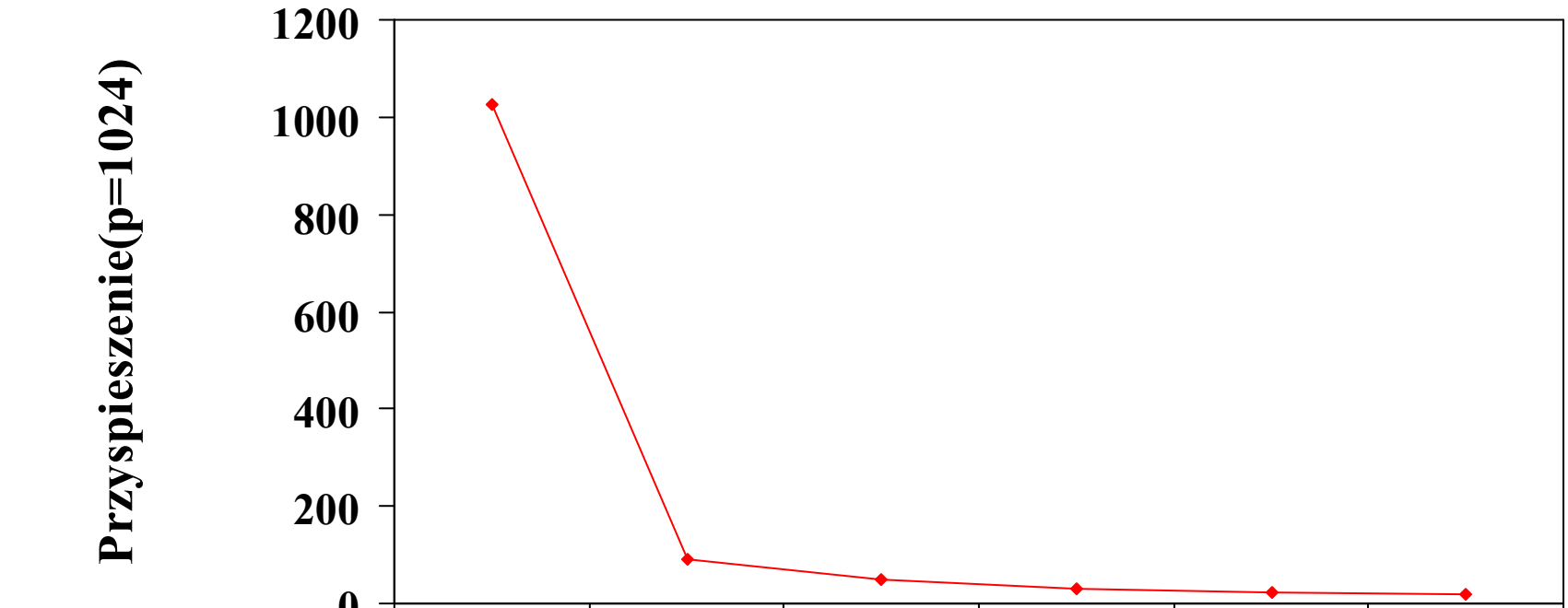
$$S(p) = \frac{s + r}{s + r/p}$$

$$\lim_{p \to \infty} S(p) = 1/s$$

$$\lim_{r \to 1} S(p) = p$$

Prawo Amdahla (4)

przyspieszenie w funkcji wielkości części sekwencyjnej
s =<0%,5%>

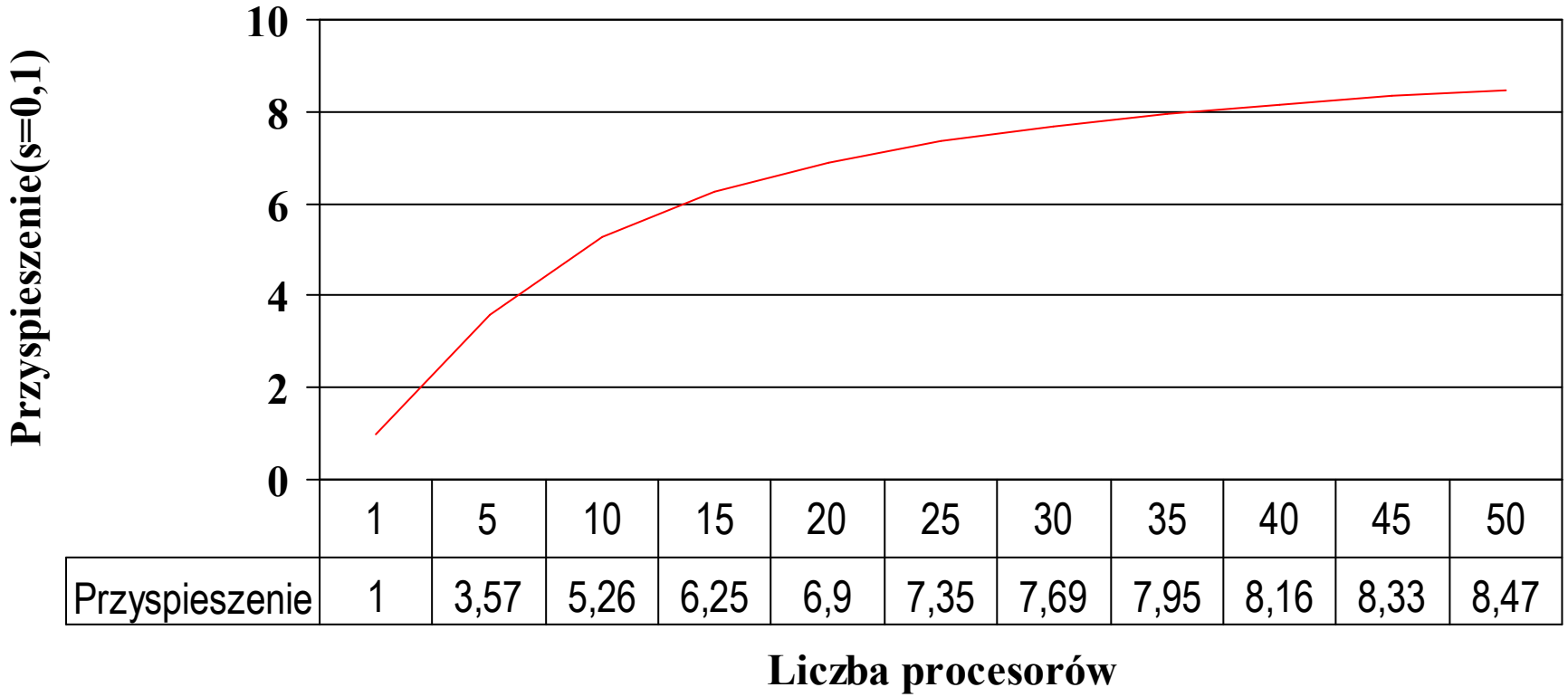


	0	0,01	0,02	0,03	0,04	0,05
PRZYSPIESZENIE	1024	91	48	32	24	20

s

Prawo Amdahla (5)

przyspieszenie w funkcji liczby procesorów (s=10%)



	1	5	10	15	20	25	30	35	40	45	50
Przyspieszenie	1	3,57	5,26	6,25	6,9	7,35	7,69	7,95	8,16	8,33	8,47

Prawo Amdahla (6)

Wnioski:

- Ograniczenie stosowalności przetwarzania równoległego ze względu na ograniczenie wielkości możliwego do osiągnięcia przyspieszenia.
- Silny spadek przyspieszenia z niewielkim wzrostem wielkości nie poddającej się zrównolegleniu części przetwarzania.

Prawo Gustafsona (1)

Założenia:

Badamy przetwarzanie dla szeregu instancji problemu dla których czas wykonania programu w sposób sekwencyjny ma dwa składniki:

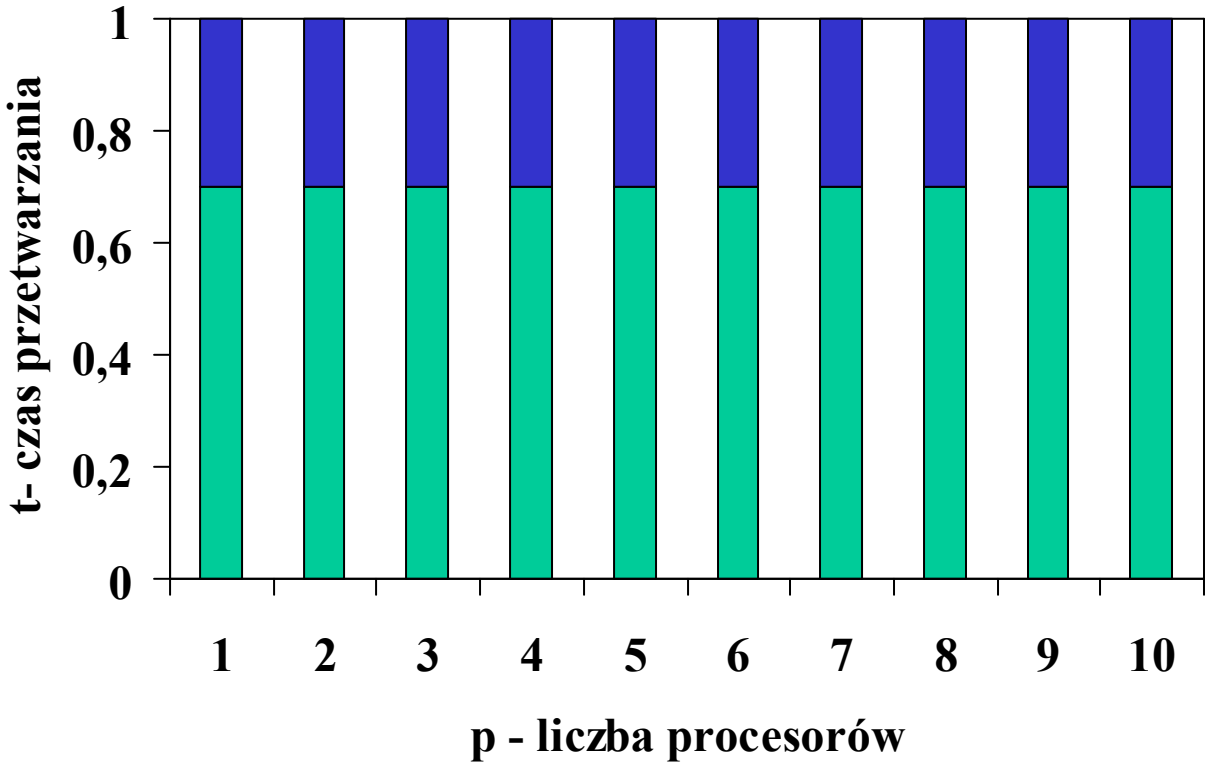
- ***s*** – możliwy do wykonania jedynie w sekwencyjnie i
- ***r*p*** (dla m = 1,2,3,...) – możliwy do wykonania w sposób równoległy.
- $s+r=1$

Instancje o sekwencyjnym czasie przetwarzania równym $s+r*p$ są przetwarzane przy użyciu p procesorów (dla p= 1,2,3,...).

Rozpatrujemy przetwarzanie szeregu instancji problemu o rosnącym rozmiarze i stałym czasie przetwarzania równoległego w funkcji użytych procesorów.

Prawo Gustafsona (3)

czas przetwarzania badanego szeregu instncji w funkcji liczby procesorów s=0,3



Prawo Gustafsona (2)

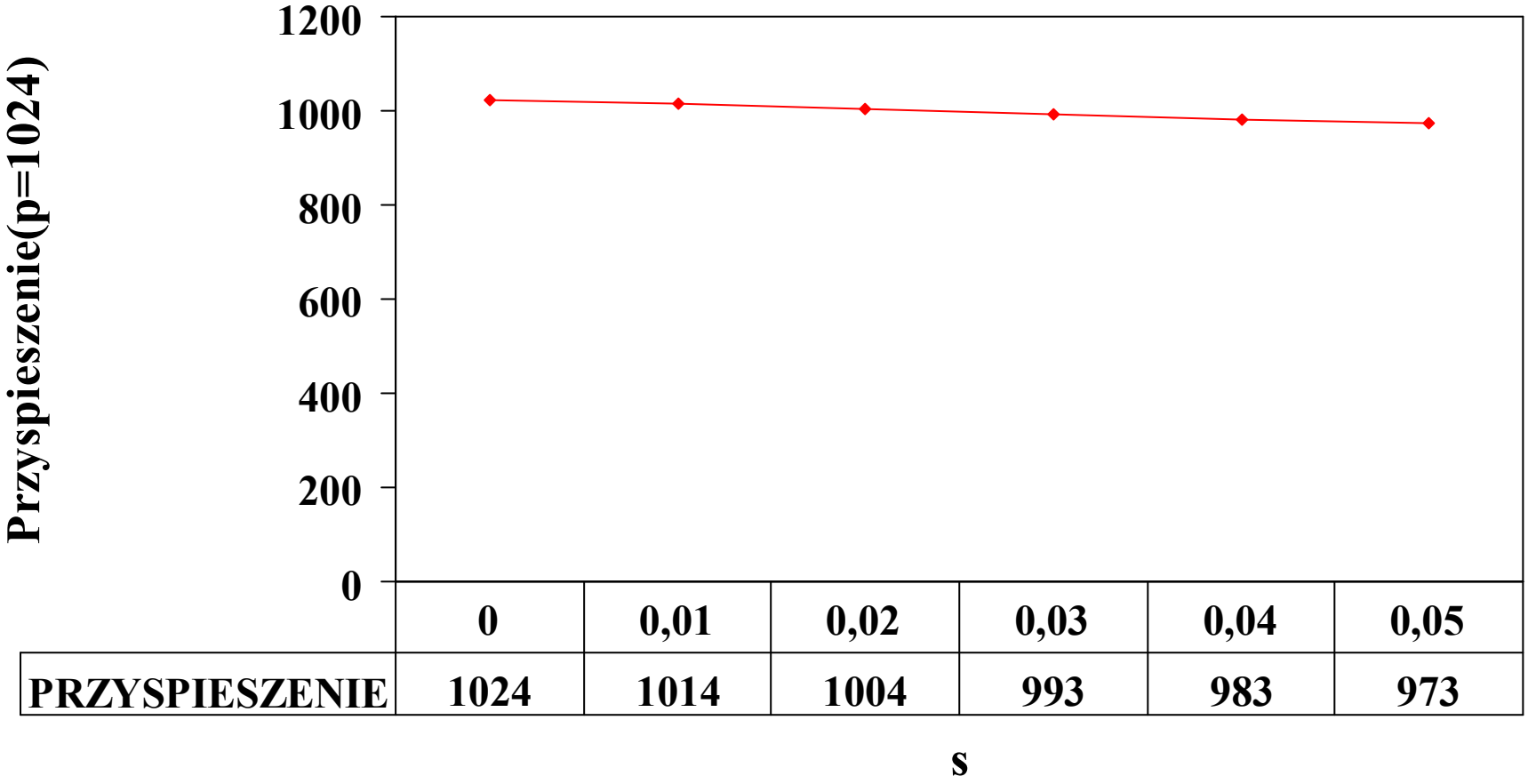
- Wyznaczamy przyspieszenie dla powyższego szeregu instancji przetwarzanych na odpowiedniej liczbie procesorów

$$S(p) = \frac{s + pr}{s + r} = s + pr = p - s(p - 1)$$

$$\lim_{p \to \infty} S(p) = \infty$$

Prawo Gustafsona (4)

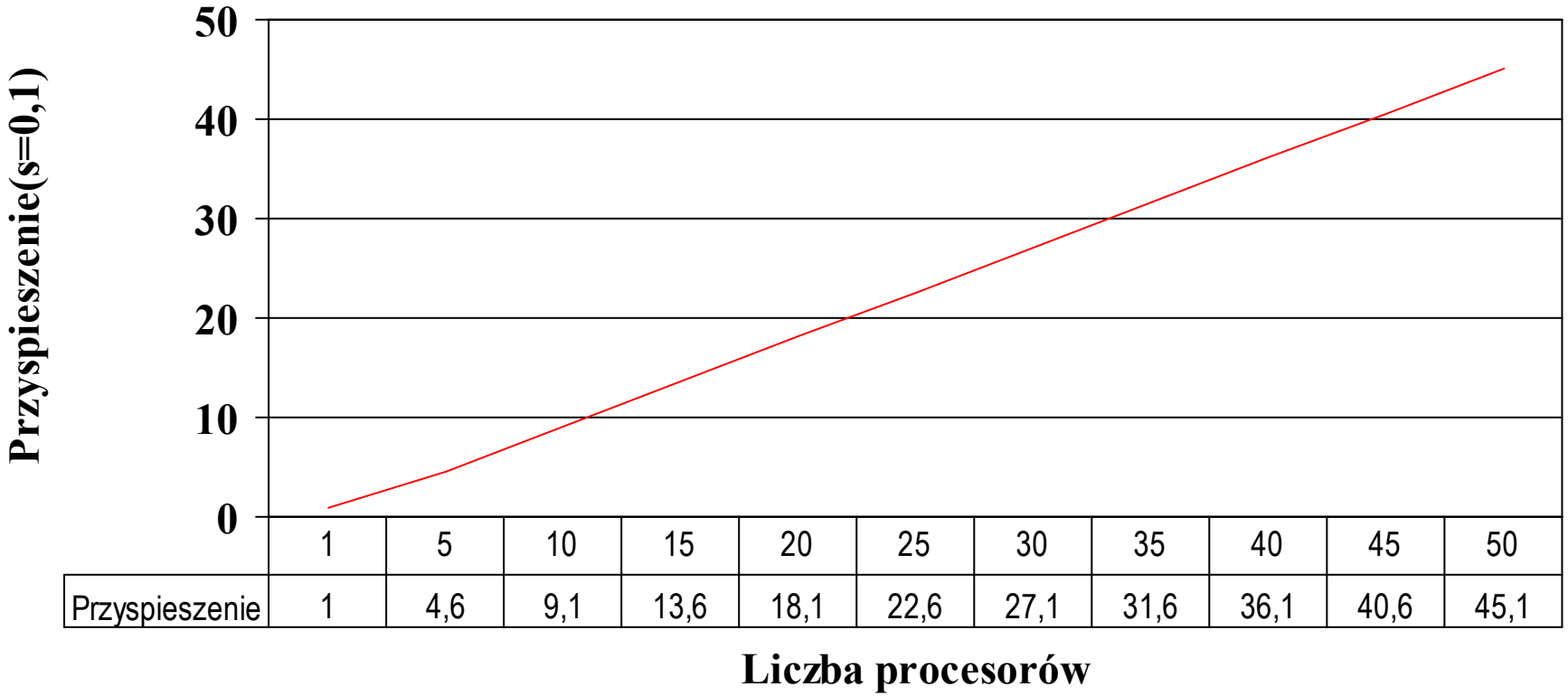
przyspieszenie w funkcji wielkości części sekwencyjnej
s =<0,0.05>



	0	0,01	0,02	0,03	0,04	0,05
PRZYSPIESZENIE	1024	1014	1004	993	983	973

Prawo Gustafsona (5)

przyspieszenie w funkcji liczby procesorów s=0,1



	1	5	10	15	20	25	30	35	40	45	50
Przyspieszenie	1	4,6	9,1	13,6	18,1	22,6	27,1	31,6	36,1	40,6	45,1

Prawo Gustafsona (6)

Wnioski:

Przy wzroście złożoności przetwarzania ze wzrostem liczby procesorów możliwe jest efektywne przetwarzanie równoległe:

- Nie ma ograniczenia na wielkość możliwego do osiągnięcia przyspieszenia.
- Liniowy wzrost przyspieszenia w funkcji liczby procesorów.
- Mniejszy wpływ wielkości **części** niepodlegającej zrównolegleniu na przyspieszenie. Założono niezależność wielkości tej części od wielkości instancji problemu. W wielu przypadkach założenie to nie jest jednak prawdziwe – np. wielkość części sekwencyjnej zależy od czasu trwania operacji wejścia-wyjścia, który to zazwyczaj rośnie z rozmiarem problemu.