Teoria Wielowątkowości

CODERS SCHOOL

https://coders.school

Łukasz Ziobroń
lukasz@coders.school

Łukasz Ziobroń & Bartosz Szurgot - autorzy

Łukasz Ziobroń

Not only programming experience:

- C++ and Python developer @ Nokia & Credit Suisse
- Scrum Master @ Nokia & Credit Suisse
- Code Reviewer @ Nokia
- Webmaster (HTML, PHP, CSS) @ StarCraft Area

Training experience:

- C++ trainings @ Coders School
- Practial Aspects Of Software Engineering @ PWr, UWr
- Nokia Academy @ Nokia
- Internal corporate trainings

Public speaking experience:

- Academic Championships in Team Programming
- code::dive conference
- code::dive community

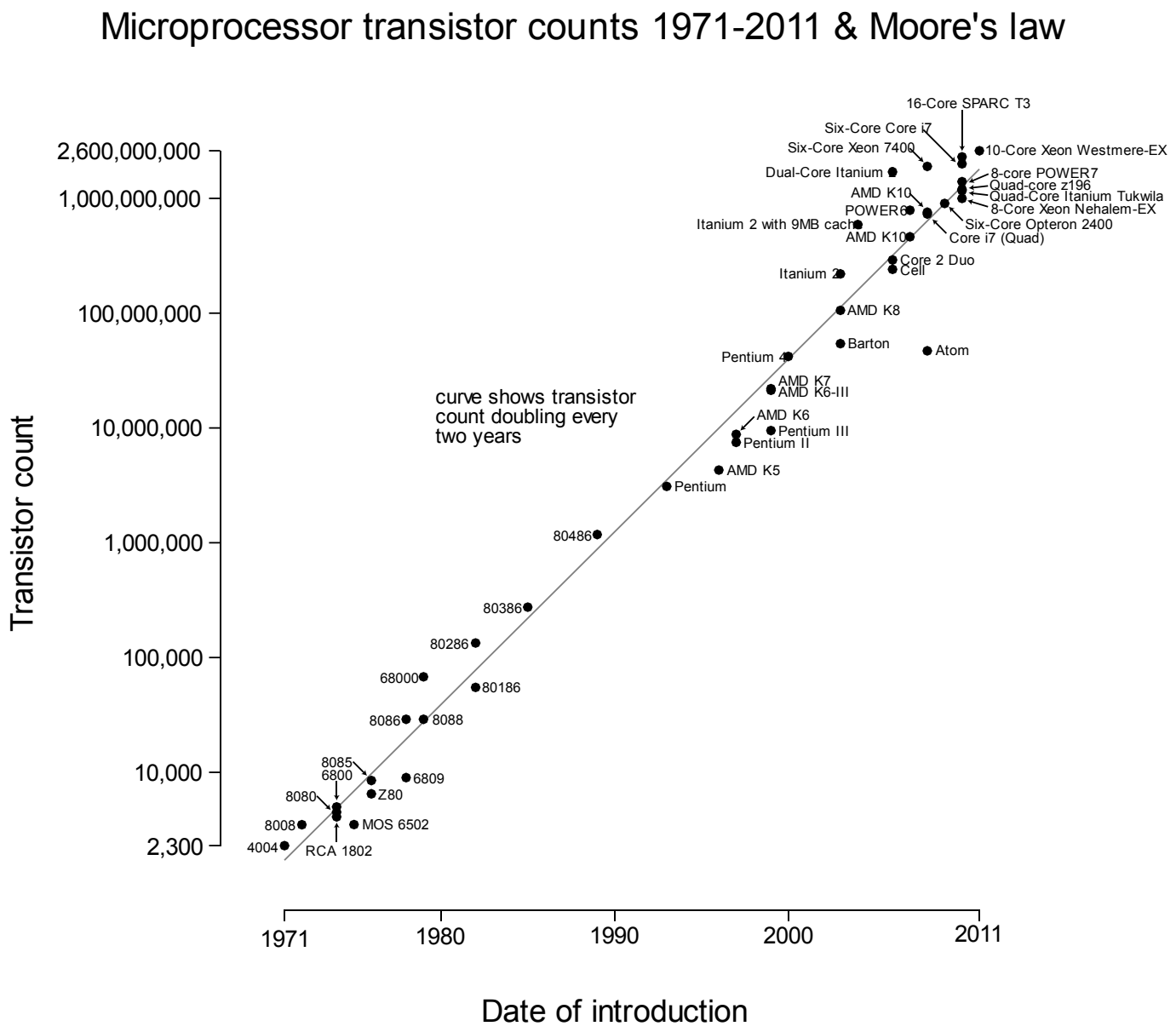
Agenda

Teoria wielowątkowości

- Prawo Moore'a
- Free lunch is over
- Program sekwencyjny vs program równoległy
- Prawo Amdhala
- Wielowątkowe Hello World
- Słownik pojęć
  - Wątek
  - Proces
  - Współbieżność
  - Równoległość
- Zalety współbieżności
- Wady współbieżności

Prawo Moore'a

Liczba tranzystorów w procesorze podwaja się co około 2 lata



Free lunch is over

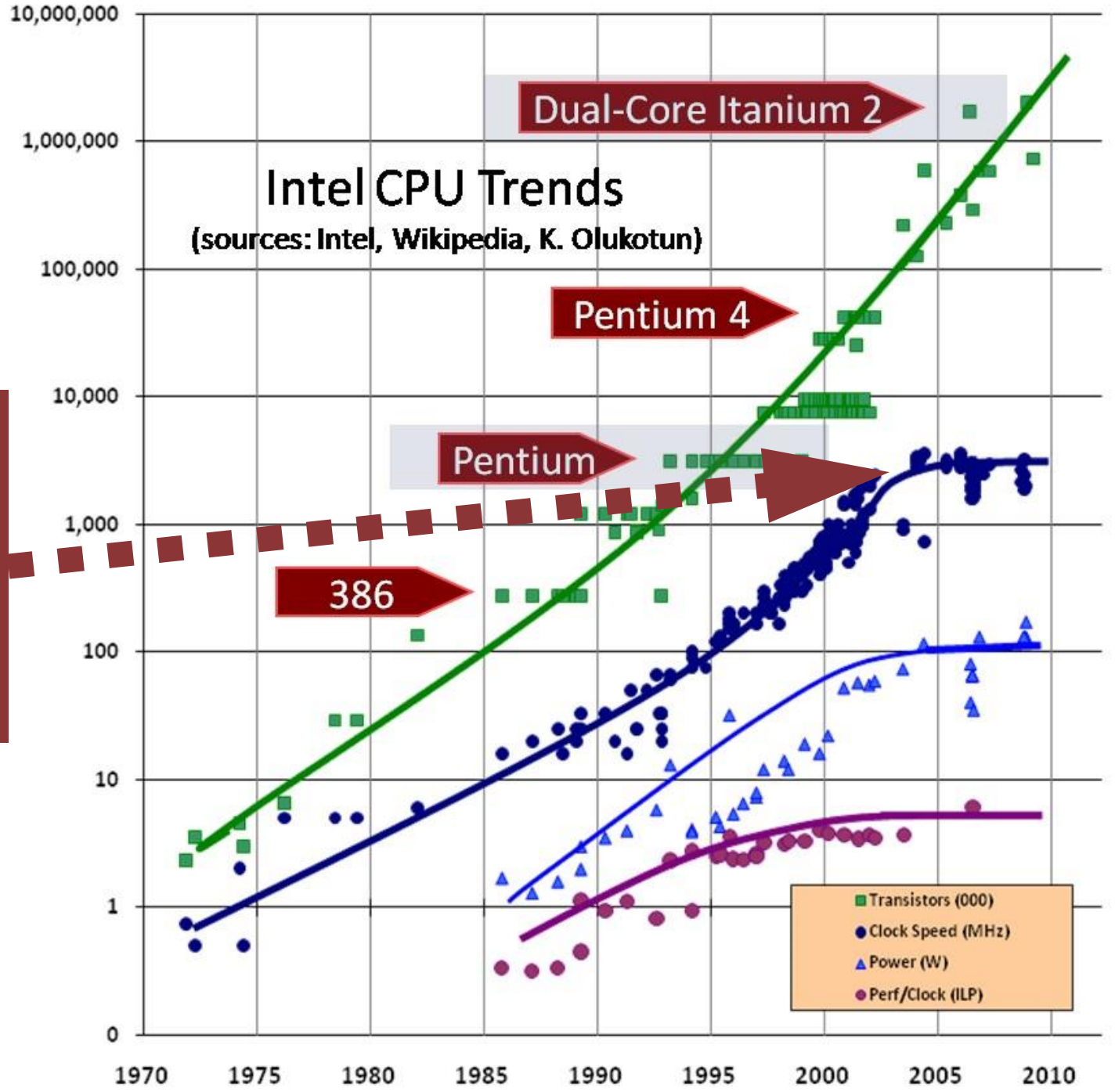
Programiści nie musieli się martwić o wydajność swoich programów. Jeśli program działał wolno, to mogli poczekać, aż pojawi się na rynku szybszy sprzęt. Mogli korzystać z „darmowego lunchu" ulepszając sprzęt.

Free lunch is over – prezentacja
Free lunch is over – artykuł Herba Suttera

Prawo Moore'a w praktyce

Free lunch is over

Częstotliwość taktowania zegarów procesora przestała rosnąć ok. 2004 roku



Program sekwencyjny

```cpp
#include <iostream>
using namespace std;

int veryComplexStuff(int) { /* ... */ }

int main() {
    auto out = 0;
    for(auto i = 0; i < 8'000'000; ++i)
        out += veryComplexStuff(i);
    cout << "answer: " << out << endl;
    return 0;
}
```

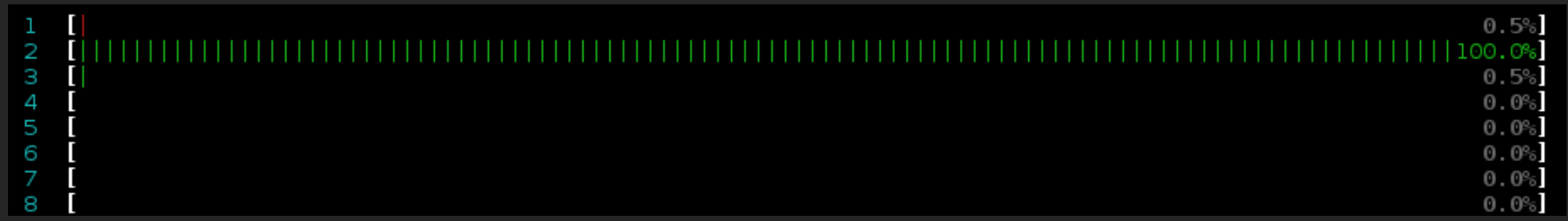
Wnioski

- Czas wykonania zależy od sprzętu
  - rok 2014: 100s
  - rok 2019: 70s
- Przybyło rdzeni
- 7/8 rdzeni (87,5%) nie robi nic
- Duże rezerwy mocy do wykorzystania

Podgląd wykonania (htop)

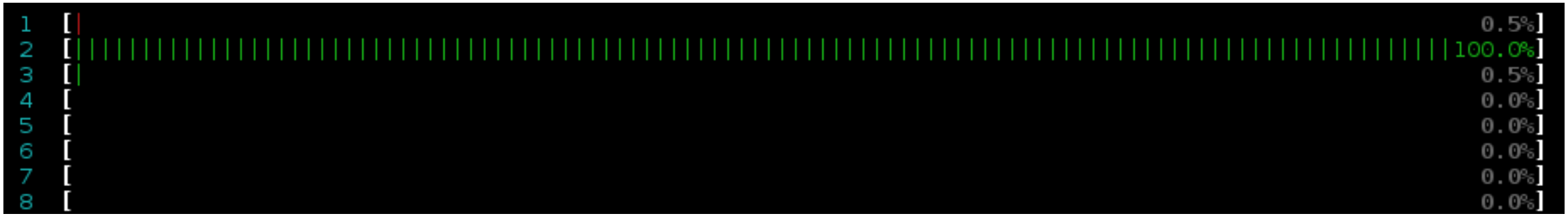
Rok 2004:

Rok 2019:

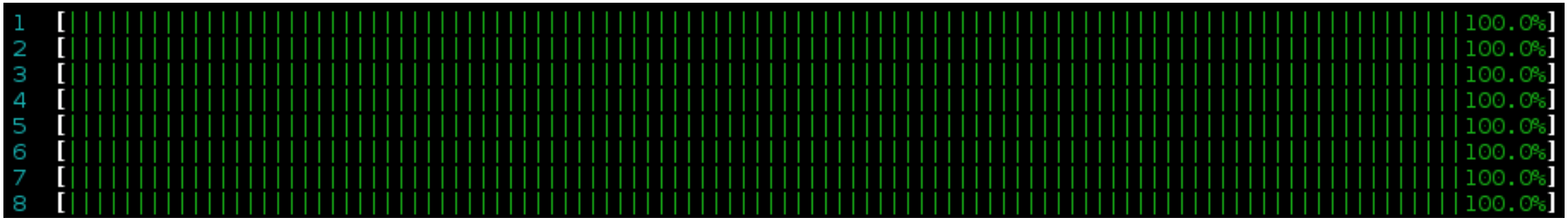


Program sekwencyjny - problem

Obciążenie procesora:



Oczekiwane obciążenie:



Jak tego dokonać?

Program równoległy – krok 1: wyciągnięcie algorytmu

```
#include <iostream>
using namespace std;

int veryComplexStuff(int) { /* ... */ }

int main() {
    auto out = 0;
    for(auto i = 0; i < 8'000'000; ++i)
        out += veryComplexStuff(i);
    cout << "answer: " << out << endl;
    return 0;
}
```

```
int veryComplexStuff(int) { /* ... */ }

auto computePart(int begin, int end) {
    auto out = 0;
    for(auto i = begin; i < end; ++i)
        out += veryComplexStuff(i);
    return out;
}
```

Program równoległy – krok 2: podział pracy

```
int veryComplexStuff(int) { /* ... */ }

auto computePart(int begin, int end) {
    auto out = 0;
    for(auto i = begin; i < end; ++i)
        out += veryComplexStuff(i);
    return out;
}

int main() {
    int part[8]; // partial results
    part[0] = computePart(0, 999'999);            // thread #1
    part[1] = computePart(1'000'000, 1'999'999); // thread #2
    // etc...
    part[7] = computePart(7'000'000, 7'999'999); // thread #8
    auto out = accumulate(begin(part), end(part), 0u);
    cout << "answer: " << out << endl;
}
```

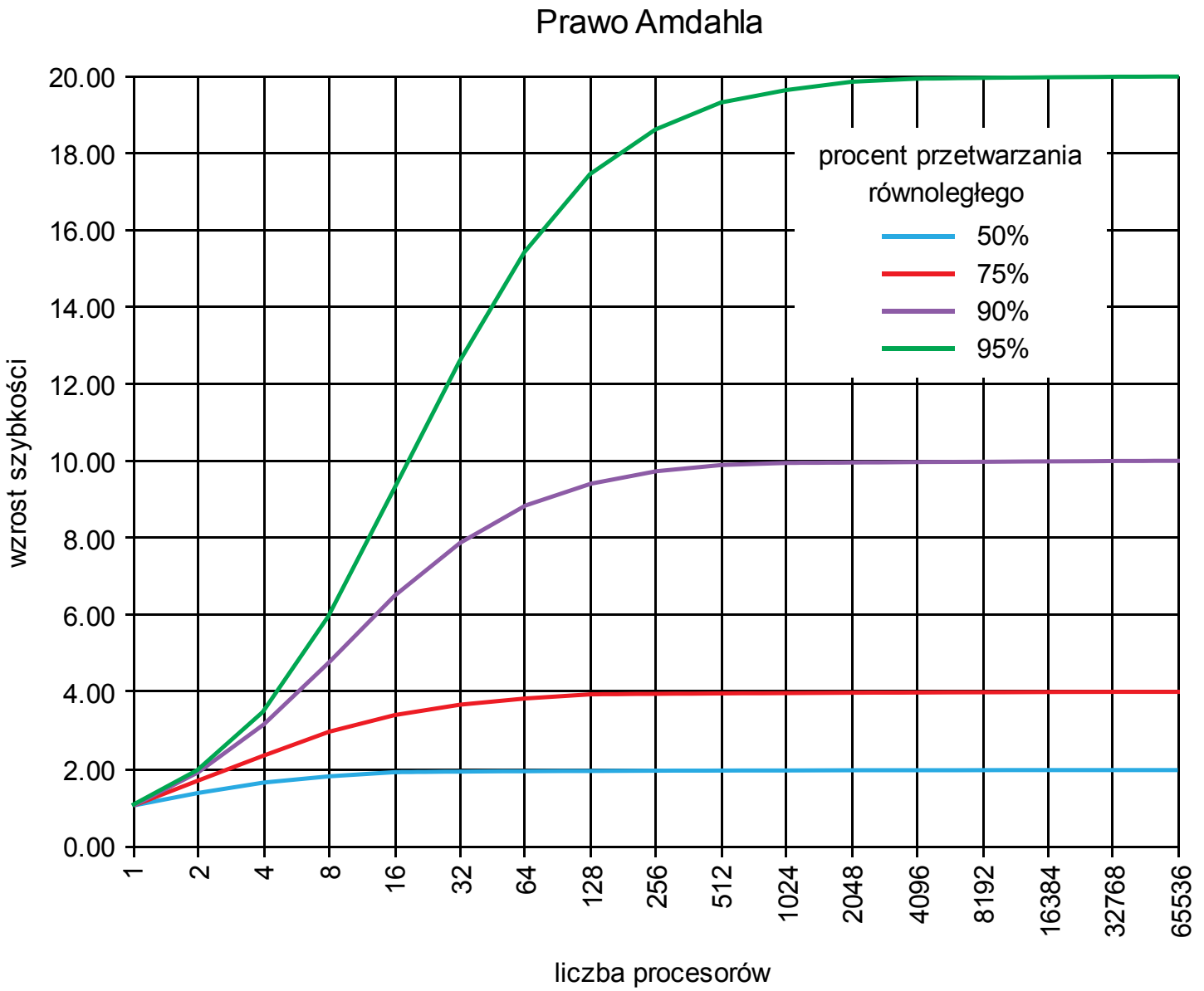
Prawo Amdhala

Zwiększenie szybkości wykonywania się programu przy użyciu wielu procesorów w obliczeniach równoległych jest ograniczane przez czas potrzebny do sekwencyjnego dzielenia programu.

Na przykład, jeżeli 95% programu może być przetworzone równolegle, wówczas maksymalny wzrost szybkości wykonania programu przy użyciu przetwarzania równoległego może wynieść 20x bez względu na ilość użytych procesorów.

$$\frac{1}{(1-P)+\frac{P}{N}}$$

1-P = część, której nie można zrównoleglić
P – część, którą można zrównoleglić
N – liczba procesorów



Wielowątkowe Hello World

Zadanie 1
Napisz program, który w oddzielnym wątku wyświetli "Hello World" na ekranie.

Zadanie 2
Skompiluj go i uruchom :)

```cpp
#include <iostream>    // https://ideone.com/4dp7PL
#include <thread>
using namespace std;

int main()
{
    thread t([]{ cout << "Hello World" << endl; });
    t.join();
    return 0;
}
```

```
$> g++ hello.cpp
/tmp/ccuz9XKm.o: In function `std::thread::thread<main::{lambda()#1}>(main::{lambda()#1}&&)':
hw.cpp:(.text+0x135): undefined reference to `pthread_create'
collect2: error: ld returned 1 exit status

$> g++ hello.cpp -lpthread
$> ./a.out
Hello World
```

Słownik pojęć

- **Wątek (thread)** - Część programu wykonywana współbieżnie w obrębie jednego procesu. Zwany też "lekkim procesem".
- **Proces (proces)** - Egzemplarz wykonywanego programu. Proces może mieć wiele wątków.

	Wątki	Procesy
Stosy	Osobne	Osobne
Sterty	Wspólne	Osobne
Komunikacja	Szybka poprzez wspólną przestrzeń adresową (stertę) - wskaźniki, struktury danych	Wolniejsza poprzez systemowe mechanizmy IPC (gniazda, pliki, sygnały, potoki, pamięć współdzielona)

Słownik pojęć

- **Współbieżność (concurrency)** - Rodzaj przetwarzania, w którym kilka obliczeń może być realizowanych niezależnie od siebie, w nachodzących okresach czasowych (nawet na jednym procesorze). Może występować konkurowanie o wspólny zasób (np. czas procesora)
- **Równoległość (parallelism)** - Rodzaj przetwarzania, w którym kilka obliczeń jest realizowanych dokładnie w tym samym czasie (np. na wielu procesorach).

Równoległość... a współbieżność

Zalety współbieżności

- Rozdzielenie odpowiedzialności
  - Grupowanie powiązanego kodu
  - Rozdzielanie odrębnych operacji, gdy mają się odbywać w tym samym czasie
- Wydajność
  - Wykorzystanie całej mocy maszyny wieloprocesorowej
  - Skalowalność aplikacji
  - *Free lunch is over*
- Lepsza responsywność aplikacji
  - Unikanie blokujących operacji I/O (dysk, karta sieciowa)
  - Brak blokowania interfejsu użytkownika

Wady współbieżności

- Kod wielowątkowy jest dużo bardziej skomplikowany - trudniejszy do pisania, czytania i testowania niż kod jednowątkowy
- Większa złożoność, więcej błędów, które są trudne do wykrycia (często trudne do odtworzenia)
- Trudna testowalność ze względu na losową powtarzalność problemów w zależności od obciążenia maszyny
- Narzuty związane z zarządzaniem wątkami mogą drastycznie obniżyć wydajność (context switching, cache ping-pong, false sharing)
- Zbyt wiele wątków uruchomionych jednocześnie
  - Zużycie zasobów systemu operacyjnego
  - System jako całość będzie działał wolniej

Podstawowa znajomość współbieżności i wielowątkowości to **konieczność**

Zaawansowana znajomość współbieżności i wielowątkowości to **elitarna wiedza**

Przydatne linki

- Prawo Moore'a
- Prawo Amdhala
- Free lunch is over – prezentacja
- Free lunch is over – artykuł Herba Suttera

CODERS SCHOOL

https://coders.school

Łukasz Ziobroń
lukasz@coders.school