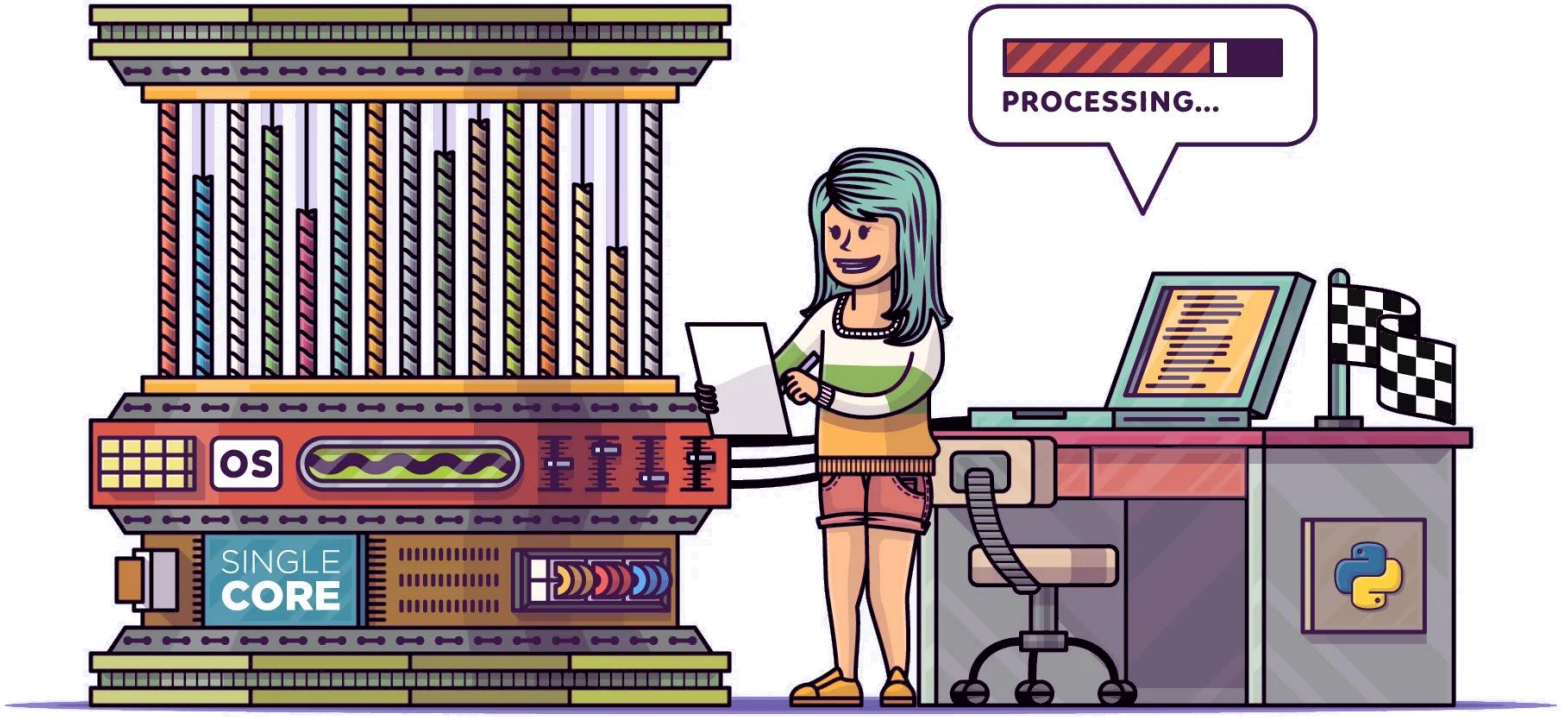
Programowanie wielowątkowe



Real Python

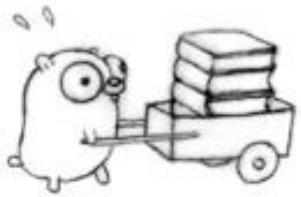
Przetwarzanie współbieżne

Przetwarzanie współbieżne - (*ang. concurrent computing*) ogólna koncepcja dekompozycji programu na takie części, które mogą być wykonywane niezależnie. Najczęstszym powodem takiej dekompozycji jest próba poprawienie wydajności programu. Jest pojęciem z zakresu architektury oprogramowania i projektowania systemów.

Zadanie

(przykład zaczerpnięty z wykładu Rob Pike)

Spal stos nieaktualnych manuali.

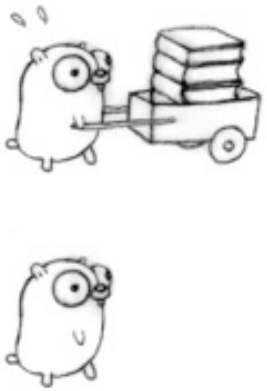


Może to zająć sporo czasu. Jak możemy usprawnić nasze rozwiązanie?

Zadanie

(przykład zaczerpnięty z wykładu Rob Pike)

Możemy dodać drugiego goffera.

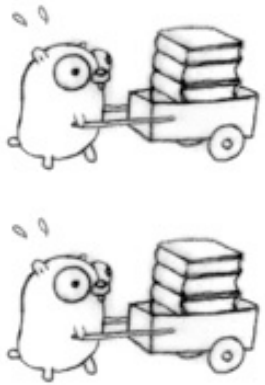


Ale to nie wystarczy, nowy goffer potrzebuje nowej taczki.

Zadanie

(przykład zaczerpnięty z wykładu Rob Pike)

Teraz powinno pójść szybciej.



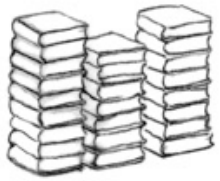
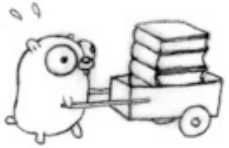
No ale mamy też wąskie gardła. Przy stosie i piecu (współdzielone zasoby).

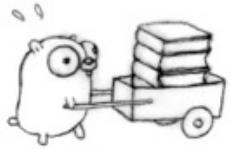
W jaki sposób możemy usunąć wąskie gardło?

Zadanie

(przykład zaczerpnięty z wykładu Rob Pike)

Całkowicie uniezależnić od siebie oba goffery.


Teraz, w najlepszym wypadku rozwiązanie będzie dwa razy szybsze (przy założeniu, że oba goffery pracują równocześnie (współbieżność nie zakłada równoczesności, ale ją dopuszcza). Czy istnieje jeszcze wydajniejsze rozwiązanie?

Zadanie

(przykład zaczerpnięty z wykładu Rob Pike)

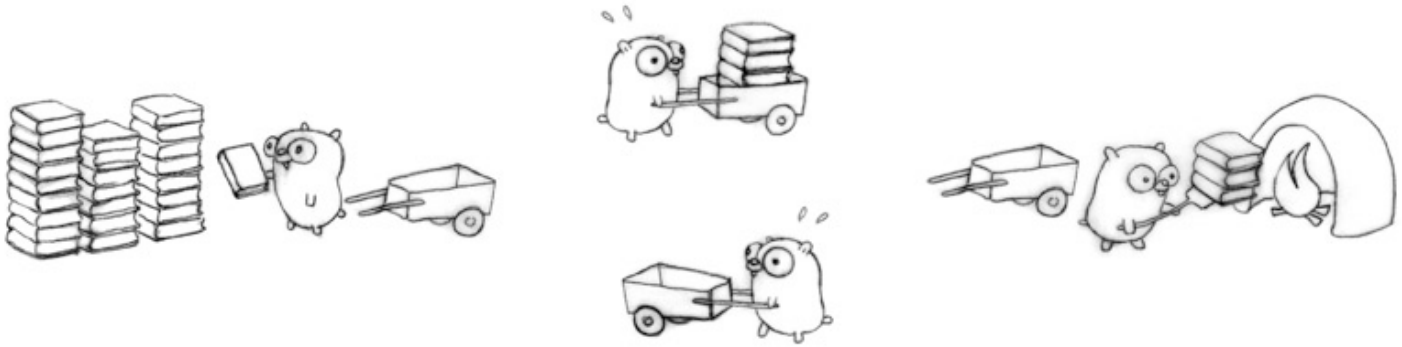
Wprowadźmy kolejnego goffera.

Przy założeniu, że goffery mogą wymieniać się taczkami, w najlepszym wypadku rozwiązanie będzie trzy razy szybsze (przy założeniu, że wszystkie trzy goffery pracują równocześnie). Czy możemy rozwiązanie zaprojektować jeszcze wydajniej?

Zadanie

(przykład zaczerpnięty z wykładu Rob Pike)

Cztery goffery.

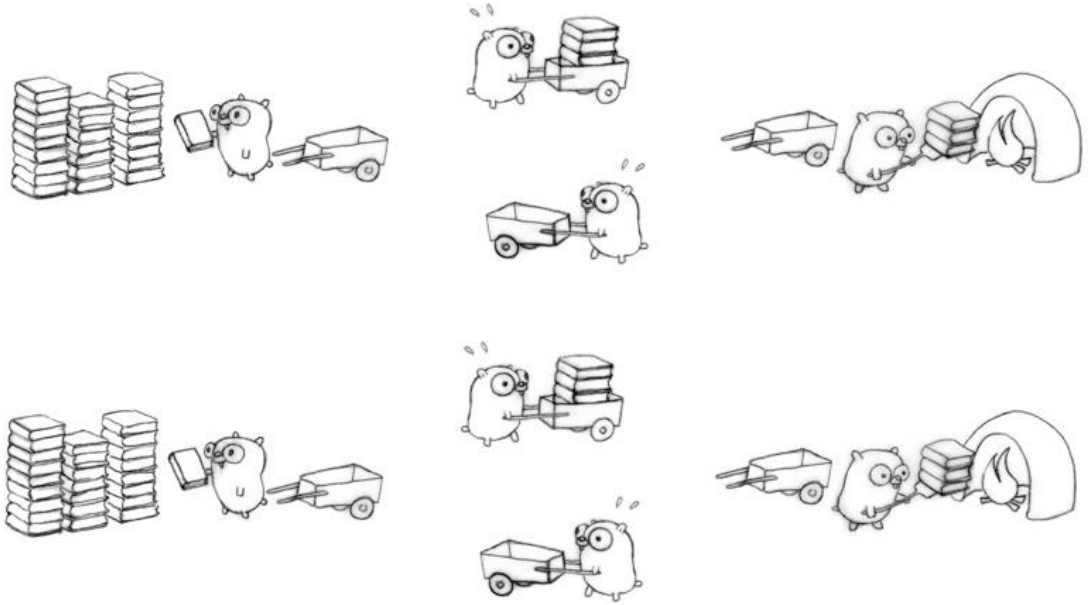


Takie rozwiązanie, kiedy dobrze **zrealizowane** może być czterokrotnie szybsze.

Zadanie

(przykład zaczerpnięty z wykładu Rob Pike)

A takie rozwiązanie



dobrze **zrealizowane** może być nawet ośmiokrotnie szybsze (kiedy wszystkie goffery pracują równocześnie). Czy istnieją inne rozwiązania?

Zadanie

(przykład zaczerpnięty z wykładu Rob Pike)

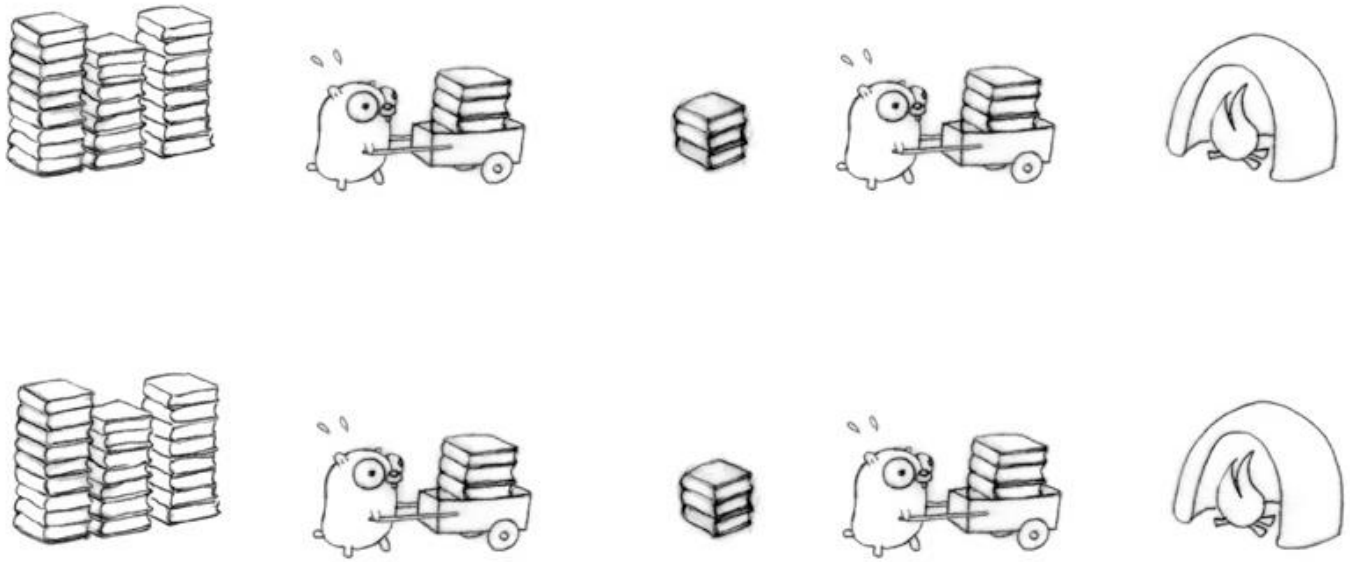
Oczywiście, całe mnóstwo! Na przykład:

takie rozwiązanie może nie będzie dwa razy szybsze, ale sam czas transportu książek może być nawet dwa razy szybszy (przy założeniu, że goffery pracują równocześnie).

Zadanie

(przykład zaczerpnięty z wykładu Rob Pike)

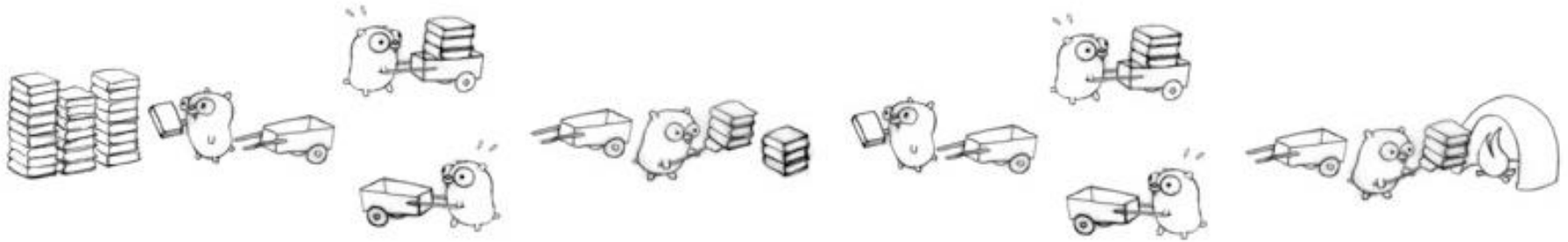
No i takie rozwiązanie możemy rozszerzyć wertykalnie



Zadanie

(przykład zaczerpnięty z wykładu Rob Pike)

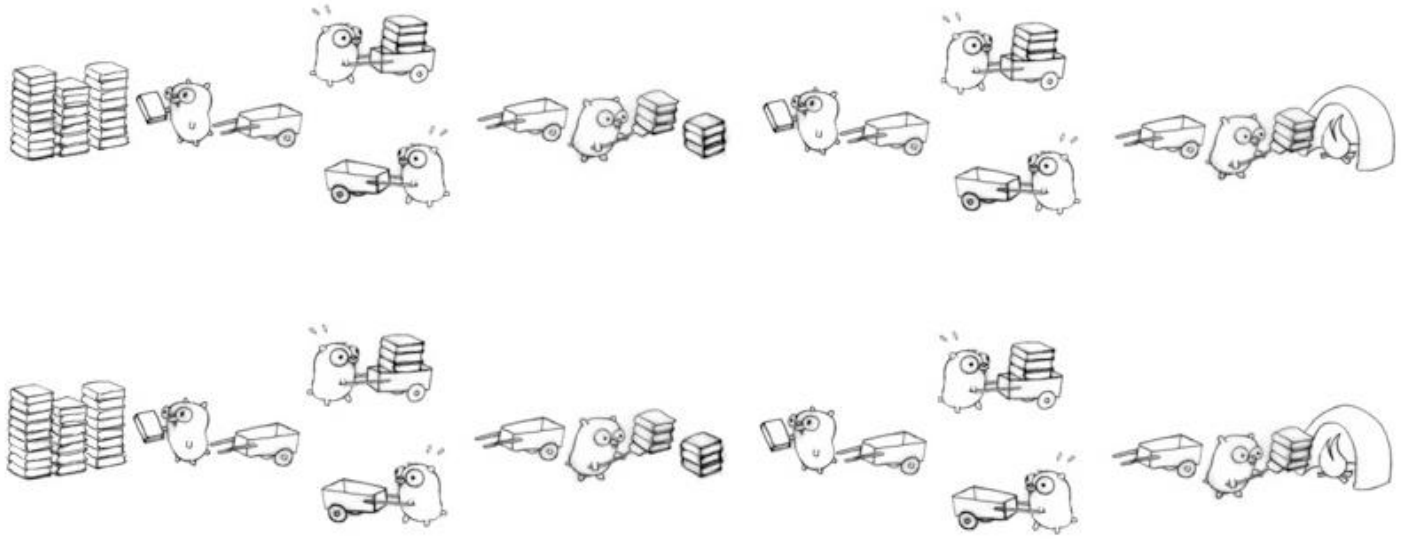
lub horyzontalnie



Zadanie

(przykład zaczerpnięty z wykładu Rob Pike)

a nawet tak



Otrzymując w ten sposób jeszcze wydajniejsze rozwiązanie.

Zadanie

(przykład zaczerpnięty z wykładu Rob Pike)

Nasze dotychczasowe rozważania stanowią przedmiot współbieżności, czyli koncepcji zgodnie z którą rozwiązanie możemy projektować poprzez dekompozycje. Rozbijamy rozwiązanie na mogące działać niezależnie części, umożliwiając w ten sposób uzyskanie lepszej wydajności. Czyli szukamy w naszym kodzie goferów (takich fragmentów, które możemy powielić i wykonywać równocześnie).

To na ile faktycznie zwiększymy wydajność rozwiązania zależy już od realizacji naszego projektu (pomysłu).

W ramach realizacji naszego projektu podejmujemy decyzję o tym w jaki sposób wykonywane będą poszczególne części.

Szersza perspektywa

Wyróżniamy trzy podstawowe techniki wykonywania wielu obliczeń:

- Wykonywanie sekwencyjne (*ang. sequential computing*) - obliczenia wykonywane są kolejno, jedno po drugim.
- Wykonywanie równoległe (*ang. concurrent computing*) - obliczenia wykonywane są równocześnie
- Wyknowywanie w przeplocie (*ang. interlaced computing*) - obliczenia wykonywane są naprzemiennie

Tylko przetwarzanie równoległe oraz w przeplocie umożliwiają "równoczesne" wykonywanie obliczeń.

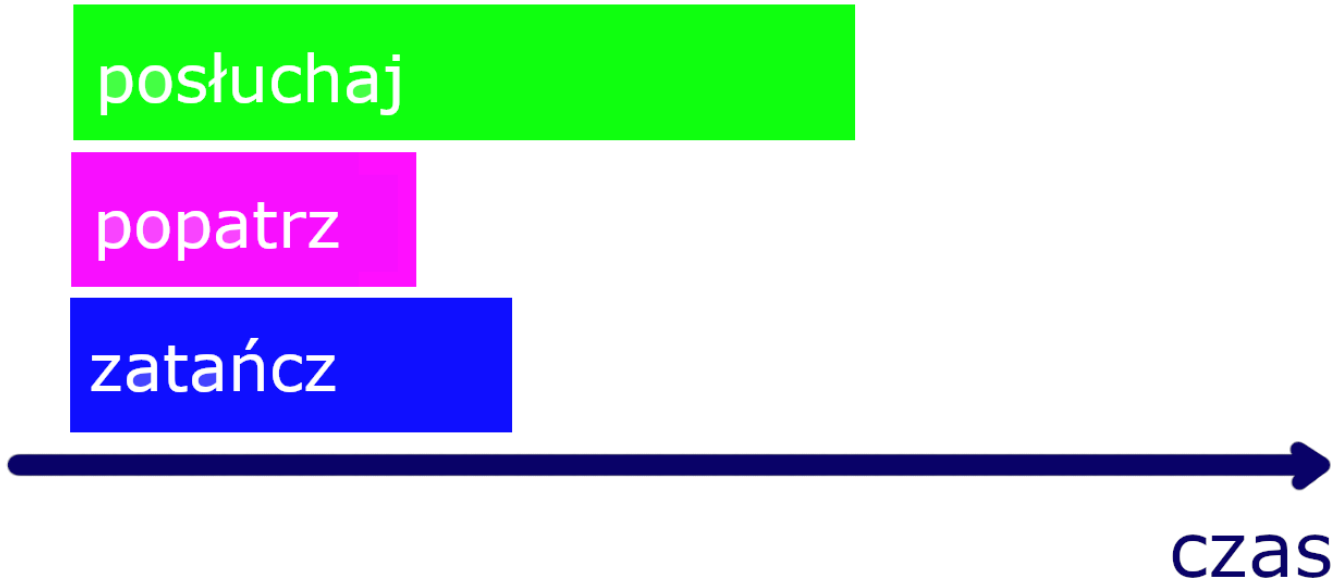
Zadanie

- zatańcz
- popatrz w oczy drugiej osobie
- posłuchaj Nad pięknym, modrym Dunajem

Trzy goffery. Jaka strategia rozwiązania tego zadania będzie najlepsza?

Podejście sekwencyjne

Podejście równoległe



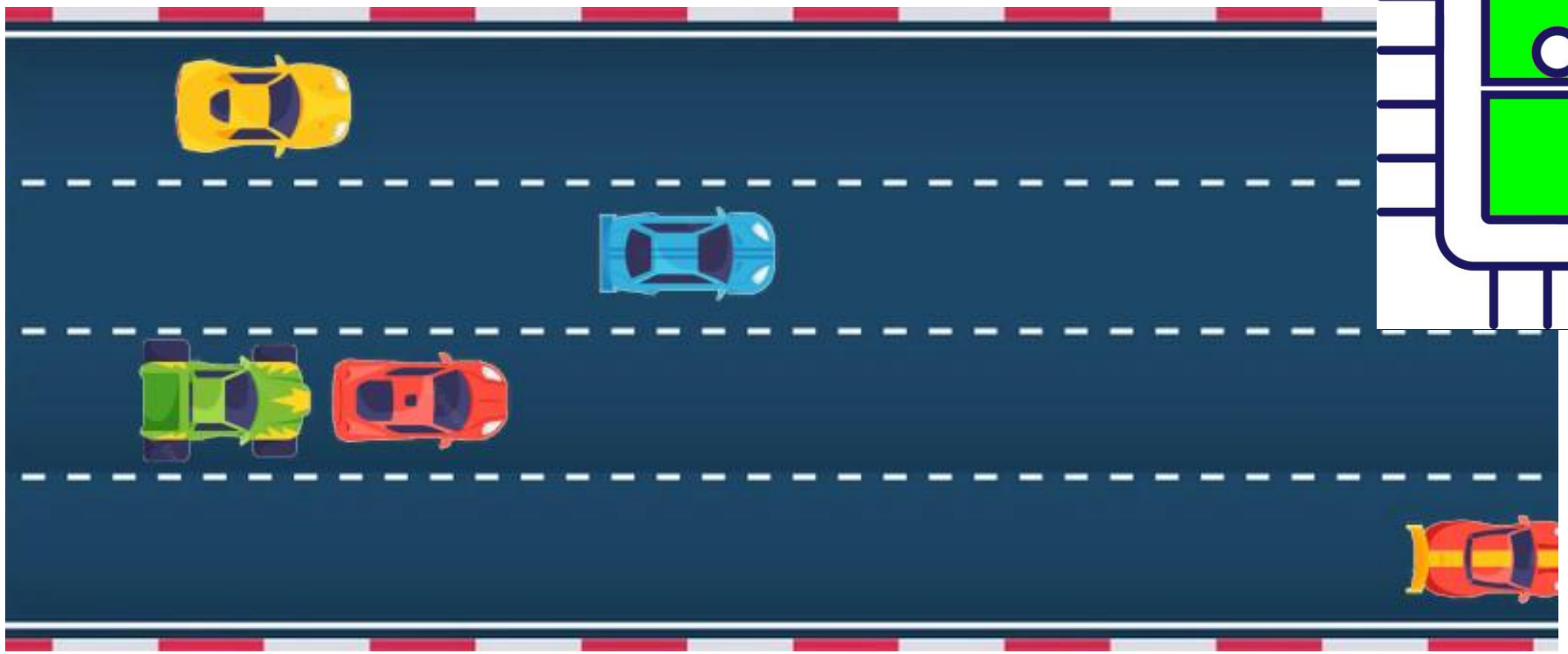
Przetwarzanie sekwencyjne
(sequential computing)

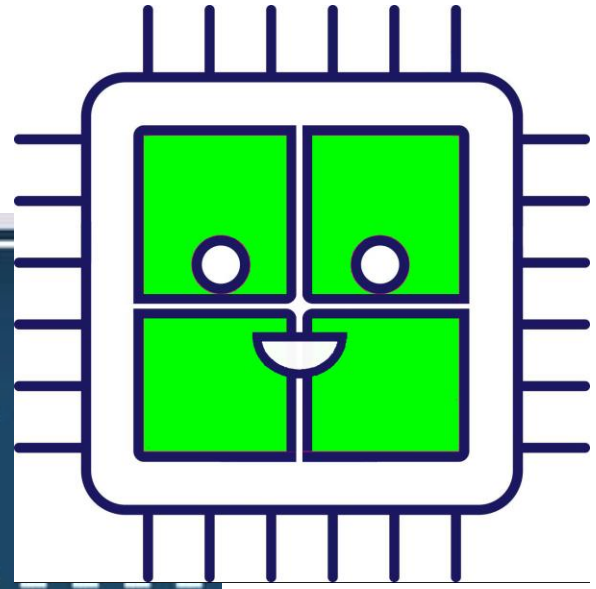
Przetwarzanie sekwencyjne
(sequential computing)

Przetwarzanie sekwencyjne
(sequential computing)

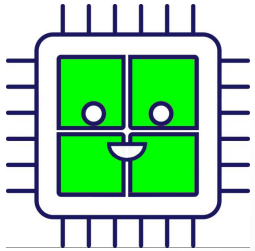
Przetwarzanie sekwencyjne
(sequential computing)

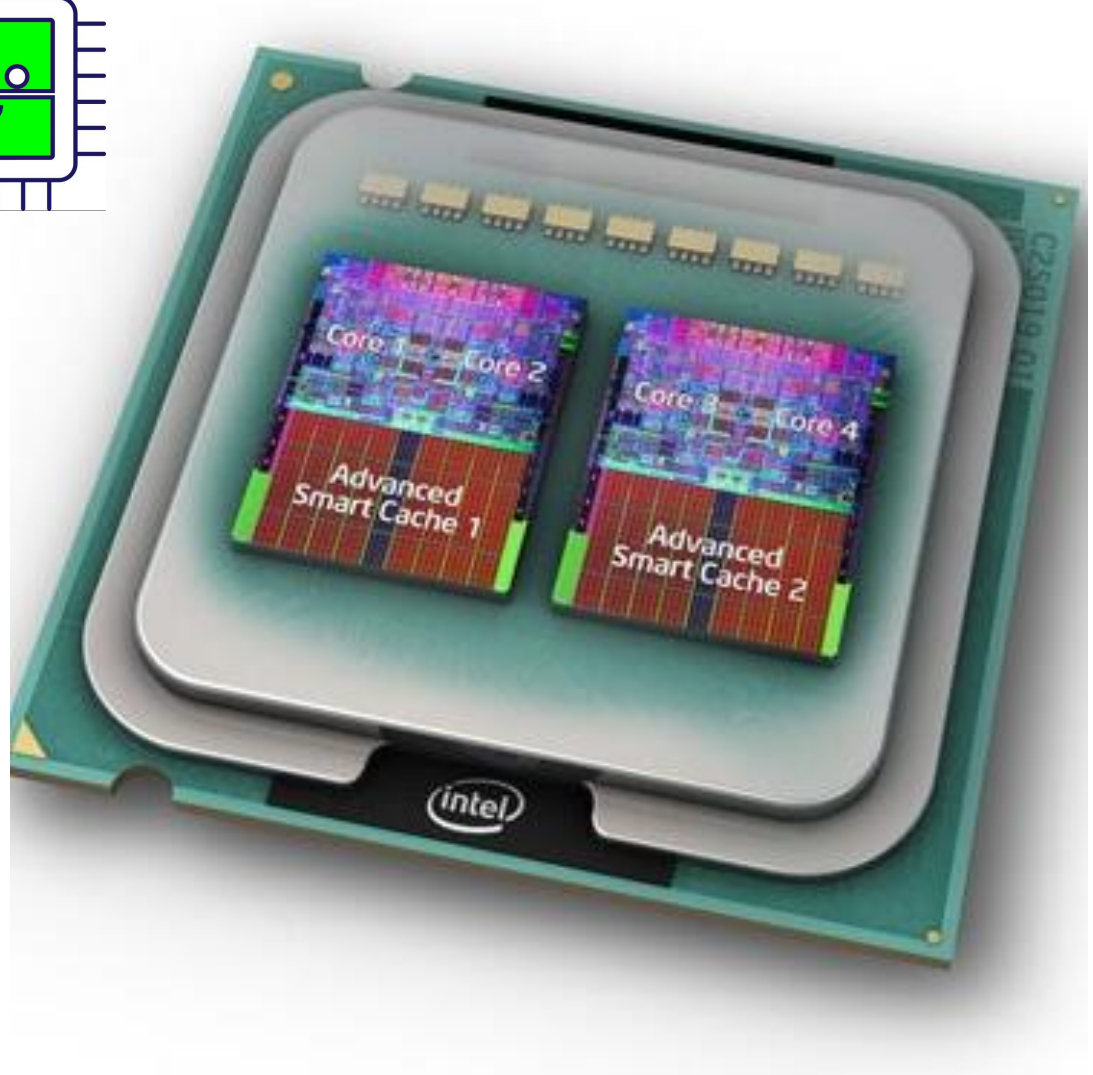
Przetwarzanie równoległe
(parallel computing)





Przetwarzanie równoległe - rdzenie





Przetwarzanie równoległe - rdzenie

Pierwszy wielordzeniowy procesor wyprodukowała firma IBM w 2001 roku.

Czy przed rokiem 2001 na komputerach nie można było wykonywać wielu obliczeń "równocześnie" ?

Przetwarzanie w przeplocie
(concurrent computing)

Zadanie

- przeczytaj (nie na głos) rozdział na temat nienadzorowanych metod uczenia maszynowego
- porozmawiaj ze znajomym/ą o planach na najbliższe wakacje

Dwa goffery. Jaka strategia rozwiązania tego zadania będzie najlepsza?

Podejście sekwencyjne

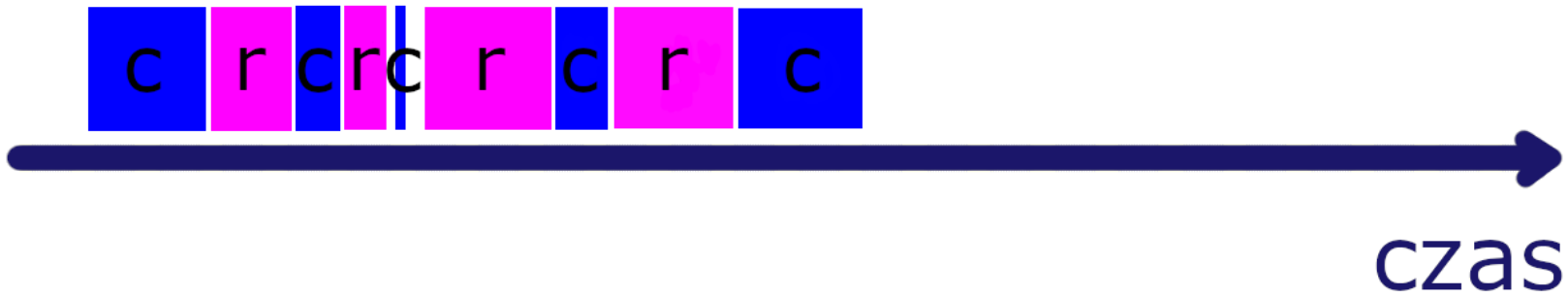
Podejście równoległe

Podejście równoległe

Wygląda na to, że nasz mózg wyposażony jest tylko w jeden "rdzeń" przeznaczony do analizy semantycznej słów. Nie jesteśmy w stanie ze zrozumieniem przetwarzać **jednocześnie** treści pochodzących z kilku różnych źródeł.

Czy możemy coś z tym zrobić?

Podejście w przeplocie



Możemy "przełączać" się pomiędzy zadaniami.

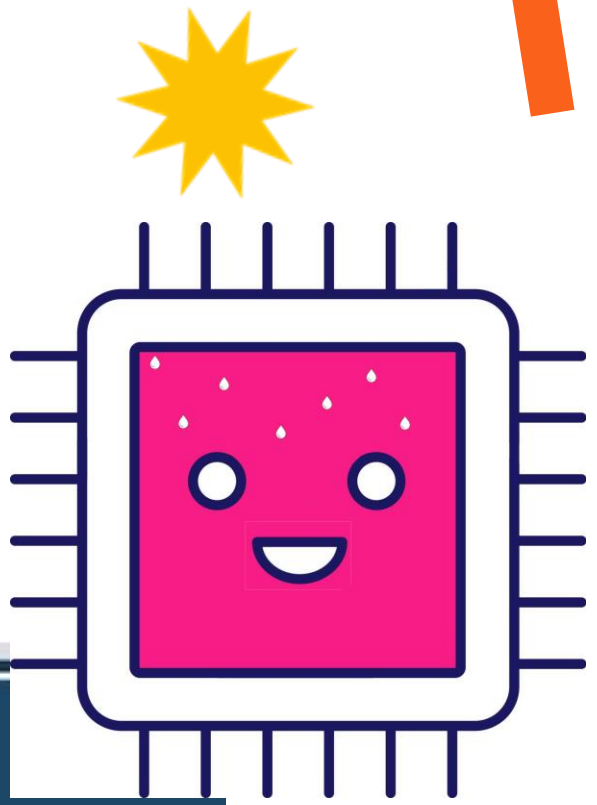
W nomenklaturze technicznej na określenie tej czynności często używa się zwrotu "przełączanie kontekstu" (*ang. context switching*)

Podejście w przeplocie

czas

Podejście w przeplocie

Przetwarzanie w przeplocie
(interlaced computing)



Przetwarzanie w przeplocie – mechanizm przełączania kontekstu

Kiedy procesor przełącza kontekst, to znaczy kiedy przerywa wykonywanie jednego wątku i zaczyna realizację innego?

Istnieją dwa możliwe mechanizmy realizujące taką operację:

- *współpraca (**cooperative multitasking**) - wątek sam decyduje kiedy oddać czas procesora innym wątkom*
- *wywłaszczenie (**pre-emptive multitasking**) - o dostępie wątków do procesora decyduje systemowy zarządca (scheduler)*

Rys historyczny

Procesory z pojedynczym rdzeniem

Pierwszy wielordzeniowy procesor wyprodukowała firma IBM w 2001 roku. Wcześniej komputery były w stanie jedynie zasymulować realizowanie kilku obliczeń równocześnie. Realizowały to za pomocą tzw. ***przełączania kontekstu*** *(context switching). Ta technika szeroko wykorzystywana jest do dzisiaj. Czas działania procesora rozdzielany jest przez planistę (ang. scheduler) pomiędzy wszystkie instrukcje wysłane do procesora. Procesory są w stanie wykonywać miliardy operacji na sekundę. Przełączanie się z tak wysoką częstotliwością wywołuje u ludzi złudzenie równoczesnego działania.*

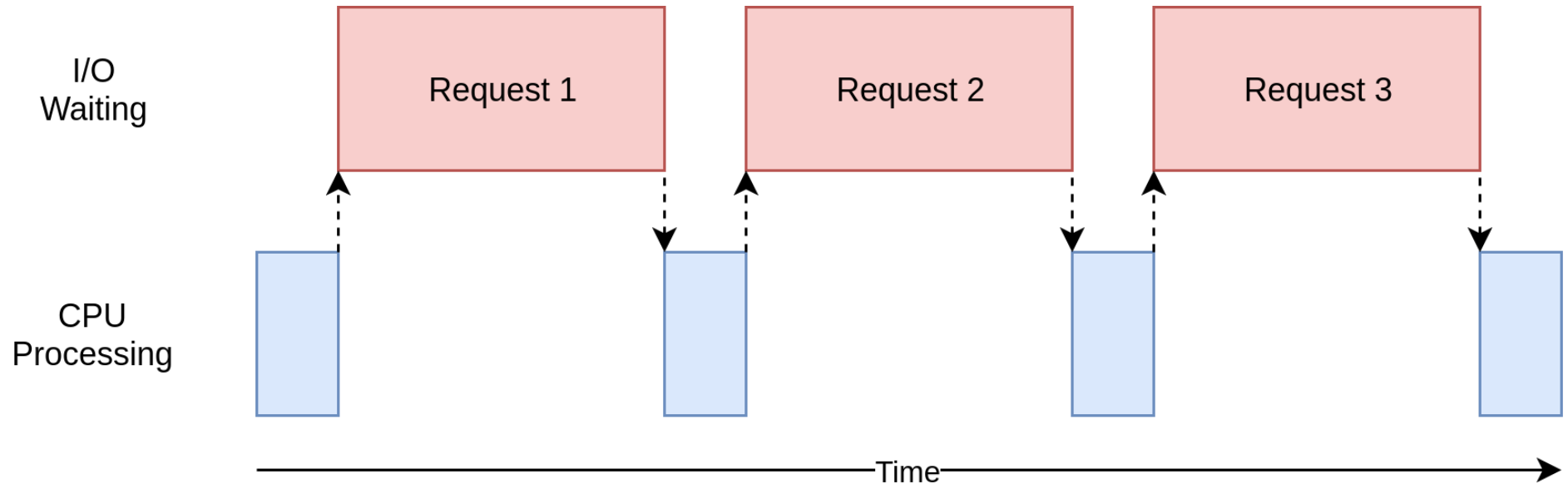
Przetwarzanie w przeplocie vs Przetwarzanie równoległe

Procesory z wieloma rdzeniami

Dopiero stworzenie procesora posiadającego wiele rdzeni (niezależnych jednostek obliczeniowych) umożliwiło faktyczną równoczesność obliczeń. W tym samym czasie każdy z rdzeni może zająć się innym obliczeniem. Liczba rdzeni procesora oznacza liczbę operacji, które procesor jest w stanie wykonywać równocześnie.

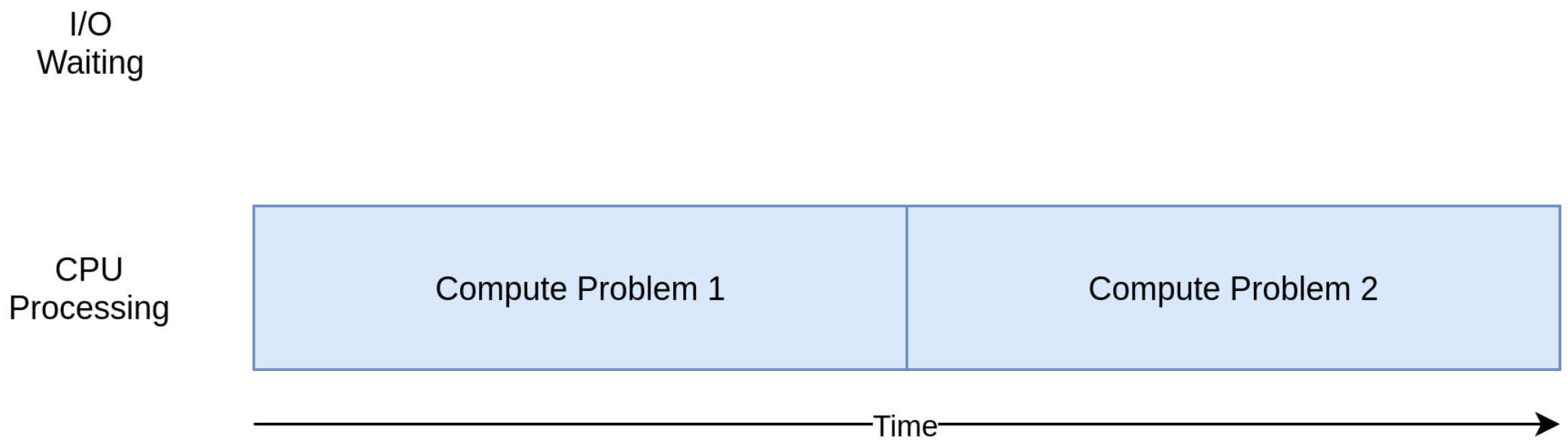
Przetwarzanie w przeplocie vs Przetwarzanie równoległe

i/o-bound



Przetwarzanie w przeplocie vs Przetwarzanie równoległe

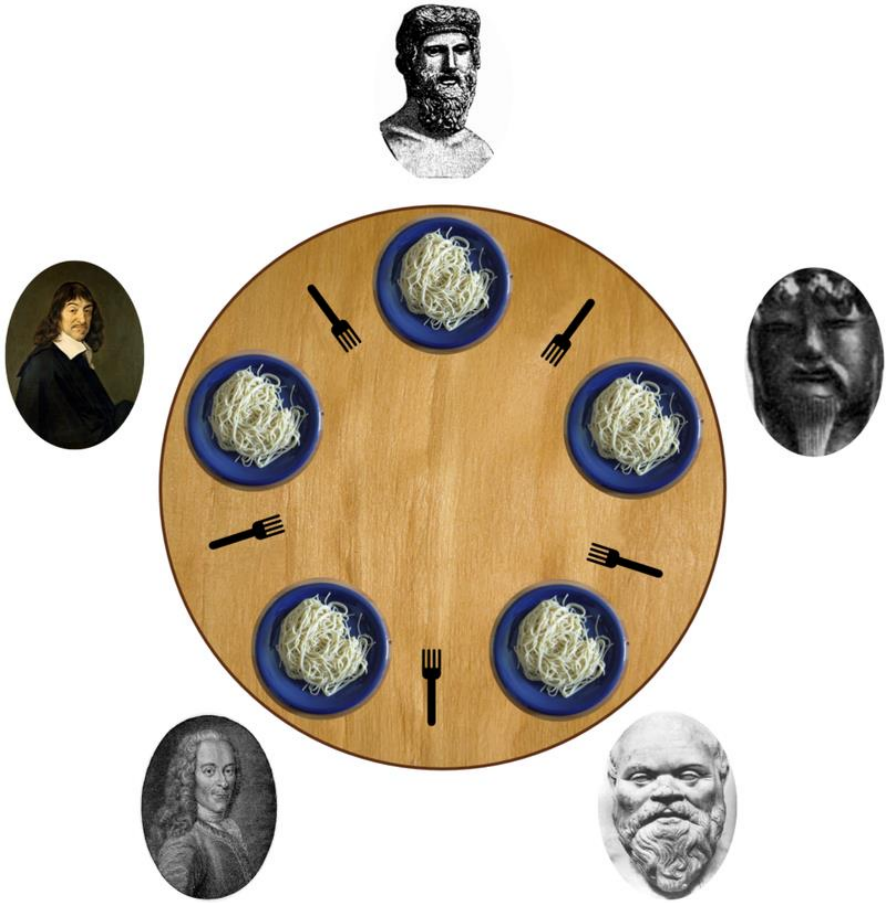
cpu-bound



Problem ucztujących filozofów

problemy synchronizacji przy współdzieleniu zasobów

Dijkstra 1965 (+ Ch. Hoare)



- *Zagłodzenie*
- *Zakleszczenie*

Dziękujemy!

software
development
academy