Sprawozdanie Lab 3

Mateusz Cyganek

# Java

## 5 Filozofów

Średnie dla 50 iteracji (posiłków)

|  |  |  |
| --- | --- | --- |
| **Podejście Naiwne** | **Podejście Asymetryczne** | **Podejście z Kelnerem** |
| P 4 finished, avg= 4,06ms  P 1 finished, avg= 4,06ms  P 3 finished, avg= 4,06ms  P 0 finished, avg= 4,06ms  P 2 finished, avg= 4,06ms | P 4 finished, avg= 3,96ms  P 3 finished, avg= 3,96ms  P 0 finished, avg= 4,04ms  P 1 finished, avg= 4,04ms  P 2 finished, avg= 4,04ms | P 4 finished, avg= 3,84ms  P 3 finished, avg= 4,10ms  P 1 finished, avg= 4,10ms  P 2 finished, avg= 4,10ms  P 0 finished, avg= 4,22ms |

Na wykresie przedstawiono poszczególnych filozofów oraz ich średni czas oczekiwania pomiędzy posiłkami w ms dla trzech podejść zaimplementowanych w Java.

W podejściu naiwnym wszyscy oczekiwali średnio tyle samo czasu, natomiast w pozostałych metodach, czas był krótszy oraz różnił się pomiędzy filozofami, co jest bardziej widoczne w wersji z arbitrem.

## 10 Filozofów

Średnie dla 50 iteracji (posiłków)

|  |  |  |
| --- | --- | --- |
| **Podejście Naiwne** | **Podejście Asymetryczne** | **Podejście z Kelnerem** |
| P 3 finished, avg= 4,06ms  P 9 finished, avg= 4,06ms  P 8 finished, avg= 4,10ms  P 0 finished, avg= 4,10ms  P 2 finished, avg= 4,14ms  P 1 finished, avg= 4,14ms  P 7 finished, avg= 4,18ms  P 6 finished, avg= 4,18ms  P 5 finished, avg= 4,18ms  P 4 finished, avg= 4,18ms | P 6 finished, avg= 4,06ms  P 5 finished, avg= 4,06ms  P 2 finished, avg= 4,08ms  P 1 finished, avg= 4,12ms  P 8 finished, avg= 4,12ms  P 9 finished, avg= 4,12ms  P 3 finished, avg= 4,14ms  P 4 finished, avg= 4,14ms  P 0 finished, avg= 4,14ms  P 7 finished, avg= 4,14ms | P 4 finished, avg= 6,06ms  P 0 finished, avg= 6,06ms  P 2 finished, avg= 6,10ms  P 3 finished, avg= 6,10ms  P 5 finished, avg= 6,04ms  P 1 finished, avg= 6,10ms  P 9 finished, avg= 6,08ms  P 8 finished, avg= 6,16ms  P 7 finished, avg= 6,16ms  P 6 finished, avg= 6,16ms |

Na wykresie przedstawiono poszczególnych filozofów oraz ich średni czas oczekiwania pomiędzy posiłkami w ms dla trzech podejść zaimplementowanych w Java.

Przy większej liczbie filozofów, podejście naiwne i asymetryczne wypadają lepiej od rozwiązania z arbitrem.

# Java Script

## 5 Filozofów

Średnie dla 50 iteracji (posiłków)

|  |  |  |
| --- | --- | --- |
| **Podejście Naiwne** | **Podejście Asymetryczne** | **Podejście z Kelnerem** |
| P 0 finished, avg= 34.94ms  P 1 finished, avg= 40.70ms  P 2 finished, avg= 33.58ms  P 3 finished, avg= 35.40ms  P 4 finished, avg= 37.96ms | P 0 finished, avg= 46.60ms  P 1 finished, avg= 46.72ms  P 2 finished, avg= 50.54ms  P 3 finished, avg= 112.58ms  P 4 finished, avg= 48.84ms | P 0 finished, avg= 120.44ms  P 1 finished, avg= 120.02ms  P 2 finished, avg= 119.44ms  P 3 finished, avg= 119.98ms  P 4 finished, avg= 120.72ms |

Na wykresie przedstawiono poszczególnych filozofów oraz ich średni czas oczekiwania pomiędzy posiłkami w ms dla trzech podejść zaimplementowanych w Java Script.

Czasy są o rząd większe niż w Javie a wersja z kelnerem jest znacznie powolniejsza od pozostałych. Jeden z filozofów w podejściu asymetrycznym ma znacznie dłuższy średni czas. Najprawdopodobniej nie udzielono mu przez kilka kolejek dostępu do stołu.

## 10 Filozofów

Średnie dla 50 iteracji (posiłków)

|  |  |  |
| --- | --- | --- |
| **Podejście Naiwne** | **Podejście Asymetryczne** | **Podejście z Kelnerem** |
| P 0 finished, avg= 31.96ms  P 1 finished, avg= 30.66ms  P 2 finished, avg= 31.96ms  P 3 finished, avg= 30.68ms  P 4 finished, avg= 31.96ms  P 5 finished, avg= 30.64ms  P 6 finished, avg= 31.98ms  P 7 finished, avg= 30.70ms  P 8 finished, avg= 31.94ms  P 9 finished, avg= 30.70ms | P 0 finished, avg= 49.70ms  P 1 finished, avg= 47.18ms  P 2 finished, avg= 50.72ms  P 3 finished, avg= 48.10ms  P 4 finished, avg= 51.00ms  P 5 finished, avg= 48.34ms  P 6 finished, avg= 49.60ms  P 7 finished, avg= 50.40ms  P 8 finished, avg= 50.26ms  P 9 finished, avg= 48.34ms | P 0 finished, avg= 264.44ms  P 1 finished, avg= 263.98ms  P 2 finished, avg= 263.46ms  P 3 finished, avg= 264.72ms  P 4 finished, avg= 266.08ms  P 5 finished, avg= 267.42ms  P 6 finished, avg= 266.08ms  P 7 finished, avg= 267.40ms  P 8 finished, avg= 269.12ms  P 9 finished, avg= 269.92ms |

Na wykresie przedstawiono poszczególnych filozofów oraz ich średni czas oczekiwania pomiędzy posiłkami w ms dla trzech podejść zaimplementowanych w Java Script.

Nie zdarzyło się zagłodzenie. Wersja z arbitrem jest znacznie wolniejsza.

# Porównanie

## Porównanie ogólne

Na wykresie przedstawiono wartości minimalne, maksymalne i średnie uzyskane z wszystkich podejść w Javie oraz JS dla 5 i 10 filozofów

Wszystkie implementacje w JS są znacznie wolniejsze a ilość filozofów ma znaczący wpływ na wyniki, w przeciwieństwie do swojego odpowiednika w Javie.

## Porównanie średnich

Na wykresie przedstawiono wartości średnie dla trzech podejść w Javie oraz JS dla 5 i 10 filozofów

W JS możemy zaobserwować spadek średniego czasu oczekiwania między posiłkami przy większej liczbie filozofów, pomijając implementacje z arbitrem w której to średnia jest dwukrotnie wyższa.

## Przypadek z 400 filozofami

Na wykresie przedstawiono wartości średnie dla trzech podejść w Javie oraz JS dla do 400 filozofów.

Czas nie różni się znacznie w podejściach asymetrycznych i naiwnych. Zauważyć można spory spadek wydajności dla podejścia z arbitrem w wersji JS.

# Wnioski

W przeciwieństwie do Javy, JS działa wyłącznie na jednym wątku, co jest jedną z przyczyn powolnego działania programu. Korzystanie z wielu procesów jednocześnie znacznie przyśpiesza działanie. JS nie wykorzystuje pełnej dostępnej mocy procesora. Minusem implementacji w Javie jest konieczność kolejkowania wątków.