Teoria współbieżności

Laboratorium 3

Problem pięciu filozofów

Przemysław Roman

03.11.2022

1 Rozwiązania problemów

1.1 Java

Implementacje w języku Java znajdują się w ścieżce `java\src\main\java\org\example`.

- Rozwiązanie z możliwością zagłodzenia - StarvingPhilosopher
- Rozwiązanie z arbitrem - JudgingPhilosopher

Obie klasy rozszerzają klasę Philosopher, w której znajdują się wspólne funkcjonalności.

1.2 JavaScript

Implementacje w języku JavaScript znajdują się w ścieżce `javascript\philosophers.js`.

2 Uruchamianie rozwiązań

Znajdując się w korzeniu dostarczonego archiwum:

2.1 Java

```
cd java && ./run.sh
```

2.2 JavaScript

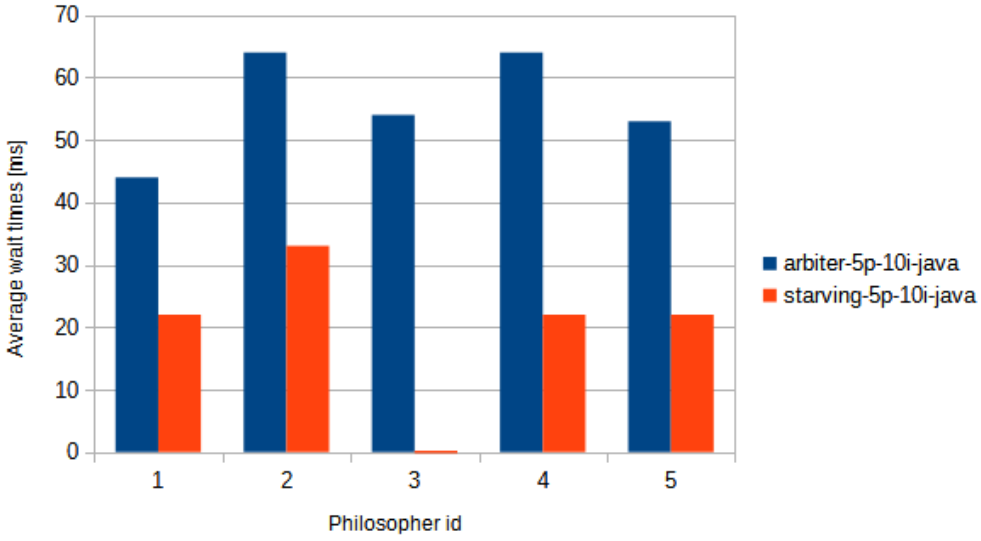
```
cd javascript && ./run.sh
```

3 Wyniki z komentarzami

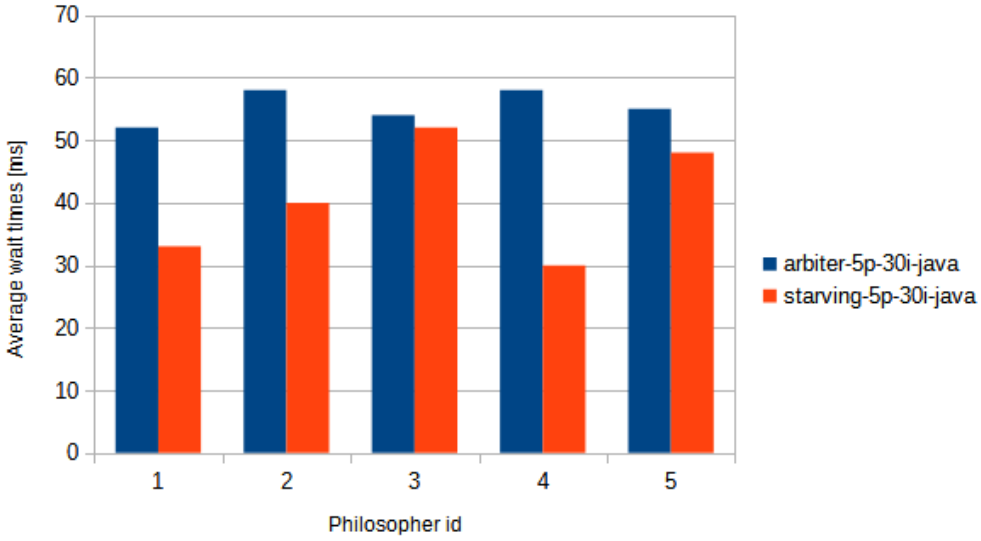
Dokonałem pomiarów średnich czasów oczekiwania filozofów na możliwość jedzenia. Pogrupowałem wyniki i sporządziłem wykresy, wyniki grupowałem po ilości filozofów oraz jedzonych posiłków. Na legendach dołączonych do wykresów korzystam ze skrótowych nazw:

- p - liczba filozofów
- i - liczba jedzonych posiłków (iteracji)
- arbiter - rozwiązanie z arbitrem
- starving - rozwiązanie z możliwością zagłodzenia
- asym - rozwiązanie asymetryczne
- pickupboth - rozwiązanie z jednoczesnym podnoszeniem widelców

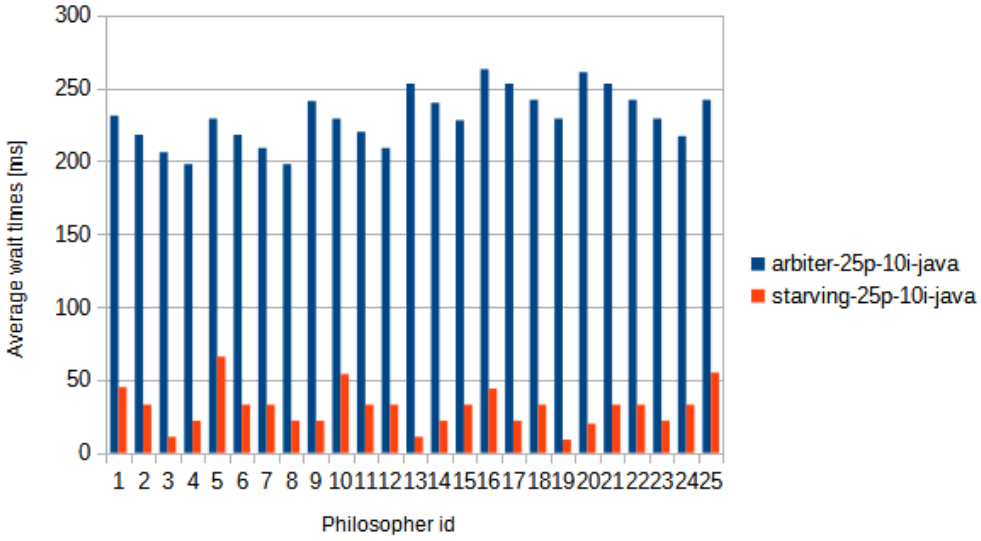
3.1 Java



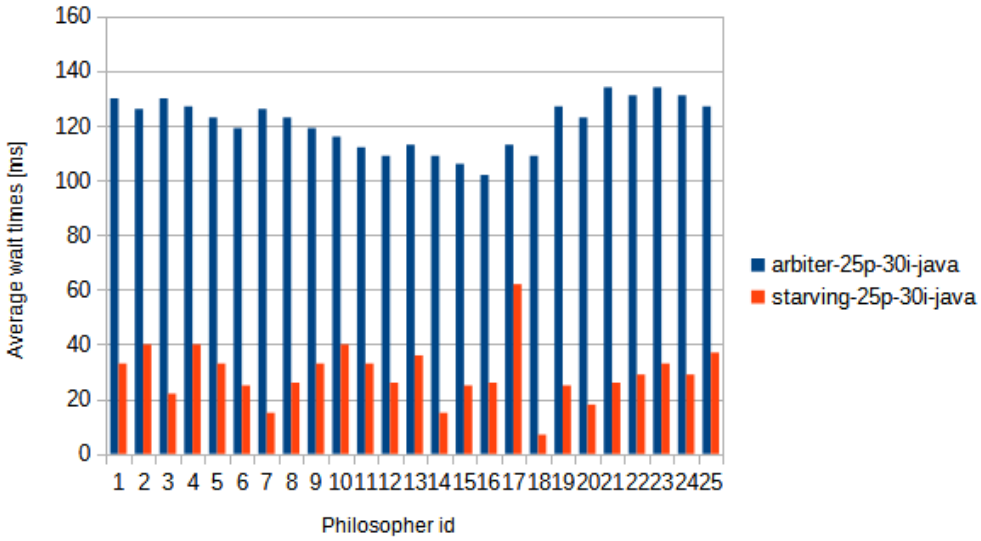
Udało się otrzymać przypadek, w którym jeden z filozofów w podejściu z możliwością zagłodzenia miał czas oczekiwania bliski zeru.



Powyższe wykresy dobrze pokazują jak nieprzewidywalne może być rozwiązanie z możliwością zagłodzenia. W pierwszym przypadku filozof nr. 3 nie czeka prawie w ogóle, w drugim czeka najdłużej ze wszystkich.



Można dostrzec pewien trend na wykresie - filozofowie dzielą się na grupy po 3-4. osoby, w każdej grupie średnie czasy oczekiwania można uporządkować.

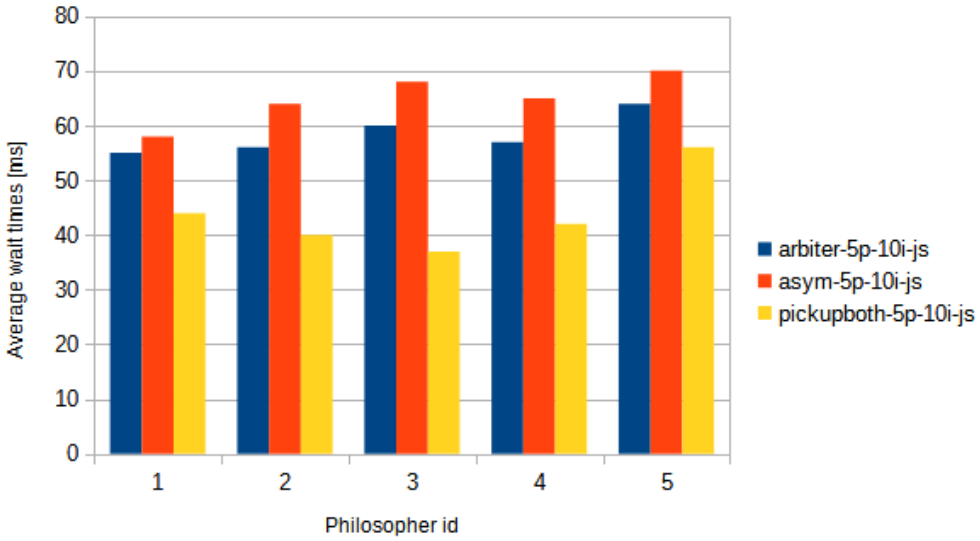


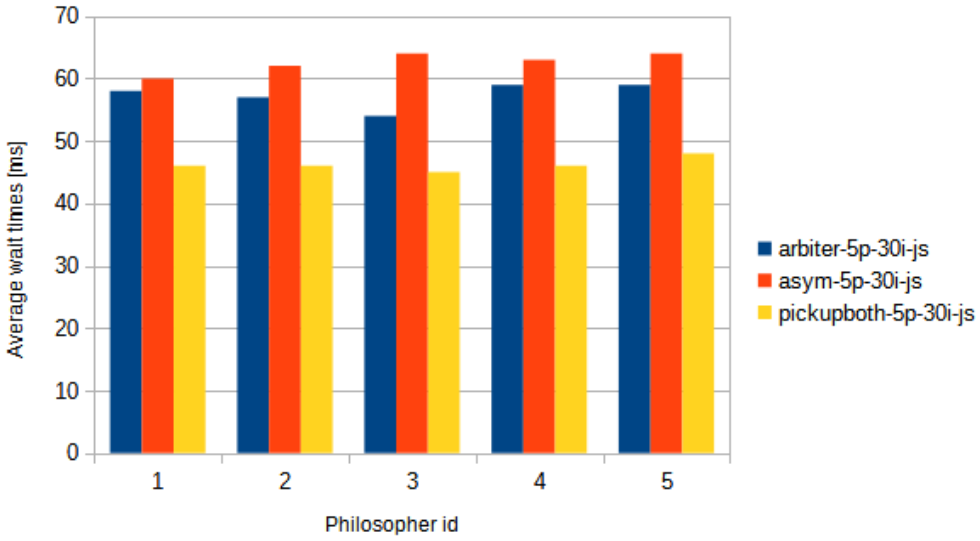
Porównując wykresy łatwo zauważyć, że rozwiązanie z możliwością zagłodzenia jest wydajniejsze. Możliwe, że spowodowane jest to doborem parametrów wejściowych, ponieważ rozwiązanie z arbitrem jest jednym z wzorcowych rozwiązań.

Kolejną obserwacją jest to, że rozwiązanie z arbitrem ma mniejsze odchylenie standardowe, najlepiej widać to na ostatnim wykresie.

Większa ilość iteracji obniżyła średni czas oczekiwania.

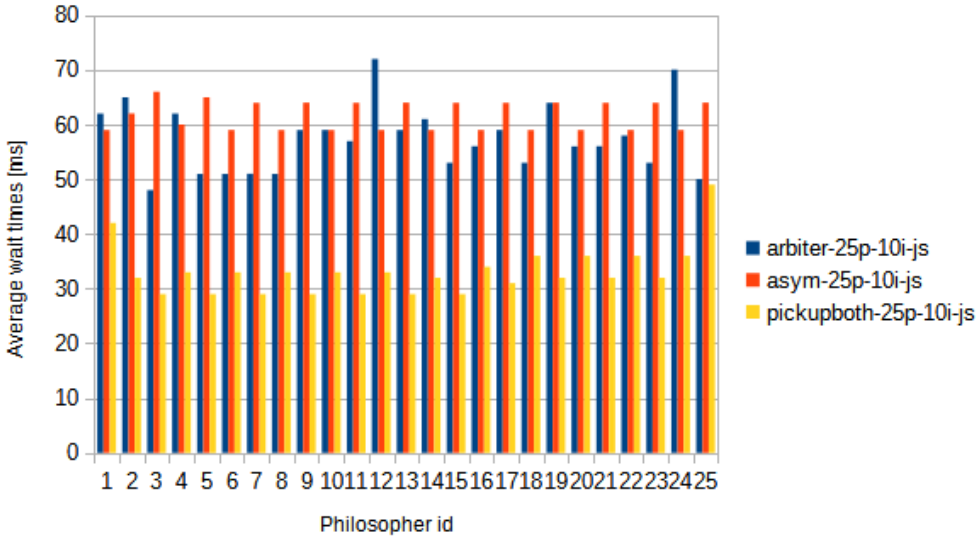
3.2 JavaScript

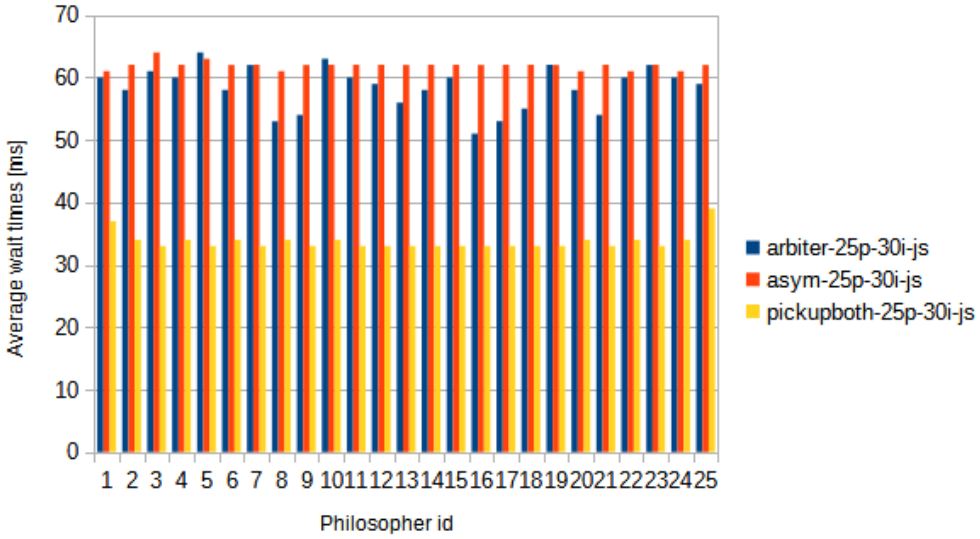




Porównując dwa powyższe wykresy można dojść do wniosku, że dla małej ilości filozofów rozwiązania można uporządkować (od najwydajniejszego):

1. Rozwiązanie z jednoczesnym podnoszeniem widelców
2. Rozwiązanie z arbitrem
3. Rozwiązanie asymetryczne





Dla większej ilości filozofów porządek wydaje się być zachowany, ale ciężko określić które rozwiązanie jest najmniej wydajne. Jeden filozof osiąga lepsze rezultaty asymetrycznym sposobem, inny rozwiązaniem z arbitrem.

Można stwierdzić, że podejście z jednoczesnym podnoszeniem widelców daje lepsze wyniki niż dla mniejszych testów (zarówno względem innych metod jak i bezwzględnie).

3.3 Porównanie między językami

Pierwsze, co rzuca się w oczy porównując wykresy wyników dla obu języków są wyniki dla arbitra. Implementacja w języku Java ma nawet 4-krotnie dłuższy czas wykonania niż implementacja w JavaScript. Spowodowane jest to doborem innej funkcji myślenia. W przypadku Javy jest to funkcja stała, natomiast w przypadku JavaScriptu jest to funkcja losowa. Zdecydowałem się na wybór funkcji losowej, ponieważ dla stałej wartości program wykonywał się znacznie dłużej. Dodatkowo pomyślałem, że dobrze będzie uwzględnić tę obserwację w sprawozdaniu.

Rozważając tylko średni czas oczekiwania podejście z zagłodzeniem wypada przeciętnie w porównaniu z implementacjami w JavaScripcie. Należy jednak uwzględnić, że jest mniej zbalansowane (bardziej faworyzuje osobników kosztem innych), więc lepiej go nie używać.