Projekt na przedmiot
Technologie Internetu Rzeczy

Autorzy:
Łukasz Wolski
Andrii Trishch
Konrad Siuzdak
Stanislav Shelemekh

1.Cel projektu

Celem projektu jest opracowanie algorytmu rozmieszczania procesów przetwarzania danych w oparciu o algorytmy stosowane do rysowania grafów–ang. Force-directed graph drawing. Opracowane rozwiązanie powinno być zweryfikowane za pomocą symulatora IoTSim-Edge.
https://en.wikipedia.org/wiki/Force-directed_graph_drawing
https://arxiv.org/abs/1910.03026
https://arxiv.org/pdf/2005.11847.pdf

2. Przebieg prac

Etap 1

Dowiedzieliśmy się do czego możemy wykorzystać poszczególne technologie,zapoznaliśmy się z terminologią, stworzyliśmy własną ideę co chcemy zrobić i jaki jest nasz cel projektu. Próbowaliśmy uruchomić IoTSim-Edge, ale próby zakończyły się niepowodzeniem i zamierzamy kontynuować próby równolegle pracując nad algorytmem.

Etap 2

Udało się skutecznie uruchomić IoTSim-Edge(instrukcja znajduje się w naszym repozytorium) oraz przyswoić jego strukturę. Doszliśmy jednak do wniosków, że to oprogramowanie nie jest w stanie w pełni odzwierciedlić naszych wyników, jakie uzyskamy na wyjściu naszego programu, co wynika głównie z różnicy plików konfiguracyjnych (możliwe byłoby jedynie zmodyfikowanie IoTSim-Edge pod nasze potrzeby co jednak nie jest naszym celem). Równolegle udało nam się zaimplementować szkielet algorytmu FDG. Teraz zamierzamy skupić się na stworzeniu wizualizacji działania naszego programu oraz zbadaniu jego poprawności.

Etap 3

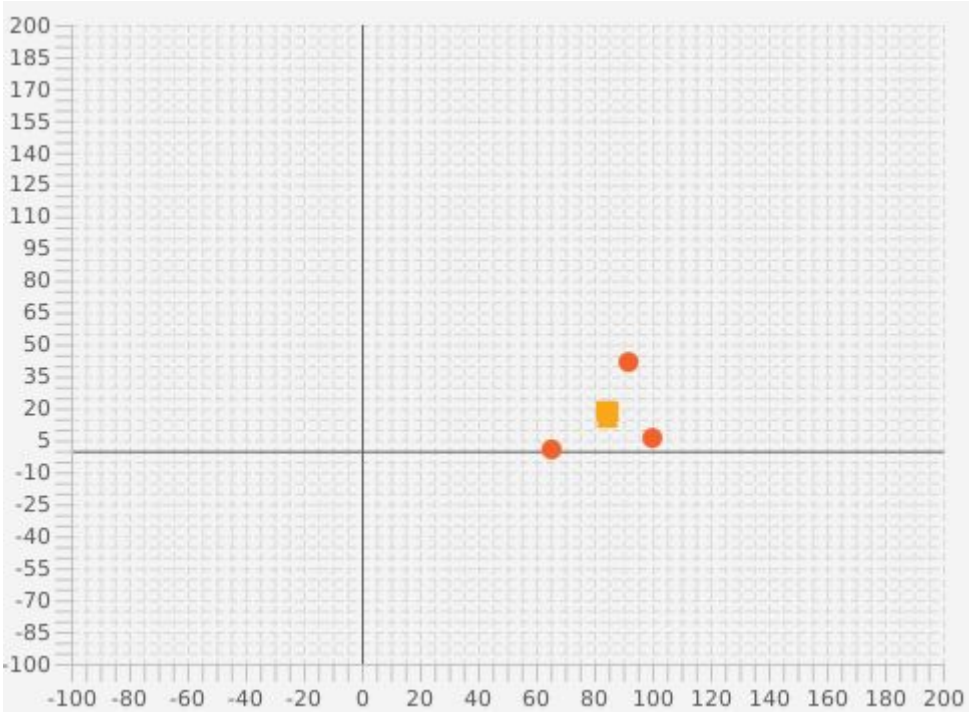
Stworzyliśmy wizualizację, prezentującą działanie programu. W okienku z układem współrzędnych, dla każdej iteracji wykonywanej z opóźnieniem odświeżane jest rozmieszczenie procesów i maszyn. Dodatkowo na konsoli wypisywane są współrzędne tych punktów. Nie zdążyliśmy jednak dopracować algorytmu przypisywania procesów do maszyn.

4. Rezultaty

Kod jest dostępny w repozytorium pod adresem:
https://github.com/konrrad/iot_calc_distribution

Zawiera się tam także zaimplementowany algorytm Force-Directed Graph Drawing wzięty z tej pracy naukowej:
https://www.researchgate.net/publication/304840569_Force-Directed_Algorithms

Głównie skupiliśmy się na tym, żeby zaimplementować ten algorytm w prawie że oryginalnej formie, ale biorąc pod uwagę niektóre aspekty naszego problemu. Na przykład zrobiliśmy tak, żeby maszyny i procesy mocniej się przyciągały do siebie, ale maszyny i maszyny już niekoniecznie.
Ważna uwagą jest to, że maszyny nie są zamocowane na stałe i poruszają się razem z procesami. Przyczyną tego była nasza chęć sprawdzenia teorii o tym, że za pomocą FDGD algorytmu będziemy w stanie lepiej rozmieścić wszystko w przestrzeni niż mocowanie maszyn na stałe.

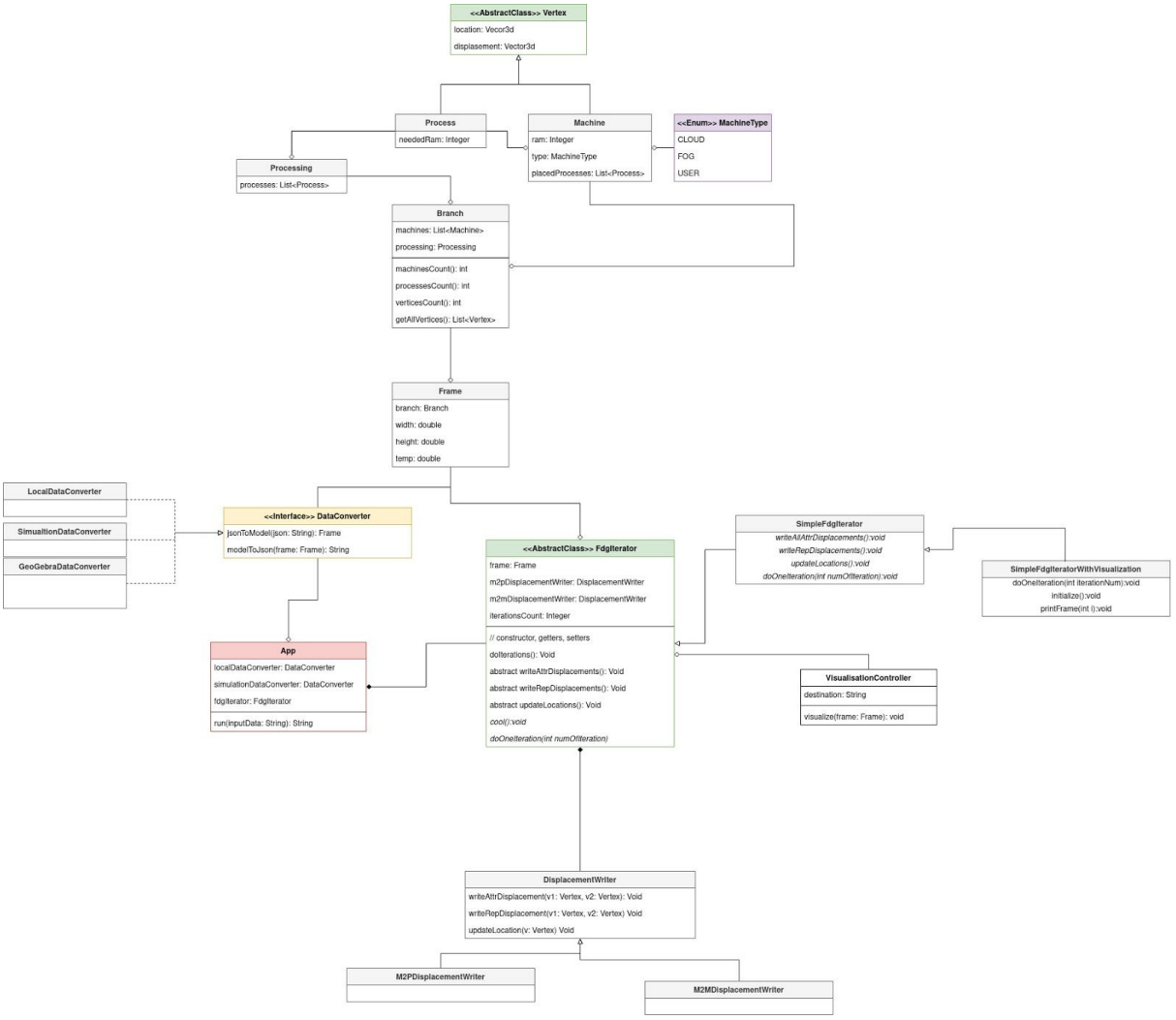



```
N: 101, T: 0.16873508277951976 | (x: 84, y: 19) (x: 85, y: 19) (x: 85, y: 16) (x: 92, y: 42) (x: 100, y: 6) (x: 65, y: 1)
```

Powyżej przedstawiono przykładową klatkę z animacji oraz wynik działania dla przykładowej konfiguracji. Widzimy, że maszyny (czerwone kółka) okrążyły procesy w trochę niepożądany sposób. Przyczyną tego jest fakt, że niestety nie zdążyliśmy zapewnić odpychanie się procesów na podstawie ich wymagań co do parametrów maszyn (na przykład RAM). Wraz z wprowadzeniem takiej modyfikacji do programu, mielibyśmy lepsze wyniki.

5. Struktura programu

W budowaniu struktury programu skupiliśmy się na tym, żeby zapewnić czytelność i łatwość zmiany kodu.



Po zaimplementowaniu tej części, została stworzona animacja prezentująca każdy krok algorytmu Force-directed Graph Drawing przy użyciu JavaFX.