Modele kolejkowe w IT
Automatyka i Robotyka (II stopień) 2022/2023

Symulacja modelu serwera Web

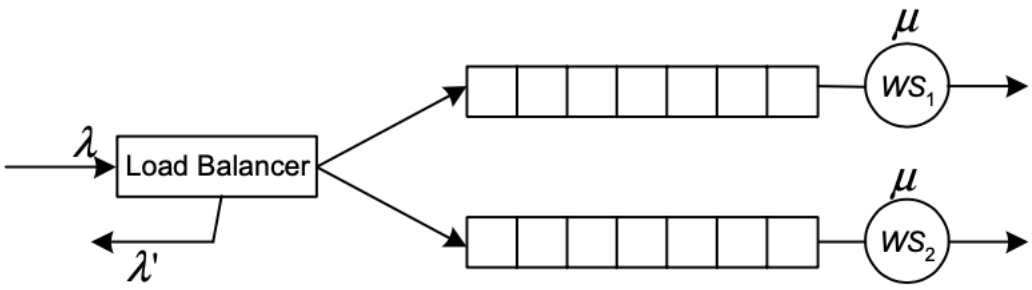
Skład zespołu: Marcin Bereźnicki,
Kamil Baradziej,
Jakub Burczyk,
Nicolas Duc

1. Wstęp

Celem projektu było zaprojektowanie i zasymulowanie systemu kolejkowego scentralizowanego modelu serwera Web. Model ten składał się z dwóch identycznych serwerów WWW, mających ograniczony rozmiar bufora. Przychodzące żądania użytkowników w pierwszej kolejności trafiają do load balancera, którego zadaniem jest przekierowanie zapytań do jednej z dwóch kolejek. Przekierowywanie żądań może odbywać się na dwa sposoby: przy użyciu polityki losowej lub najkrótszej ścieżki. Zapytania pojawiają się zgodnie z procesem Poissona o współczynniku przybywania λ. Czas obsługi w danym serwerze WWW podlega rozkładowi wykładniczemu o parametrze μ. Load balancer przekierowuje żądania bez opóźnień i są one obsługiwane w serwerach wg zasady FIFO. Jeśli serwer jest pełen, to żądania są odrzucane.

2. Opis zagadnienia

Schemat poglądowy analizowanego modelu został przedstawiono na rysunku 1. Widoczne są na nim dwa identyczne serwery Web (WS_1 i WS_2) oraz system równoważenia obciążenia.



Rysunek 1. Schemat modelu [Zhongju Zhang, Weiguo Fan: Web server load balancing: A queueing analysis, 2007].

Obsługa żądań różni się w zależności od zastosowanej metody routingu. W przypadku kiedy mamy do czynienia z polityką losową, można przyjąć, że strumień żądań jest podzielony na dwie równe części. W takiej sytuacji, dla każdego serwera, zapytania pojawiają się zgodnie z procesem Poissona o współczynniku przybywania $\lambda/2$. Dzięki temu można założyć, że są to dwa osobne serwery z identycznymi kolejkami M/M/1/FIFO/K. W przypadku polityki najkrótszej ścieżki, sprawa jest nieco bardziej skomplikowana. W tej sytuacji bowiem, load balancer, po otrzymaniu zgłoszenia, sprawdza liczbę żądań w obu kolejkach serwerów i wysyła obsługiwane zgłoszenie do krótszej kolejki. W przypadku, kiedy są one równej długości, przekierowanie odbywa się do kolejki nr 1. Zastosowanie zasady obsługi FIFO pozwala na obsługę najstarszych zapytań w pierwszej kolejności.

3. Aplikacja

Aplikacja do utworzenia i zasymulowania modelu została stworzona w języku programowania Python. W celu przeprowadzenia dodatkowych obliczeń oraz wizualizacji wykorzystano dwie biblioteki: *numpy* oraz *matplotlib*. Język Python został wybrany ze względu na przyjemność i szybkość pisania kodu oraz proste i minimalistyczne wykresy symulacji.

Aplikacja składa się z czterech plików:

- main.py - główny plik pozwalający na wybór parametrów oraz rozpoczęcie symulacji,
- web_server.py - plik zawierający główną klasę Web serwera, pozwalający na utworzenie kolejek, wygenerowanie czasów przybywania i obsługi zgłoszeń, wyświetlenie wykresów oraz obliczenie niektórych parametrów,
- queue.py - plik zawierający klasy Queue oraz QueueShortest, pozwalające na utworzenie pojedynczej kolejki i zasymulowanie obsługi żądań,
- math_utils.py - plik odpowiadający za umożliwienie przeprowadzenia obliczeń wybranych prawdopodobieństw, średniego czasu obsługi czy średniej liczby zapytań w systemie.

Klasa WebServer podczas inicjalizacji przyjmuje wiele argumentów istotnych do stworzenia symulacji, tj. nazwę typu obsługi kolejki, liczbę kanałów, rozmiar buforów, częstotliwość obsługi, współczynnik przybywania czy całkowitą liczbę żądań. Parametry te muszą zostać zdefiniowane w pliku main.py przed rozpoczęciem obliczeń.

Klasa Queue podczas inicjalizacji przyjmuje parametry takie jak liczbę kanałów obsługi, rozmiar bufora, czasy przybywania i częstotliwości obsługi zgłoszeń, liczbę żądań oraz wektory czasów przybywania i obsługi kolejnych zgłoszeń. Działanie tej klasy można opisać w następujący sposób:

- WebServer tworzy dwie identyczne kolejki i generuje dla nich wektory czasów przybywania i obsługi kolejnych zgłoszeń.

- Queue dla każdego żądania sprawdza czy zakończyła się obsługa poprzedniego żądania i w zależności od tego modyfikuje wielkość kolejki.

Klasa QueueShortest podczas inicjalizacji przyjmuje liczbę kanałów obsługi, która pozostaje niezmienna podczas całego ćwiczenia i wynosi – 1. Metoda klasowa simulate_shortest_queue przyjmuje w każdej iteracji czas nadejścia oraz czas obsługi danego żądania. Działanie omawianej klasy można opisać w następujący sposób:

- WebServer dla każdej iteracji żądania, dla których są „z góry" znane czasy nadejścia oraz obsługi, sprawdza wielkość bufora w każdej z kolejek oraz wybiera ten, w którym rozmiar tego bufora jest mniejszy (zmienna: QueueShortest.queue_size) lub kolejkę numer 1, w przypadku, kiedy rozmiary buforów są jednakowe.
- QueueShortest dla pierwszego wywołania metody simulate_shortest_queue przypisuje czas rozpoczęcia „pracy" kolejki do zmiennej service_start. Następnie dla każdej kolejnej iteracji metody, obliczane jest, czy żądanie poprzednie zostało zakończone, jeśli zostało licznik wielkości bufora pozostaje bez zmian, w przeciwnym przypadku – zwiększa się o wartość 1.

Zaimplementowany sposób prezentacji wyników nie uwzględnia czasu na generowanych wykresach. Dziedziną funkcji są kolejne, analizowane żądania. Stąd czytanie wykresów należy przeprowadzać w taki sposób, że prezentowana jest wielkość kolejki dla serwera nr 1 oraz 2 w chwili, kiedy przerabiane jest żądanie numer x. Dla polityki najkrótszej kolejki dodatkowo, przy użyciu czerwonych gwiazdek na wykresie, oznaczono wybór kolejki, do której trafia przerabiane żądanie. W przypadku, kiedy symbol gwiazdki występuje na przecięciu się dwóch wykresów, oznacza to, że wyborem dla żądania została kolejka nr 1.

4. Wyniki

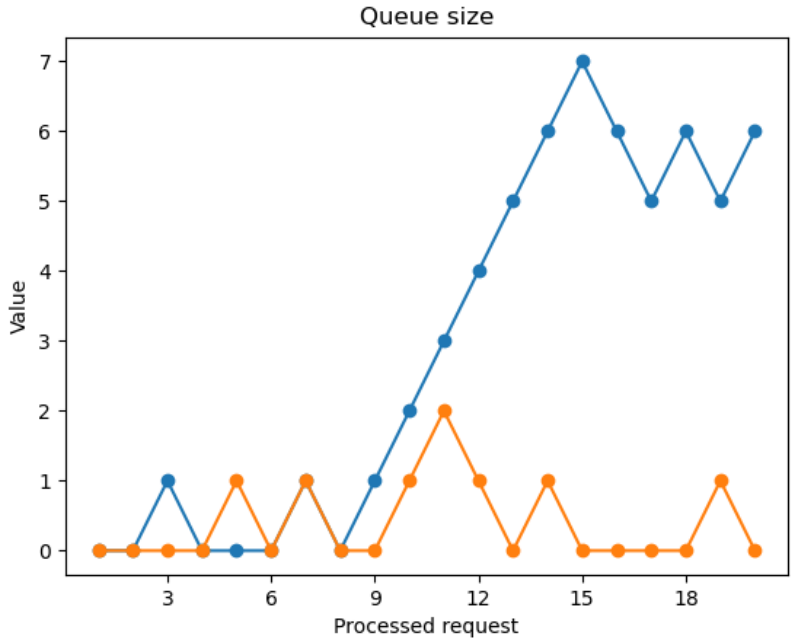
W celu oceny działania aplikacji przeprowadzono symulację dla obydwu polityk routingu oraz przeprowadzono krótkie porównanie otrzymanych wyników.

Test 1

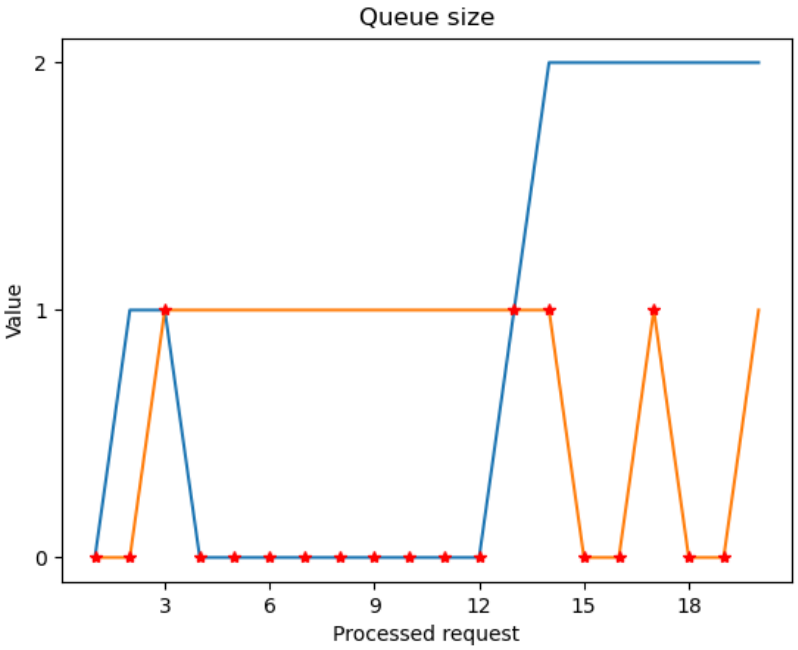
W pierwszym teście przeprowadzono eksperyment dla następujących danych wejściowych:

- rozmiar bufora „K" -> 10
- współczynnik przybywania zgłoszeń „λ" -> 20
- częstotliwość obsługi zgłoszeń „μ"-> 20
- liczba żądań „N"-> 40

Rezultat symulacji przedstawiono na rysunkach 2 i 3.



Rysunek 2. Polityka routingu: losowa.



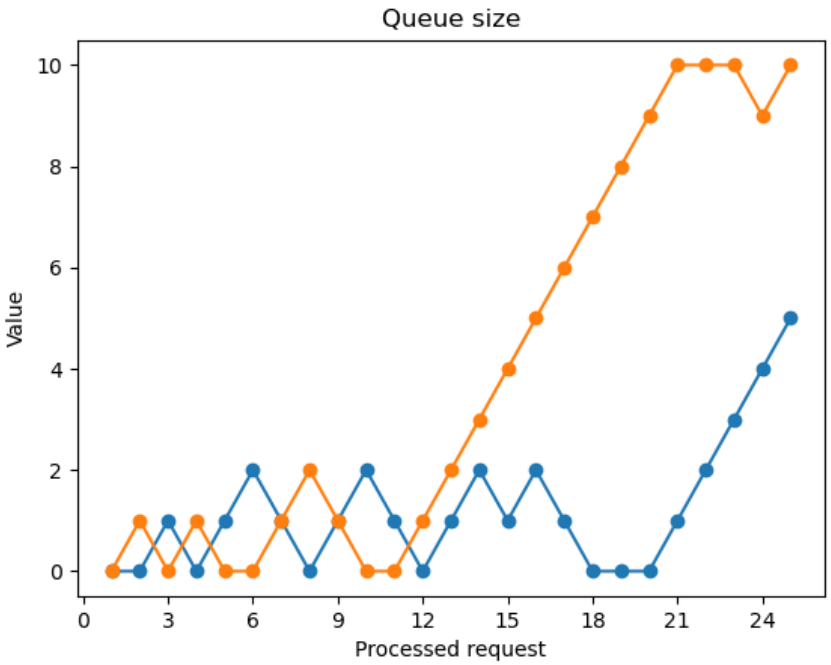
Rysunek 3. Polityka routingu: najkrótszej kolejki.

Test 2

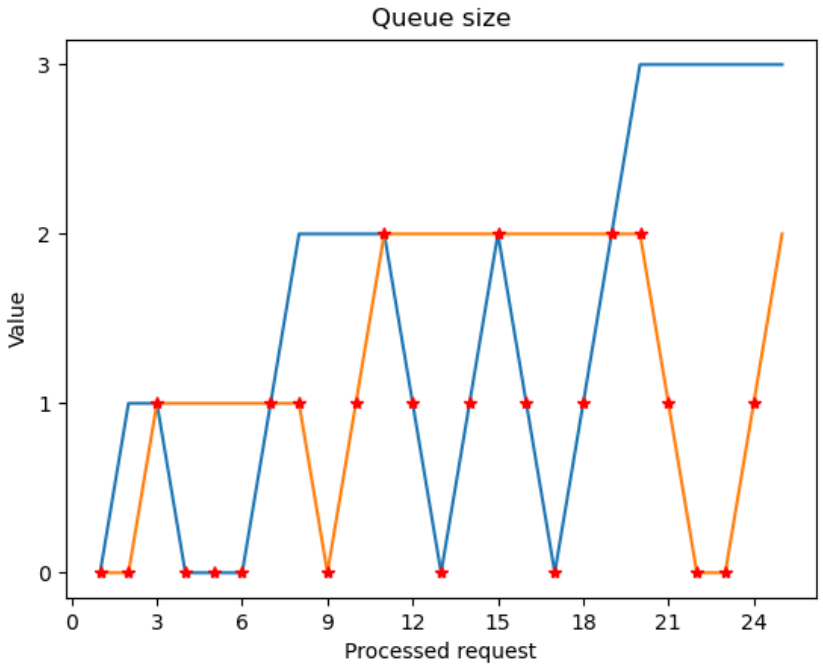
W drugim teście zmodyfikowano dane wejściowe w następujący sposób:

- rozmiar bufora „K” -> 10
- współczynnik przybywania zgłoszeń „λ” -> 50
- częstotliwość obsługi zgłoszeń „μ”-> 30
- liczba żądań „N”-> 50

Rezultat symulacji przedstawiono na rysunkach 4 i 5.



Rysunek 4. Polityka routingu: losowa.



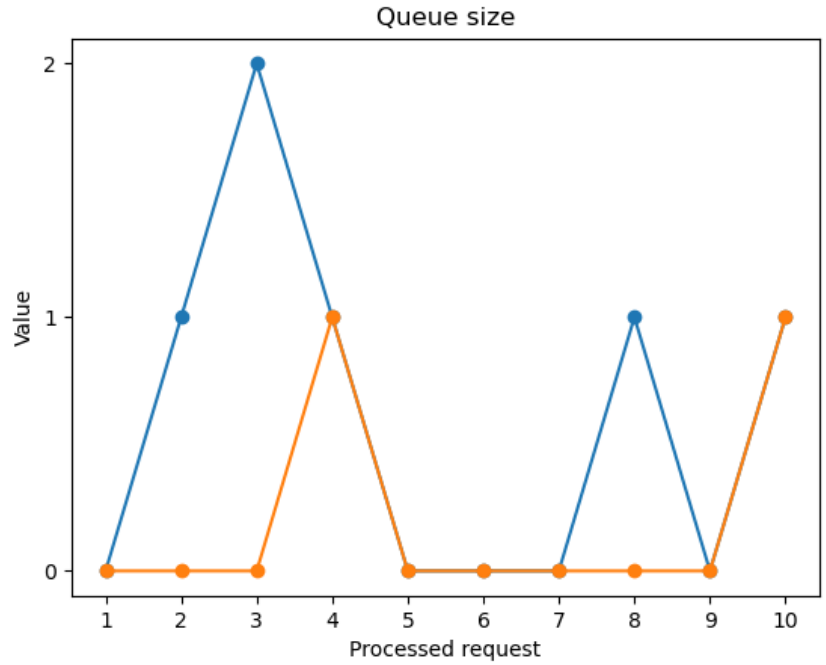
Rysunek 5. Polityka routingu: najkrótszej kolejki.

Test 3

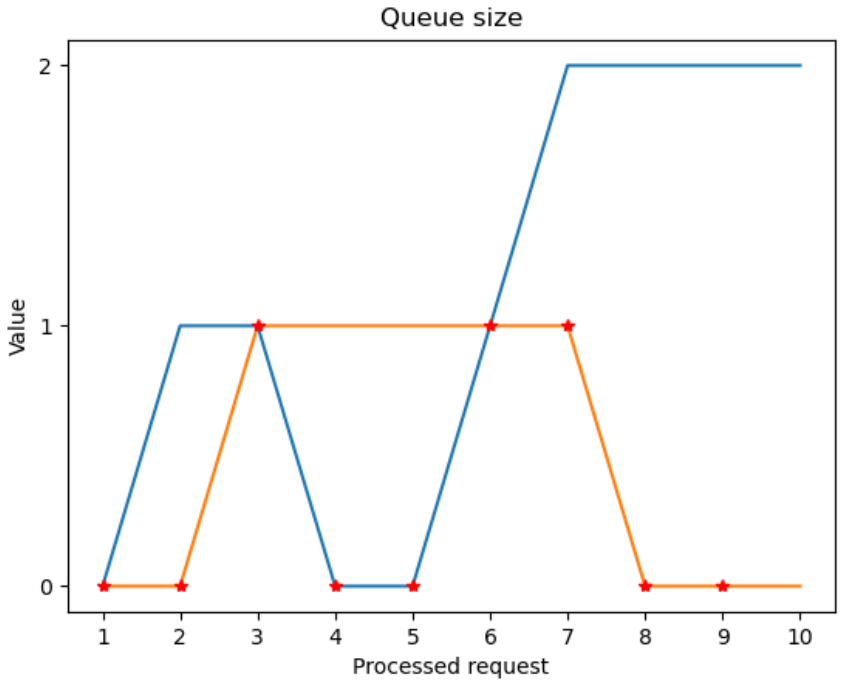
W trzecim teście postarano się tak dobrać parametry, aby rezultat działania polityki losowej był co najmniej nie gorszy od polityki najkrótszej ścieżki. W tym celu użyto następujących danych wejściowych:

- rozmiar bufora „K” -> 4
- współczynnik przybywania zgłoszeń „λ” -> 12
- częstotliwość obsługi zgłoszeń „μ”-> 15
- liczba żądań „N”-> 20

Rezultat symulacji przedstawiono na rysunkach 6 i 7.



Rysunek 6. Polityka routingu: losowa.



Rysunek 7. Polityka routingu: najkrótszej kolejki.

5. Podsumowanie

Zrealizowana aplikacja do tworzenia i symulowania modelu serwera Web pozwala na przeprowadzenie poprawnej analizy. Na podstawie uzyskanych wyników można stwierdzić, że zaimplementowane polityki różnią się od siebie. Polityka losowa działa zadowalająco, dopóki nie nastąpi zapełnienie jednej z kolejek. Niestety ze względu na losowość w otrzymywaniu żądań, może to nastąpić nieoczekiwanie szybko. Polityka najkrótszej ścieżki pozwala na wyeliminowanie tego problemu, poprzez każdorazowe sprawdzenie obecnego stanu kolejek, przed przekierowaniem żądania. Dzięki temu szansa na odrzucenie zgłoszenia jest mniejsza.

Warto również zwrócić uwagę na zależność dobranych parametrów oraz średniej długości kolejek. Analizując te parametry można zauważyć, że w przypadku testu pierwszego i drugiego gdy współczynnik przybywania zgłoszeń „λ" jest większy lub równy współczynnikowi obsługi zgłoszeń „μ" polityka losowa radzi sobie dobrze do czasu gdy zgłoszenia nie są wrzucane do jednej z kolejek częściej. Wtedy zgłoszenia zaczynają się nawarstwiać. Na Rysunku 2 widać że następuje wtedy wzrost ilości zgłoszeń nawet do 7, lub bufor przepełnia się i osiąga wartość 10 jak w przypadku testu drugiego (Rysunek 4). W przypadku polityki najkrótszej kolejki jednak zgłoszenia rozkładane są równomiernie między kolejkami co pozwala na znaczne zmniejszenie nawarstwiania się (Rysunki 3 i 5). Ilość zgłoszeń w kolejkach, pomimo że jest rosnąca to nie przekroczyła 3.

Zbadano również szczególny przypadek w teście nr 3, w którym współczynnik przybywania zgłoszeń był mniejszy od częstotliwości ich obsługi. W takim przypadku obie polityki zachowywały się dość porównywalnie, czasem z korzyścią dla losowej. Jest to spowodowane tym że obsługa zgłoszeń jest szybsza niż ich częstotliwość przychodzenia co znacznie zmniejsza prawdopodobieństwo nawarstwiania się zgłoszeń w jednej z kolejek.