Sprawozdanie z symulacji

Robert Kraut

7 czerwca 2020

1 Informacje wstępne

Serwer nieblokujący jaki będziemy badać to implementacja Netty - Reactor Netty. Zostały zebrane dane o serwerze ze strony reactor-netty.
Maksymalna długość kolejki oczekujących żądań wynosi 1000, maksymalna ilość połączeń 500 a 45s to maksymalny czas na uzyskanie połączenia.

2 Etap 1 - Przeprowadzenie rzeczywistej symulacji

Do przeprowadzanie rzeczywistego tak zwanego "stress testu" posłużę się programem JMeter.
Wszystkie testy zostają przeprowadzone na już rozgrzanej JVM, by móc się skupić na wydajności serwera.
Sprzęt, na którym testy zostały wykonanie to laptop Dell Inspiron o następujących parametrach:

- 8GB RAM
- 256GB pamięci SSD
- procesor Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz

z system operacyjnym LinuxMint 19.1 (tessa).
W sumie zostało wykonanych 15 testów

- 5000 Użytkowników(requestów)
- 10 000 Użytkowników
- 25 000 Użytkowników
- 50 000 Użytkowników
- 100 000 Użytkowników

Dla każdego zestawu, zostały wykonene 3 testy, jedno z opóźnieniem(czas wykonania żądania po stronie serwera)
100ms, jedno z 200ms i ostatnie 500ms. Każdy taki zestaw był(ilość wątków) była wykonywana na przestrzeni 10 sekund, można więc powiedzieć, że użytkownicy/10sek to nasze *requests/sec*.

Wyniki prezentują się nastepująco:

2.1 5k

Opóźnienie	% obsłużonych	średni czas odpowiedzi	minimalny czas odpowiedzi	maksymalny czas odp
100	100	105	100	210
200	100	204	200	310
500	100	506	500	634

2.2 10k

Opóźnienie	% obsłużonych	średni czas odpowiedzi	minimalny czas odpowiedzi	maksymalny czas odp
100	100	104	100	179
200	100	204	200	276
500	100	504	500	572

2.3 25k

Opóźnienie	% obsłużonych	średni czas odpowiedzi	minimalny czas odpowiedzi	maksymalny czas odp
100	100	109	100	207
200	100	203	200	275
500	100	503	500	572

2.4 50k

Opóźnienie	% obsłużonych	średni czas odpowiedzi	minimalny czas odpowiedzi	maksymalny czas odp
100	78.57	116	100	275
200	78.98	206	200	473
500	79.09	506	500	826

2.5 100k

Opóźnienie	% obsłużonych	średni czas odpowiedzi	minimalny czas odpowiedzi	maksymalny czas odp
100	77.42	120	100	1213
200	77.74	212	200	1264
500	78.07	512	500	1535

2.6 200k

Opóźnienie	% obsłużonych	średni czas odpowiedzi	minimalny czas odpowiedzi	maksymalny czas odp
100	76.23	123	100	1658
200	77.60	216	200	1648
500	78.69	517	500	2020

3 Etap 2 - Model

Ze względu na ograniczony czas pomijam specyfikację TCP/IP i skupiam się tylko na warstwie aplikacji smodelu ISO/OSI i protokole HTTP.
Jest wiele możliwych parametrów, które można rozważyć, m.in:

- ilość żądań na sekundę
- czas zaakceptowania połączenia
- czas jaki upłynął od momentu zaakceptowania żądania do splasowania odpowiedzi w buforze
- czas wykonania określonego zadania
- długość kolejki
- czas umieszczenia danych w buforze
- jak długo użytkownik czekał na odpowiedź
- odsetek obsłużonych użytkowników

Jednak w naszym przypadku skupimy się tylko na ilości żądań na sekundę, czasu wykonania określonego zadania oraz odsetku obsłużonych żądań.

4 Etap 3 - Symulacja

Symulacja będzie polegała na wizualizacji funkcji kilku zmiennych.
Jak już wcześniej wspomniane, by uprościć model zakładamy, że "TCP handshake" jest natychmiastowy oraz że nie wspieramy cachowania zapytań, jak i optymalizacji, które robi za nas przeglądarka internetowa. Nie bierzemy także pod uwagę stanu maszyny oraz systemu operacyjnego, na którym serwer działa Funkcja będzie mieć nastapującę stałe wynikające z specyfikacji serwera:

- 1000 - rozmiar kolejki
- 500 - maksymalna ilość jednoczesnych połączeń

oraz następujące parametry:

- d - delay, czas wykonania żądania

- us - ilość użytkowników na sekundę
- u - sumaryczna ilość użytkowników
- t - konkretna chwila w czasie
- p - odsetek obsłużonych żądań

Co z tych danych możemy wywnioskować, że w danym momencie może być maksymalnie tyle użytkowników, które serwer obsługuje lub może obsłużyć. Możemy także zauważyć, że im większe opóźnienie, tym większy procent, prawdopodobnie spowodowane jest to tym, że w czasie kiedy nic się nie dzieje wątek główny dalej może obsługiwać inne żadania.

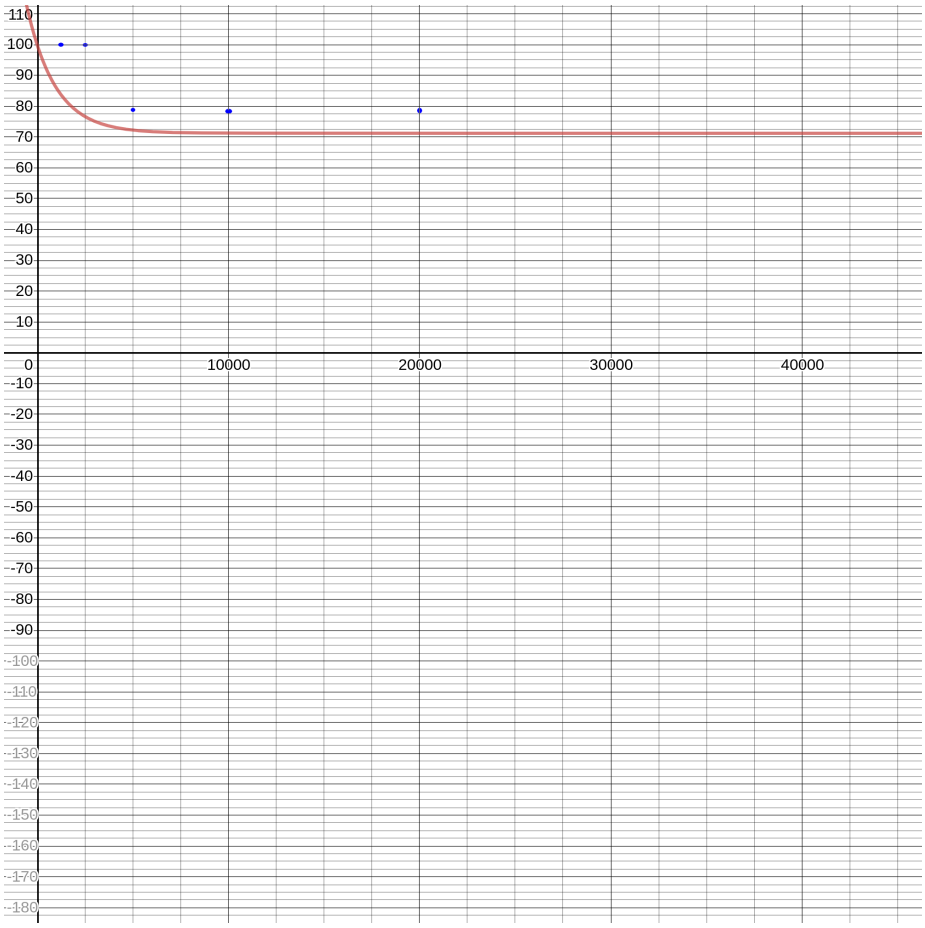
Dane z symulacji zostały wrzucone do programu interpolującego funkcję. Najbardziej odwzorującą wydaje być się funkcja eksponencjalna.
Jako oś x została wybrana liczba użytkowników, a jako oś y procent obsłużonych requestów.

$$y = 100 - \frac{d-3}{d} * 29 * (1 - e^{\frac{-x}{Q+M}}))$$

,gdzie x to liczba użytkowników na sekundę,
y to procent udanych obsłużeń użytkowników,
d to czas wykonania żądania(delay)
a Q i M to odpowiednio rozmiar kolejki oraz maksymalna liczba równoległych połączeń.

Jako iż według naszych testów odsetek obsłużonych requestów nie maleje liniowo, tylko istnieje jakiś punkt x_0,
przy którym osiągnięto maksymalną liczbę jednoczesnych połączęń oraz kolejka została zapełniona, od którego funkcja przyjmuje nagły spadek.
Z tych też właśnie powodów została wybrana funkcja eksponencjalna zaczynające się od punktu (0,100). Zostało to uwzględnione w naszej funkcji, gdzie nasze x dzielimy przez właśnię sumę tych dwóch czynników.
Kolejną rzeczą, którą trzeba było uwzględnić to to, że wraz ze wzrostem opóźnienia(czasu wykonania żądania), procent obsłużonych requestów rośnie. Jak spojrzymy na funkcję to widzimy, że mamy współczynnik $\frac{d-3}{d}$, czyli wraz z większym opóźnieniem ten współczynnik maleje. Cały odjemnik został jeszcze przemnożony przez liczbę pierwszą, aby była bliższa rzeczywistości.

Wykres tej funkcji dla $d = 300$ wygląda następująco.



5 Wnioski

Z wykresu można wywnioskować, że funkcja zbiega do ok. 70%.
Niestety funkcja za wcześnie zaczyna maleć, bo na wykresie widać, że od wartości $2500r/s$ funkcja drastycznie maleje, co jest niezgodne z prawdą, gdyż według naszych testów dopiero od $5000r/s$, jednakże jest wiele czynników, które na to wpływają jak między innymi: wybrana implementacja danego nieblokującego serwera, narzędzie do testowania jak i wersja Javy, dlatego można przyjąć, że błędy są w zakresie do zaakceptowania.
Widać także, że funkcja nie maleje tak drastycznie od pewnego momentu, tylko bardzo powoli. Przyczyną tego zdarzenia jest prawdopodobnie to, że jak serwer zapełnimy już żądaniami do jego limitu, to po obsłużeniu ich może przyjmować nowe połączenia.