Web server scaling simulation

Robert Kraut

SPIS TREŚCI

Wprowadzenie

Użytkowników internetu przybywa coraz więcej.

Aplikacje i stojące za nimi serwery pod wpływem dużej ilości zapytań dochodzą do ich limitu.

Do niedawna standardem były blokujące serwery typu Apache Tomcat, które dla dla każdego requesta tworzyły nowy wątek, który czeka aż operacja dobiegnie końca. Jednakże w momencie, kiedy wątek czeka na np. jakąś operację IO, wątek jest zawieszony i nie można z niego korzystać. Jest to problematyczne, gdyż pula wątków w serwerze jest ograniczona i przy pewnym bardzo dużym obciążeniu może się zdarzyć, że użytkownik będzie czekał na obsłużenie swojego żądania, do tego dochodzą jeszcze problemy związane z zamianą kontekstu Z pomocą przychodzi Non-Blocking IO, popularnie teraz określane buzzwordem Reaktywne.

Czym jest Non-Blocking IO i z czym się wiąże?

Operacje Non-Blocking to takie, które nie blokują wątku. Są podobne do asynchronicznych, czyli od razu dostajemy odpowiedź, taką na jaką w danej chwili możemy otrzymać, ale oczekuje się od nas, żeby później operację powtórzyć. Jako iż samo w sobie jest to bezużyteczne, dochodzi pojęcie pętli zdarzeniowej (event loop).

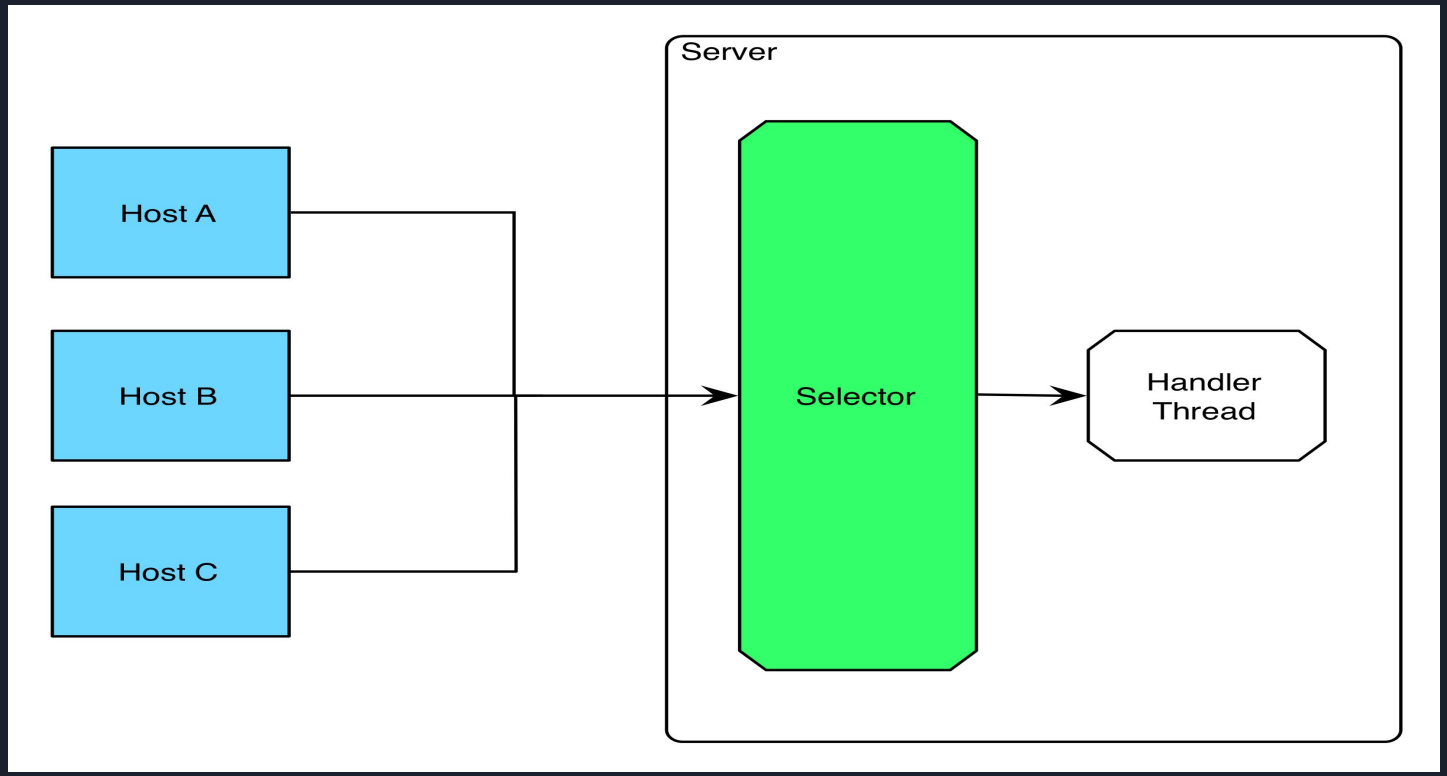
Pętla zdarzeniowa zdarzeniowa jest pętla, która sprawdza czy są jakieś nowe zdarzenia, np. nowe dane na sockecie i potem deleguje je do właściwego handlera. Pętla zdarzeniowa biegnie zawsze na jednym wątku.

W kontekście serwerów http te zdarzenia oznaczają możliwość obsłużenia IO przez sockety. W świecie javy takim mechanizmem jest Selector z Netty, który jeśli socket jest gotowy do obsłużenia operacji IO, jest wybierany i wykonany na nim zostaje read lub write.

Zastosowanie

Non Blocking IO ma zastosowanie w serwerach HTTP, jak np. te, które opierają się na Netty, który bazuje też na pętlach zdarzeniowych i Selektorach z javy.nio. Służą one do budowania wysoce skalowalnych aplikacji, używając jedynie kilka wątków(pętli).

Ilustracja



Cel symulacji

Celem symulacji jest sprawdzenie o ile więcej żądań jest w stanie obsłużyć reaktywny serwer w porównaniu do zwykłego blokującego.

Symulacja powinna dać nam wgląd dlaczego te podejście skaluje się lepiej niż blokujące.

W moim przypadku wybiorę Netty jak non-blocking oraz Tomcat jako blocking

System

Naszym systemem będzie architektura klient serwer.

Encjami, które udało mi się wyróżnić to użytkownik, prosta aplikacja oraz serwer.

Po wielu godzinach wnikliwej analizy literatury uznałem, że najlepiej odzwierciedlać będzie nasz system model Queueing Network.

Model Queueing Networks

Queueing Networks to model, w którym system jest reprezentowany jak sieć kolejek rozwiązywanych **analitycznie.** Rozwiązanie analitycznie opiera się na softwarze do rozwiązania równań narzuconych przez sieć.
Użytkownicy(requesty) przychodzą do **serwisu**(serwer) i są obsługiwani. Jeśli serwis jest zajęty trafiają do kolejki.
Ważnymi parametrami tego modelu są liczba użytkowników, liczba serwisów(w naszym przypadku 1 serwer). Innymi parametrami to także: częstotliwość przychodzenia użytkowników(requestów) oraz czas obsłużenia go, czyli po prostu wykonanie żądania przez aplikację.

Jest to dokładnie to czego szukamy, przychodzą do serwera requesty, przy odpowiedniej dużej ilości będzie je musiał kolejkować i chcemy wiedzieć, które podejście da radę dłużej bez kolejkowania.

Ilustracja naszego modelu dla 1 serwisu(serwera)

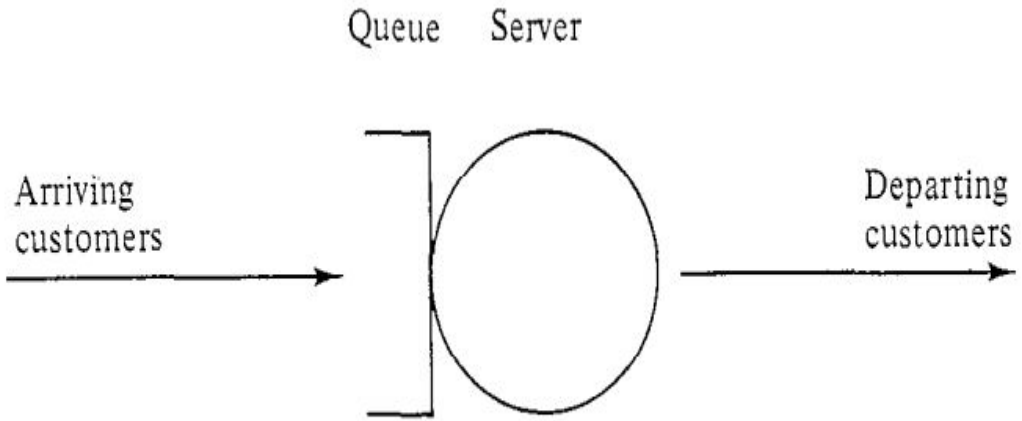


Figure 1.1 – A Single Service Center

Rozwiązanie problemu

Dla pewnych parametrów model ten można przybliżyć równaniem biorące pod uwagę:

- czas jaki serwer jest zajęty
- czas spędzony na odbieranie i wysyłanie requestów
- rozmiar kolejki(średnia ilość requestów w kolejce)
- wydajność(częstotliwość przechodzenia requestów przez nasz serwer)

Plan działania

1. Zebranie danych o użytych serwerach
2. Opracowanie prostej aplikacji służącej do obsługi requestów
3. Opracowanie modelu i weryfikacja
4. Implementacja modelu
5. Stworzenie aplikacji do wizualizacji symulacji
6. Symulacja dla obu serwerów

Pytania

1. Czy wybrany temat jest sensowny i realizowalny w naszym pozostałym czasie?
2. Czy model wybrany przeze mnie jest odpowiedni?
3. W jakim kierunku mam iść, czym zająć się dalej?

Literatura

- https://homes.cs.washington.edu/~lazowska/qsp/Images/Chap_01.pdf
- https://www.researchgate.net/publication/329917229_Network_Traffic_Modeling_Case_Study_The_University_of_Jordan
- https://trak.kia.prz.edu.pl/uploads/research/document/752c00abeb466a71059b553b8a3000e94e6f5097.pdf
- https://www.researchgate.net/publication/282076891_Computer_Networks_Performance_Modeling_and_Simulation/link/59dc80020f7e9b1460037a4a/download
- https://www.researchgate.net/publication/221445897_Prefetching_Inlines_to_Improve_Web_Server_Latency#pf5
- http://bluebox.ippt.pan.pl/~bulletin/(56-4)379.pdf

Dziękuję za uwagę!