Ze świata USOS. Część 10 – Ciągnąć czy pchać?

Maxymilian ŚMIECH student, Instytut Informatyki, Wydział Matematyki, Informatyki i Mechaniki, Uniwersytet Warszawski

W okolicy mojego wydziału znajduje się stylowy bar mleczny. Potrawę należy wybrać przy kasie, zapłacić, a następnie podejść do „okienka". Tam podajemy paragon i rozpoczynamy oczekiwanie na posiłek.

Niestety, okienko jest wyjątkowo małe i znajduje się bardzo nisko. Ciężko jest wsunąć rękę, a próba spojrzenia do wnętrza kuchni wymaga schylania się. Zatem skąd mam wiedzieć, czy po drugiej stronie stoi mój talerz? Przy okienku nie ma dość miejsca na ciągłe czekanie, a zachowanie wygiętej postawy jest męczące – po prostu nie mogę cały czas tam zaglądać.

Spróbujmy opisać zachowania różnych klientów tego baru. Jednocześnie porównajmy przedstawioną historię ze światem komputerów na przykładzie najprostszej i niezwykle popularnej analogii, będącej w naszym świecie od lat: serwera WWW. Kuchnia to taki serwer. Produkty spożywcze to dane, a kucharki są jak programy (procesy, wątki) obsługujące żądania ludzi-klientów. Wszyscy są głodni, nikt nie chce dostać zimnego jedzenia.

Człowiek, który wciąż zagląda, by sprawdzić „czy już", stara się skrócić okres stygnięcia. Im częściej będzie patrzył, tym krócej jego posiłek będzie czekał na odebranie. Niestety, każde pytanie zabiera trochę czasu i energii. Należy wypracować kompromis między naszym zaangażowaniem a świeżością potrawy. Jeśli rzadziej pytamy, możemy w przerwach wykonywać inne czynności – czytać gazetę lub rozmawiać z kolegą.

Najbardziej rozpowszechnionym przykładem takiego zachowania jest indeksowanie sieci. W tym przypadku klient-wyszukiwarka prosi o stronę WWW, po czym analizuje jej treść. Jeśli spodziewa się regularnych aktualizacji, może częściej odpytywać serwer – łatwo zauważyć, że popularne portale informacyjne są sprawdzane co kilka minut.

Niestety, nawet jeśli inteligentnie dostosujemy częstotliwość sprawdzania, zawsze pojawią się opóźnienia. To staje się dotkliwe, gdy wiemy, że na stronie wkrótce coś się zmieni, ale nie możemy sobie pozwolić na intensywne pytanie. Taka sytuacja pojawia się na portalach społecznościowych – pojedyncze osoby raczej rzadko publikują, ale nowe zdarzenie od któregokolwiek z naszych licznych znajomych pojawia się dość często. Chcemy być na bieżąco, żeby inni nie posądzili nas o towarzyskie wycofanie. Tutaj nieco żartuję, ale na giełdzie podobna gra toczy się o duże pieniądze.

Wróćmy do naszego baru. Możemy sobie wyobrazić tłok, jaki powstanie przy okienku, gdy wielu klientów będzie bardzo często sprawdzać „czy już". Jeśli będzie ich za dużo, wszyscy się zablokują i nikt nowy nie będzie mógł podejść. Gdy zaczną napierać jeszcze mocniej – ściana nie wytrzyma, a kuchnia obróci się w ruinę. Tak samo w Internecie spotykamy serwisy, które nie nadążają z obsługą klientów. Z reguły wracają do działania, gdy zainteresowanie maleje. Jednak gwałtowny wzrost ruchu może spowodować fizyczną awarię sprzętu, co wstrzymuje działanie usługi na dłuższy czas.

Odpytywanie będziemy ładnie nazywać „ciągnięciem" (*pull*), aby wyrazić działanie klienta: zgłasza się po dane wtedy, gdy ich potrzebuje. Zobaczyliśmy już, że klient powinien uważać ze zbyt częstym odpytywaniem, gdyż to męcząca czynność, także dla serwera.

Zastanówmy się teraz, jak nasz bar działa w rzeczywistości. Klienci nie muszą wciąż zaglądać do okienka – mogą spokojnie siedzieć przy stole. Gdy nadejdzie ich pora, zostaną zawiadomieni głośnym wywołaniem nazwy dania. W tej sytuacji serwer-kuchnia „wypycha" (*push*) informację z własnej inicjatywy – wie najlepiej, kiedy skończy gotować i do tego czasu nie musi nas zalewać bezużytecznymi komunikatami „proszę czekać".

Potrzeba wypychania bieżących informacji pojawiła się wraz z serwisami udostępniającymi bogate w treść i dynamicznie zmieniające się strony WWW. Pierwsze próby były naiwne: aby użytkownik nie musiał nerwowo odświeżać strony, wprowadzono możliwość jej automatycznego przeładowania po ustalonym czasie. Mimo zwolnienia użytkownika z tej czynności, takie rozwiązanie to wciąż zwykłe ciągnięcie.

Dynamiczne strony zaczęły tworzyć nadmierny ruch w sieci. Aby uniknąć przeciążenia serwera, który przy każdym żądaniu musiał na nowo generować całą zawartość, zastosowano pobieranie tylko tych fragmentów, które mogły się zmienić – w przypadku śledzenia znajomych będą to ich nowe komentarze czy zdjęcia. Reszta strony pozostanie ta sama – serwer oszczędzi czas i pamięć na generowaniu statycznego szkieletu, a wszyscy po drodze ucieszą się z mniejszego użycia łącza. Pomocny jest tu zestaw technologii AJAX, pozwalający pobrać dane z serwera i umieścić na wyświetlonej stronie bez ładowania jej od nowa. Niestety, to wciąż nic więcej, jak ciągnięcie.

W tym momencie ograniczeniem stały się tradycyjne serwery WWW, takie jak Apache. Ich model przetwarzania – wczytaj żądanie, wykonaj operację, przekaż wynik – nie pasuje do schematu, w którym serwer niejako „sam z siebie" decyduje o wysłaniu czegoś do przeglądarki. Dążenie do dwukierunkowej komunikacji zmieniło sposób działania serwerów oraz wykorzystywane metody programowania. Od teraz żądanie jednego klienta może wywołać kaskadę zdarzeń, które doprowadzą do wysłania komunikatu do kogoś zupełnie innego. Udało się rozwiązać problem serwerów, lecz na drodze stanęły ograniczenia kanału transmisyjnego, a dokładniej: sposobu wymiany informacji między przeglądarką a serwerem WWW – znanego wszystkim jako protokół HTTP.

Fundamentalne cechy HTTP powodują, że pasuje on niemal wyłącznie do modelu żądanie-odpowiedź. Po pierwsze: zawsze klient nawiązuje połączenie z serwerem. Skomplikowana architektura Internetu i połączeń między komputerami sprawia, że w olbrzymiej większości przypadków przeglądarka jest ukryta przed światem, zatem nie może być wywołana przez serwer. Tylko, gdy z własnej inicjatywy nawiąże z nim połączenie, można przesłać do niej dane.

Programiści zaczęli się zastanawiać, jak można nadużyć tradycyjnego modelu żądanie-odpowiedź, aby tworzyć serwery WWW korzystające z HTTP i potrafiące wypychać informacje. Podstawy są banalne: przeglądarka inicjuje połączenie. Wiemy już, że inaczej się nie da. Mamy natomiast kontrolę nad tym, kiedy takie połączenie rozpoczynać i jak długo je utrzymywać.

Pierwsze rozwiązanie to cykliczne odpytywanie (*polling*) – klient pyta serwer, czy są jakieś dane, jeśli tak, to je pobiera, a na koniec rozłącza się. Po chwili łączy się ponownie. Taki schemat bardzo łatwo zrealizować, mając do dyspozycji tradycyjny serwer WWW. Pewnym wariantem, który wymaga specjalnych rozwiązań po stronie serwera, jest długie odpytywanie (*long polling*). Różnica w stosunku do zwykłego odpytywania jest subtelna: jeśli serwer stwierdzi, że nie ma nic do wysłania, wstrzymuje odpowiedź. Nie robi nic, nawet nie kończy połączenia. Z punktu widzenia klienta, serwer zachowuje się tak, jakby bardzo długo liczył. W rzeczywistości czeka, aż pojawią się interesujące dane do przekazania. Dopiero wtedy wysyła treść i się rozłącza.

Kiedyś zakładano, że serwer szybko zacznie wysyłać odpowiedź. Jeśli przez dłuższy czas nic nie wysyłał, to najprawdopodobniej coś poszło nie tak i należało zerwać połączenie. Natomiast gdy serwer rozpoczął wysyłanie, połączenie mogło trwać bardzo długo, gdyż Internet nie był tak szybki jak dzisiaj. Ważne, żeby co jakiś czas pojawiały się nowe bajty do odebrania. Wysyłanie drobnych danych podtrzymujących połączenie (tzw. „bicie serca") można zrealizować na dowolnym serwerze za pomocą odpowiedniej konfiguracji.

To tylko przykłady kilku nadużyć, które wymyślili sprytni programiści. W pewnym momencie zaczęto zastanawiać się nad standardowym protokołem dwukierunkowej komunikacji na stronach WWW. Tak powstał WebSocket. Wiemy, że HTTP zbudowany jest na bazie protokołu TCP. Mimo że TCP pozwala stworzyć trwałe połączenie, żądania HTTP są niezależne i każde z nich musi zawierać zestaw informacji identyfikacyjnych, co niepotrzebnie wykorzystuje pasmo. HTTP nadaje się do przesyłania pełnych stron WWW, ale jest zbyt kosztowny w przypadku drobnych komunikatów. WebSocket opakowuje HTTP w zaskakujący sposób: tworzy naprawdę trwałe połączenie z serwerem, co pozwala uniknąć przesyłania zbytecznych danych w czasie dalszej transmisji. W ten sposób HTTP zostaje odchudzony do niemal surowego TCP.

Jakie serwisy internetowe są odpowiednikami ruchliwych barów? Mówiliśmy już o portalach społecznościowych. Inne przykłady to podawanie bieżących statystyk finansowych czy dynamiczne gry w przeglądarce. Nawet USOS, a dokładniej niektóre jego aplikacje webowe, zwłaszcza rejestracje żetonowe, ale także USOSweb, w czasie zapisów na przedmioty bądź egzaminy odczuwają wyjątkowe zainteresowanie. Studenci bardzo chcą „ugryźć" miejsce w dobrej grupie.

Dotychczasowa obsługa jest mało fachowa: o ustalonej godzinie rejestracja zostaje otwarta, a setki studentów starają się jak najszybciej przeładować stronę, a następnie kliknąć „zielony koszyk". Najczęściej wszystkie miejsca w najlepszych grupach zostają błyskawicznie zajęte. Niektórzy od razu po wejściu na stronę mają do wyboru tylko te grupy, których nikt nie chciał. Można sobie wyobrazić strategie, jakie stosują studenci, żeby złapać dobrą grupę, a przynajmniej uniknąć trafienia do najgorszej. Krążą różne legendy i tajne sposoby na pomyślną rejestrację. W rzeczywistości decyduje przypadek. Krytycznym punktem systemu jest centralna baza danych, która musi za każdym razem zdecydować o przyjęciu zapisu. W zależności od liczby studentów, serwer staje się niedostępny na minutę lub godzinę. Niektórym udaje się zobaczyć stronę, pozostali otrzymują błąd. To niezbyt sprawiedliwe.

Jako programiści USOS staramy się wykorzystać nowoczesne techniki programistyczne, aby uniknąć „zabijania" serwera w czasie rejestracji. Po pierwsze, odchodzimy od natychmiastowego rejestrowania chętnych studentów – chcemy zbierać ich kliknięcia przez dłuższą chwilę jako „prośby". W ten sposób odciążymy główną bazę danych, aby nie doprowadzić do zablokowania całego systemu. Gdy nadejdzie odpowiednia pora, prośby będą przetwarzane tak, aby dać uczciwą szansę każdemu studentowi.

Prawdziwe pchanie osiągniemy, tworząc specjalny interfejs w przeglądarce: stronę, na którą student wchodzi raz, na początku rejestracji, po czym wszystkie najważniejsze informacje i wydarzenia zostaną mu podane we właściwym momencie. Główny widok zawiera podsumowanie próśb składanych przez wszystkich uczestników. Najważniejsza cecha komunikacji to wysyłanie statystyk do przeglądarki tylko wtedy, gdy coś się zmieni. Ponadto, możemy inteligentnie dobierać częstotliwość powiadamiania w zależności od obciążenia. Dzięki temu dajemy równe szanse wszystkim studentom. Każdy z nich zobaczy działającą stronę, każdy będzie otrzymywał taką samą ilość informacji oraz każdy zdąży podjąć odpowiednie decyzje.

Dwukierunkowa komunikacja pozwala zatrzeć różnice między przeglądarką a serwerem: obie strony stają się niemal równoprawnymi częściami nowoczesnej aplikacji webowej. Wkrótce premiera naszych nowych rozwiązań. Być może Czytelnik, rejestrując się na swój wymarzony przedmiot czy zajęcia z WF-u, wspomni ten artykuł, doceniając zaawansowaną technologię stojącą za pozornie prostą stroną WWW.