AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI, INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ

KATEDRA INFORMATYKI STOSOWANEJ

Praca dyplomowa inżynierska

dHTTP - Rozproszony system wsparcia serwerów sieci web

dHTTP - Distributed companion for central-server based web

Autor: *Dominik Adamiak*
Kierunek studiów: *Informatyka*
Opiekun pracy: *dr inż. Piotr Matyasik*

Kraków, 2018

Uprzedzony o odpowiedzialności karnej na podstawie art. 115 ust. 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.): „Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystycznego wykonania albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, wideogram lub nadanie.", a także uprzedzony o odpowiedzialności dyscyplinarnej na podstawie art. 211 ust. 1 ustawy z dnia 27 lipca 2005 r. Prawo o szkolnictwie wyższym (t.j. Dz. U. z 2012 r. poz. 572, z późn. zm.): „Za naruszenie przepisów obowiązujących w uczelni oraz za czyny uchybiające godności studenta student ponosi odpowiedzialność dyscyplinarną przed komisją dyscyplinarną albo przed sądem koleżeńskim samorządu studenckiego, zwanym dalej «sądem koleżeńskim».", oświadczam, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i że nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.

Serdecznie dziękuję promotorowi za wolność działania, a rodzinie i przyjaciołom za wiarę w to, że się uda.

Spis treści

1. Wprowadzenie

Dzisiejsza informatyka jak nigdy stoi przed wyzwaniami związanymi z wydajnością, wymuszanymi poprzez miliony klientów z szerokopasmowym dostępem do sieci. Ich rozwiązywanie nie jest już możliwe przez wzmacnianie podzespołów pojedyńczych komputerów – wymaga odpowiedniego sposobu projektowania, który skupia się na możliwościach maksymalnego rozłożenia i łatwego skalowania obciążenia. Ta praca proponuje rozwiązanie, które wpisuje się w ten motyw i w przystępny klientom sposób może znacznie odciążyć rozwijające się części sieci web.

1.1. Cele pracy

Celem tej pracy jest stworzenie protokołu i oprogramowania pozwalającego na rozproszony i bezpieczny dostęp do współdzielonych zasobów, w warstwie użytkowej odpowiadającego obecnemu Hypertext Transfer Protocol (HTTP). W toku pracy, system określany będzie skrótem **dHTTP** – distributed HTTP.

Głównym założeniem projektu jest odciążenie stron, które nagle stają się popularne – często opcją dla właścicieli takich stron jest inwestycja w szybsze łącza czy więcej sprzętu, wiążąca się ze sporymi kosztami. Kosztami które mogą zresztą nie znaleźć uzasadnienia – dodatkowy sprzęt pomoże w chwilach szczytowego ruchu, będzie jednak przez większość czasu leżeć odłogiem, wciąż jednak generując koszty łącz i prądu.

Częściowym rozwiązaniem tego problemu są chmury i udostępniany przez nie *autoscaling* – użytkownicy, zamiast utrzymywać własną infrastrukturę, mogą nie tylko zdać się na serwery zarządzane przez zewnętrzny podmiot, ale także wykorzystać mechanizmy automatycznego rozszerzania i zmniejszania ilości urządzeń czy wykorzystywanego łącza, w zależności od obciążenia całego klastra. To pozwala na drastyczną minimalizację kosztów i jest rozwiązaniem coraz chętniej stosowanym także przez podmioty o gigantycznym ruchu sieciowym, jak serwis streamingowy Netflix ([1]).

Wciąż jednak są to rozwiązania obciążające właściciela serwera, co może stanowić problem w przypadku projektów hobbystycznych czy krajów rozwijających się – koszt autoscalingu może przecież okazać się nieproporcjonalnie wysoki w stosunku do czasów spokojnego ruchu.

System zaprezentowany w poniższej pracy – dHTTP – ma posłużyć jako samoskalująca się, rozproszona alternatywa do tego podejścia. Musi wziąć on pod uwagę optymalne rozłożenie danych, zabezpieczenie przed zmienianiem zawartości zasobów przez niepowołane podmioty

(hashe i podpisy kryptograficzne), szybki i możliwie najmniej scentralizowany sposób współdzielenia informacji o stanie systemu (*kto jest online i może dać mi plik* `logo.png` *z hosta* `example.com`*?*) i aktualizacjach zawartości oraz stronę kliencką, będącą w tym przypadku wtyczką do przeglądarki Google Chrome, nakładającą warstwę dHTTP na polecenia pobierania zasobów. Z punktu widzenia użytkownika system jest *przezroczysty*: zostanie węzłem i klientem sieci dHTTP będzie polegało wyłącznie na instalacji wtyczki, działającej autonomicznie w tle. Oprócz aplikacji klienckiej, dHTTP udostępniony zostanie także w trybie *headless* – węzeł uruchamiający może zostać częścią klastra bez użycia przeglądarki.

1.2. Zawartość

Rozdział 1. stanowi krótkie wprowadzenie i definicję celu pracy.

Rozdział 2. zawiera rozważania na temat historycznych i teoretycznych podstaw systemów rozproszonych, oraz przykłady najpopularniejszych narzędzi je implementujących. Wprowadza także do teoretycznych i narzędziowych podstaw implementacji systemu dHTTP.

Rozdział 3. jest szczegółowym opisem wymagań, projektu, modułów, a także implementacji samego systemu, w szczegółach omawiającym funkcjonalności, biblioteki i narzędzia pozwalające na działanie aplikacji, napotkane w toku projektowania problemy i ich rozwiązania, rys architektoniczny sposobu przechowywania danych czy ich propagacji, w rozróżnieniu podejścia węzłowego i podejścia klienta końcowego.

Rozdział 4. to próba walidacji systemu dHTTP – analiza użytkowości systemu pod kątem wrażeń użytkowych, zbiór badań nad wydajnością systemu (z uwzględnieniem różnych mechanizmów propagacji danych) i propozycji na jej dalsze optymalizacje.

Rozdział 5. zawiera krótkie podsumowanie, podkreślające osiągnięte cele i potencjalne kierunki dalszego rozwoju projektu.

2. Droga do rozproszenia

Osławione prawo Moore'a ([2]), wspominające o podwajaniu ilości tranzystorów w procesorach, często parafrazowane jako podwajanie ich mocy obliczeniowej, przestaje działać, podczas gdy złożoność problemów i ilość użytkowników szerokopasmowego internetu rośnie. Próba skalowania wertykalnego – polegającego na wzmacnianiu pojedynczych węzłów, przestaje zdawać próbę czasu.

Problem posiada także drugą stronę – nie wszyscy użytkownicy internetu mogą pozwolić sobie na łącze szerokopasmowe. Szacuje się, że w roku 2017 dostęp do internetu posiada ponad połowa populacji świata; sam wzrost ilości użytkowników sieci Web z Afryki od roku 2000 do 2017 wynosi aż 8503.1% ([3]) – znaczną część tych połączeń stanowią jednak połączenia starych generacji sieci komórkowej, o znacznie ograniczonej przepustowości i ogromnych latencjach. Każdy kolejny węzeł niezbędny do połączenia użytkownika z serwerem końcowym może dokładać cennych milisekund czasu odpowiedzi.

Autorzy IPFS zauważają w swojej pracy ([4]) także inne słabości obecnego internetu – sieć oparta o HTTP jest w prawdzie zdecentralizowana, jako iż treści rozdzielane są pomiędzy miliony węzłów, od gigantycznej sieci Amazon Web Services aż po mikroserwery stojące w domach pasjonatów; brakuje jej jednak faktycznego rozproszenia: ta infrastruktura nie jest gotowa „by design" na przyjmowanie gigantycznego ruchu, nie jest w stanie efektywnie przechowywać i udostępniać wielkich zestawów danych; jest również podatna na „znikanie" danych, jako iż awaria pojedynczego dysku twardego może zatrzymać udostępnianie całej witryny.

2.1. Zarys historyczny

W świecie informatyki szybko wyewoluowały rozwiązania określane jako współbieżne. Istnienie wielu wątków – niezależnych od siebie logicznie toków wywołań, operujących na wspólnej pamięci – i ich przeplot rozwiązywał takie problemy, jak nierówny rozmiar żądań – działający sekwencyjnie serwer blokowałby się przy poleceniach zajmujących więcej czasu. Dzięki przeplataniu wątków można uniknąć tej sytuacji, a także pozwolić na iluzję symultaniczności – mimo korzystania z jednego procesora, polecenia wykonywane są *pseudorównolegle*, co pozwala między innymi na responsywność interfejsów użytkownika.

Upowszechnienie się komputerów wieloprocesorowych i procesorów wielordzeniowych zaowocowało rozwojem obliczeń równoległych. Istotne jest rozgraniczenie tych dwóch pojęć – współbieżność jest raczej paradygmatem, sposobem strukturyzacji oprogramowania w sposób,

który pozwala na wykonywanie wielu poleceń niezależnie i jednocześnie; równoległość z kolei to możliwość uruchamiania tego typu oprogramowania w tym samym czasie, dzięki mnogiej liczbie procesorów ([5]).

i zdecentralizowane, w niedługim czasie stanowiące podstawę dla rozwiązań rozproszonych.

2.2. Historia i istniejące narzędzia

2.3. Podstawa projektu

‘’

3. Projektowanie i implementacja

3.1. Wymagania funkcjonalne

3.2. Definicje, architektura i technologie

3.2.1. Wykorzystane technologie i narzędzia

3.2.2. Problemy warstwy sieciowej

3.2.3. Decentralizacja czy rozproszenie?

3.2.4. Bezpieczeństwo

3.2.4.1. Czy autentykacja to nasza brożka?

3.2.4.2. Gdzie leżą granice zdrowych heurystyk?

3.2.5. Propagacja i przechowywanie danych

3.3. Aplikacje

3.3.1. Węzeł

3.3.2. Klient

4. Testy i optymalizacja

4.1. Zarys użytkowości i działania systemu

4.2. Wydajność systemu

4.2.1. Wydajność klienta

4.2.2. Wydajność serwera

4.2.2.1. Wydajność rozwiązań propagacji danych

4.2.2.2. Wydajność protokołów komunikacji między węzłami

4.3. Podsumowanie rozwiązań optymalizacyjnych

5. Podsumowanie

5.1. Potencjalne kierunki rozwoju

Bibliografia

[1] Amazon Web Services Inc. *Auto Scaling*. 2017. URL: *http://web.archive.org/web/20171127225615/https://aws.amazon.com/autoscaling/* (term. wiz. 2017-12-31).

[2] Gordon E Moore. „Cramming more components onto integrated circuits". W: *Proceedings of the IEEE* 86.1 (1998).

[3] Miniwatts Marketing Group. *World Internet Users and 2017 Population Stats*. 2017. URL: *http://web.archive.org/web/20171226094500/http://www.internetworldstats.com/stats.htm* (term. wiz. 2017-12-31).

[4] Juan Benet. „IPFS - Content Addressed, Versioned, P2P File System". W: *CoRR* abs/1407.3561 (2014). arXiv: *1407.3561*.

[5] Rob Pike. „Concurrency is not Parallelism". W: 2012.