Uniwersytet Mikołaja Kopernika
Wydział Fizyki, Astronomii i Informatyki Stosowanej

Maciej Janczyk
nr albumu: 259014
Praca inżynierska
na kierunku informatyka stosowana

Usługi sieciowe w kontenerach systemu operacyjnego CoreOS

Opiekun pracy dyplomowej
dr hab. Jacek Kobus
Instytut Fizyki

Toruń 2016

Pracę przyjmuję i akceptuję

.....
data i podpis opiekuna pracy

Potwierdzam złożenie pracy dyplomowej

.....
data i podpis pracownika dziekanatu

Dziękuję promotorowi za poświęcony czas i udzieloną pomoc.

Spis treści

Rozdział 1

Wstęp

Od wielu lat administratorzy serwerów używają technik wirtualizacyjnych w celu efektywnego zarządzania usługami sieciowymi. Zwykle przeznacza się jedną oddzielną maszynę wirtualną, by dostarczała odpowiedniego środowiska dla funkcjonowania pojedynczej aplikacji. Rozwiązanie takie pozwala na skuteczne odizolowanie od siebie działających usług, ułatwia zarządzanie nimi, w tym planowanie i wdrażanie skutecznej polityki bezpieczeństwa. Jednak realizacja tego podejścia na większą skalę, np. w zastosowaniach chmurowych, czy dużych sieciach, wymaga użycia potężnych serwerów, gdyż każda maszyna wirtualna musi otrzymać swój przydział pamięci dyskowej, pamięci operacyjnej oraz czasu procesora. Ten problem można w znacznej mierze ograniczyć, jeżeli wybierzemy sposób wirtualizacji na poziomie aplikacji. Technologia ta znana jest już od przeszło 40 lat i była pierwotnie wykorzystywana przez twórców gier komputerowych w celu przystosowania do działania na różnych architekturach sprzętowych, m.in. firma *Infocom* opracowała *Z-machine* [1]. Innym przykładem wykorzystania tego typu wirtualizacji może być znany użytkownikom systemu Linux program Wine, który umożliwia uruchamianie w ramach systemu linuksowego programów przeznaczonych dla systemu Windows [2]. W roku 2008 zespół programistów z firm *Parallel*, *IBM* i *Google* powołał do życia projekt *Linux Containers*, który zakładał, że wirtualizacja pojedynczej aplikacji będzie realizowana poprzez umieszczenie jej w specjalnym izolowanym środowisku (*sandbox*) [3].

Tego typu podejście było wykorzystywane już wcześniej. Firma *Sun Microsystems* przedstawiła w 2004 roku *strefy* dla systemu Solaris, które umożliwiały wirtualizowanie zasobów systemu operacyjnego i uruchamianie programów w izolowanych i bezpiecznych środowiskach [4]. Rok później pojawił się system FreeBSD w wersji 4 zawierający mechanizm *Jails*, który pozwala na podział systemu na mini-instancje nazywane *celami* [5]. Innym ważnym przykładem wirtualizacji na tym poziomie jest znane użytkownikom systemów linuksowych środowisko `chroot` [6]. Polecenie te daje możliwość uruchomienia procesu izolując go od najważniejszych plików systemowych.

Osadzanie aplikacji w klatkach przynosi te same korzyści co używanie maszyn wirtualnych. Klatki jednak nie uruchamiają własnego jądra linuksowego i nie potrzebują do działania hipernadzorcy, więc ich użycie prowadzi do wydatnego ograniczenia zużycia zasobów sprzętowych. Pojedyncza klatka LXC oprócz samej usługi musi zawierać wszelkie pakiety potrzebne do jej poprawnego działania. Dzięki takiemu podejściu każdy program działa we właściwym dla niego środowisku, co nie zawsze jest łatwe do zagwarantowania w typowym podejściu, kiedy wszystkie procesy mają dostęp do wspólnych zasobów.

Klatki LXC są wspierane przez wiele systemów operacyjnych, m.in. przez opracowany w 2013 roku – i stale intensywnie rozwijany – system operacyjny CoreOS. W ramach tego podejścia do zarządzania klatkami używa się platformy `Docker`, która umożliwia tworzenie i zarządzanie klatkami, a także uruchamianie osadzonych w nich aplikacji. `Docker` jako zarządca kontenerów umożliwia pełną przenaszalność klatek pomiędzy różnymi systemami operacyjnymi, które wspierają ten system [7].

Celem niniejszej pracy inżynierskiej była szczegółowa analiza działania systemu CoreOS i jego najważniejszych składowych: podsystemów `Docker`, `etcd` i `fleet`. W tym systemie procesy działające w klatkach mają dostęp do wspólnych zasobów przechowywanych w magazynach danych typu *klucz-wartość* (dzięki odpowiedniemu interfejsowi programista ma dostęp do tych danych w formacie JSON). Dzięki podsystemowi `fleet` można uruchamiać kontenery odpowiedzialne za poszczególne usługi przy wykorzystaniu demona `systemd`. Do głównych zadań podsystemu `fleet` należą: wybór węzła klastra, na którym będzie uruchomiona usługa oraz ewentualne przeniesienie serwisu z węzła, na którym był pierwotnie uruchomiany (a który uległ awarii), na inny, sprawny węzeł.

Analiza systemu CoreOS wymagała zbudowania małej, testowej infrastruktury chmurowej dostarczającej wybrane usługi sieciowe. W tym celu zainstalowano system CoreOS na trzech maszynach i połączono je w taki sposób, aby utworzyły oddzielny klaster o minimalnym rozmiarze. Dwa takie klastry zbudowano wykorzystując różne środowiska wirtualizacyjne, mianowicie na komputerze autora pracy w ramach środowiska VirtualBox oraz na serwerach wydziałowych – środowiska QEMU/KVM. Dzięki temu możliwa stała się szczegółowa analiza procesu instalacji i konfiguracji poszczególnych węzłów klastra, a także działania podsystemów `etcd` oraz `fleet`. Następnie przystąpiono do uruchamiania w ramach klastra kilku podstawowych usług sieciowych. Przygotowano i testowano kontenery wspierające takie usługi jak DHCP, WWW oraz LDAP. Wybór usług był podyktowany ich znaczeniem dla funkcjonowania wydziałowej sieci komputerowej: konfigurowania hostów w lokalnej sieci, dostępu do stron www wydziału oraz zagadnienia uwierzytelniania użytkowników przy dostępie do rozmaitych usług oferowanych przez serwery uczelniane (centralny punkt logowania) i wydziałowe. Uruchomienie tych usług na klastrze wydziałowym wymagało dodatkowych prac związanych z automatycznym przenoszeniem danych wyjściowych przygotowywanych przez centralny serwer zarządzający usługami na poszczególne węzły, aby były one dostępne we właściwych kontenerach. Do chwili składania pracy udało się uruchomić na klastrze wydziałowym usługi DHCP oraz LDAP. Z uwagi na tempo rozwoju systemu CoreOS jego dokumentacja nie zawsze była aktualna i często trzeba było rozwiązywać problemy przez odwoływanie się do informacji zamieszczanych na odpowiednich listach dyskusyjnych.

Układ niniejszej pracy jest następujący. Po wstępie, w rozdziale 2. zostaną omówione najważniejsze elementy składające się na system operacyjny CoreOS. Rozdział 3. zawiera: instrukcję instalacji tego systemu, opis metod umożliwiających przekształcanie oddzielnych węzłów pracujących pod nadzorem systemu CoreOS w klaster oraz sposoby uruchamiania wybranych usług w chmurze. Pracę kończy krótkie podsumowanie wraz ze spisem wykorzystanej w pracy literatury przedmiotu.