Wydział Zastosowań Informatyki i Matematyki
Szkoła Główna Gospodarstwa Wiejskiego w Warszawie

Warszawa, 5.12.2016

INFORMACJA O PROPONOWANEJ DO OTWARCIA ROZPRAWIE DOKTORSKIEJ

Zdecentralizowany system do zarządzania aplikacjami chmurowymi w postaci mikrousług działających wewnątrz kontenerów wirtualizacyjnych

Doktorant: mgr inż. Grzegorz DWORNICKI
Promotor: dr hab. inż. Marian RUSEK

Uzasadnienie podjęcia tematu

Ostatnio coraz większą popularność przy projektowaniu aplikacji uruchamianych w chmurze obliczeniowej uzyskuje architektura mikrousług (*microservices architecture*). Mikrousługi są odizolowanymi od siebie małymi programami, każdy z których realizuje dobrze określoną funkcjonalność. Jedynym sposobem wymiany danych między nimi jest protokół HTTP [1,2]. Aplikacja stworzona w tej architekturze jest skalowalna (przez multiplikację mikrousług danego rodzaju) oraz odporna na awarię (awaria niektórych mikrousług powoduje najwyżej czasowe zubożenie funkcjonalności aplikacji, której rdzeń bez przeszkód działa nadal).

Mikrousługi napisane są w różnych językach programowania i dlatego z powodu różnorakich zależności nie jest możliwe uruchamianie ich bezpośrednio na tym samym serwerze. Z pomocą przychodzą technologie wirtualizacji. Kontenery wirtualizacyjne dzielą między sobą jądro systemu gospodarza i dlatego ich gęstość na serwerze jest 10 razy większa niż gęstość maszyn wirtualnych (każda z których zawiera kompletny system operacyjny) [3]. Dlatego też właśnie kontenery wirtualizacyjne stanowią preferowany sposób uruchamiania mikrousług.

Napisana zgodnie z architekturą mikrousług aplikacja może działać szybciej, jeśli komunikujące się ze sobą często mikrousługi znajdują się na tym samym serwerze. Dlatego też stosowane obecnie systemy zarządzające mikrousługami zaimplementowanymi wewnątrz kontenerów wirtualizacyjnych, takie jak Docker Swarm, Kubernetes, czy też Mesos [4] starają się optymalizować ich rozmieszczenie na serwerach podczas uruchamiania aplikacji. Czasem konieczna jest jednak dynamiczna rekonfiguracja położenia mikrousług (np. w celu wyłączenia nieużywanych serwerów). Koszt przesłania mikrousługi przez sieć (kod + dane) jest oczywiście nieco większy niż koszt przesłania samych danych, ale umożliwia natychmiastowe wznowienie pracy mikrousługi na serwerze docelowym (gdyby przesłać same dane należałoby i tak pobrać wstępny obraz kontenera wirtualizacyjnego z mikrousługą i zainicjować go danymi). Przyśpieszenie oczywiście nie następuje przy komunikacji jednokierunkowej - druga z mikrousług musi odpowiadać pierwszej i najlepiej, gdy dialog między nimi toczy się wielokrotnie.

Obecnie stosowane środowiska uruchomieniowe kontenerów wirtualizacyjnych takie jak rkt czy Docker [5] pozwalają na przeniesienie kontenera na inny serwer poprzez zatrzymanie jego działania, przeniesienie pliku obrazu jego plików na inny serwer i ponowne uruchomienie. Przenoszone są tylko dane zapisane przez mikrousługę na dysk - dane zawarte w jej pamięci RAM są bezpowrotnie tracone. Czas niedziałania usługi jest taki sam, jak czas przenoszenia pliku obrazu. Plik obrazu zawiera w tym przypadku także kod mikrousługi.

Niedawno środowisko uruchomieniowe LXC od wersji 2.0 zaczęło wspierać także migrację kontenerów wirtualizacyjnych na żywo (*live migration*) - metoda ta została pierwotnie opracowana do przenoszenia maszyn wirtualnych między serwerami. Dzięki zastosowaniu optymalizacji w postaci migracji iteracyjnej bądź leniwej (*lazy*) czas niedziałania mikrousługi jest około 10 razy krótszy od czasu przenoszenia pliku obrazu. Dodatkowo dane zawarte w pamięci ulotnej są również przenoszone. Dlatego też zastosowanie migracji na żywo w zarządzaniu aplikacjami chmurowymi działającymi w formie zbioru kontenerów wirtualizacyjnych z mikrousługami powinno przynieść poprawienie ich wydajności.

Istniejące scentralizowane systemy zarządzania mikrousługami opierają się na założeniu, że czas niedziałania kontenera jest taki sam, jak czas jego przenoszenia pomiędzy serwerami. To dlatego baza danych przechowująca informacje o położeniach kontenerów na serwerach może być spójna z faktycznym stanem aplikacji. Założenie to nie jest spełnione podczas zastosowania migracji na żywo. Podczas przenoszenia kontener nadal działa, znajduje się fizycznie jednocześnie w dwóch miejscach i traci kontakt z siecią na czas 10 razy krótszy od czasu przenoszenia. Scentralizowana bazy danych nie może w żaden sposób być stale spójna ze stanem faktycznym. Dlatego też potrzebny jest zdecentralizowany system zarządzający kontenerami.

Dostępne systemy zarządzania mikrousługami posiadają wydzielone centrum i dlatego charakteryzują się ograniczoną skalowalnością i odpornością na awarię. Na przykład w systemie zarządzającym Docker Swarm istnieje grupa wydzielonych węzłów typu manager, które replikują miedzy sobą dane za pomocą algorytmu RAFT. Z powodów wydajnościowych firma Docker Inc. radzi jednak, aby serwerów tych nie było więcej niż 7 (3 z których mogą ulec awarii). Stanowi to ograniczenie dla wielkości chmury - obecnie zademonstrowano działanie systemu Docker Swarm na 1000 serwerach z 50 000 kontenerów.

Cel rozprawy

Pierwszym z celów planowanej rozprawy doktorskiej jest udowodnienie tezy, iż opracowane nowe rozproszone algorytmy zarządzania kontenerami wirtualizacyjnymi w klastrach komputerowych pozwalają na lepszą wydajność, skalowalność i bezpieczeństwo aplikacji chmurowych..

Zastosowane algorytmy zainspirowane są zachowaniem rojów autonomicznych robotów z oddziaływaniem feromonowym. Odpowiednikiem ruchu robota jest migracja kontenera na żywo. Feromony przyciągające lub odpychające kontenery wyliczane będą niezależnie przez każdy serwer na podstawie wymagań i liczby obecnych na nim kontenerów oraz zasobów serwera. Na przykład w najprostszej wersji wymagania kontenera mogą być opisywane przez wektor: ilość pamięci, wielkość przestrzeni dyskowej, liczba rdzeni procesora, obecność bazy danych, obecność określonych typów kontenerów itp., zaś wyliczanie optymalnych liczb kontenerów na serwerze bazowałoby na algorytmie Dominant Resource Fairness. Kontenery nadmiarowe podejmowałyby autonomiczną decyzję o migracji na inny serwer (który po otrzymaniu zapytania o wolne miejsce wyliczałby ponownie optymalne liczby kontenerów różnych typów i sprawdzał, czy ma miejsce dla kolejnego). Kontener podejmujący próbę migracji widzi wynik tych wyliczeń jako feromon przyciągający bądź odpychający.

Proponowany system zarządzający kontenerami wirtualizacyjnymi na serwerach chmury obliczeniowej nie ma centrum zarządzającego i dlatego też jest lepiej skalowalny i odporny na awarię od istniejących systemów. Stworzenie go jest drugim celem rozprawy doktorskiej.

Aby ułatwić praktyczne wdrożenie proponowanych koncepcji jako trzeci i ostatni cel doktoratu przewidziano stworzenie systemu hybrydowego, w którym rój kontenerów będzie działał jako planista drugiego stopnia systemu Mesos. Powoli to na łatwe uruchomienie aplikacji mogących uzyskać z niego korzyści na istniejącej infrastrukturze chmurowej zarządzanej przez ten dojrzały system z wygodną nakładką administracyjną DC/OS. Rola systemu Mesos sprowadzałaby się tu do oferowania miejsca na serwerach kontenerom chcącym się przenieść.

Metodyka badań

- Stworzenie przykładowych aplikacji w architekturze mikrousług i porównanie ich wydajności w klastrach zarządzanych przez zaprojektowany zdecentralizowany system z migracją kontenerów na żywo i istniejące scentralizowane systemy bez tej cechy. Mierzone są przerwy w pracy mikrousług podczas migrowania na żywo zawierających ich kontenerów. Migracja odbywa się na zlecenie systemu zarządzającego w odpowiedzi na zmianę zachowania chmury. Może to być optymalizacja położenia kontenerów na istniejących serwerach, zapełnianie nowych serwerów, czy też ucieczka z serwerów ulegających awarii.
- Symulacje działania stworzonego zdecentralizowanego systemu zarządzającego kontenerami. Wyznaczane są rozkłady czasów dochodzenia układu do stanu równowagi. Analiza ich skalowania dla rosnącego rozmiaru chmury i liczby działających w nich kontenerów pozwoli przewidzieć zachowanie proponowanego rozwiązania w rzeczywistym centrum przetwarzania danych. Parametry symulatora będą dopasowane tak, aby uzyskać maksymalną zgodność z przeaprowadzynmi dla stosunkowo małych liczb serwerów i kontenerów eksperymentami praktycznymi.
- Eksperymenty praktyczne z pustymi kontenerami w rzeczywistym klastrze komputerowym umożliwią zbadanie wpływu skończonej przepustowości sieci i jej topologii na wydajność pracy wybranej implementacji zaprojektowanego systemu zarządzającego oraz przetestowanie jego odporności na symulowane awarie sprzętu. Analiza różnych systemów konteneryzacji i sposobów zlecania migracji przez kontener pozwoli maksymalnie zwiększyć bezpieczeństwo stworzonego systemu.

Zakres rozprawy

1. Opracowanie rozproszonego algorytmu zarządzającego
2. Zaimplementowanie go w kontenerach wirtualizacyjnych pod kątem maksymalnego bezpieczeństwa
3. Budowa przykładowego klastra komputerowego z uwzględnieniem odporności na awarie

4. Eksperymenty praktyczne badające wydajność implementacji algorytmu zarządzającego
5. Porównanie eksperymentów z symulacjami i wyciągnięcie wniosków na temat skalowalności systemu
6. Stworzenie przykładowej aplikacji w architekturze mikrousług
7. Uruchomienie aplikacji w proponowanym systemie i przetestowanie jej wydajności i skalowalności przy symulowanych awariach serwerów klastra i dodawaniu nowych pustych serwerów
8. Przeniesienie aplikacji do któregoś z istniejących systemów i porównanie wyników

Uzyskane wyniki

- Zbadano iteracyjną migrację na żywo procesu aplikacji i tego samego procesu aplikacji zamkniętego w kontenerze wirtualizacyjnym. Stwierdzono, iż w drugim przypadku czas niedostępności aplikacji w sieci był 3 razy dłuższy niż w pierwszym. Migrowanie procesu wymagało jednak dodatkowych zabiegów (nawiązywanie pomocniczego połączenia sieciowego i przepisanie gniazdek), które są zbędne przy zamknięciu procesu w kontenerze (z uwagi na wirtualizację zasobów). Wynik opublikowano w pracy [6].
- Przetestowano rozproszony algorytm zarządzający w scenariuszu równoważenia liczby kontenerów na serwerach chmury. Pokazano, iż w celu ustabilizowania algorytmu należy wprowadzić założenie o szeregowej migracji i przy wyznaczaniu prawdopodobieństwa migracji odejmować liczbę kontenerów czekających w kolejce od liczby kontenerów w systemie plików. Wyniki doświadczalne dla 2 serwerów opublikowano w pracy [7]. Założenie o szeregowym wychodzeniu dodatkowo upraszcza implementację oraz, jak zbadano eksperymentalnie w scenariuszu jeden serwer pełny pozostałe puste - zwiększa czas osiągania równowagi przez system jedynie o 8% w stosunku do wychodzenia równoległego (z uwagi na skończoną przepustowość sieci).
- Przeanalizowano teoretycznie zachowanie algorytmu z losowym wybieraniem hosta docelowego w scenariuszu dodawania pustego serwera do chmury i zajmowania go przez kontenery z aplikacja. Model niezależnych skoków daje wówczas geometryczny rozkład czasów dochodzenia układu do równowagi.Wynik porównano z symulacją uwzględniającą oddziaływanie kontenerów poprzez kolejkowanie (ale zakładającą nieskończoną przepustowość sieci) i rzeczywistym eksperymentem (z powodu skończonej przepustowości sieci wartość średnia rozkładu była o 50% większa niż symulacja).
- Poprawiono osiągi algorytmu modyfikując sposób wyboru serwera docelowego przez kontener. Zrezygnowano z losowania na rzecz wyboru kolejnych serwerów i sprawdzania ich stanu. Ulepszono szczegóły implementacji poprzez dodanie dodatkowych funkcji jądra i drugiej kolejki. Wyniki eksperymentu dla klastra składającego się z 18 serwerów i poprzednio omawianego scenariusza równoważenia obciążenia z początkowym stanem jeden serwer pusty pozostałe pełne, zostały opublikowane w pracy [8].

Literatura

1. Malavalli, Divyanand, and Sivakumar Sathappan. "Scalable microservice based architecture for enabling DMTF profiles." Network and Service Management (CNSM), 11th International Conference on. IEEE, 2015.
2. Balalaie, Armin, Abbas Heydarnoori, and Pooyan Jamshidi. "Migrating to cloud-native architectures using microservices: An experience report." European Conference on Service-Oriented and Cloud Computing. Springer International Publishing, 2015.
3. Pahl, Claus. "Containerisation and the PaaS cloud." IEEE Cloud Computing (2015): 24-31.
4. Hindman, Benjamin, et al. "Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center." NSDI. Vol. 11. 2011.
5. Merkel, Dirk. "Docker: lightweight linux containers for consistent development and deployment." Linux Journal 2014.239 (2014): 2.
6. Dwornicki, Grzegorz, „Migration of processes as a mean to ensure system load balancing and high availability” Proceedings of 8th Conference on Information Systems in Management, WULS Press, Warsaw 2013.
7. Rusek, Marian., Grzegorz Dwornicki, and Arkadiusz Orłowski „Swarm of Mobile Virtualization Containers.” Information Systems Architecture and Technology: Proceedings of 36th In-ternational Conference on Information Systems Architecture and Technology–ISAT 2015–Part III. Springer International Publishing, 2016.
8. Rusek, Marian, Grzegorz Dwornicki, and Arkadiusz Orłowski. "A Decentralized System for Load Balancing of Containerized Microservices in the Cloud." International Conference on Systems Science. Springer International Publishing, 2016.