Kraków, 19.07.2023

dr hab. inż. Bartosz Baliś, prof. ucz.
Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie
Instytut Informatyki
Al. Mickiewicza 30
30-059 Kraków

Recenzja

rozprawy doktorskiej mgra Pawła Żuka

pt. „Resource allocation methods for serverless cloud computing platforms"

1. Problematyka pracy

Autorem przedstawionej mi do recenzji pracy doktorskiej jest mgr Paweł Żuk, a jej promotorem jest dr hab. Krzysztof Rządca, prof. ucz (Uniwersytet Warszawski).

Problematyka pracy dotyczy metod przydziału zasobów w bezserwerowych platformach przetwarzania w chmurze, zwanych również platformami typu FaaS ("funkcja jako usługa"). Wybrana tematyka pracy jest ważna i aktualna. Obserwujemy obecnie intensywny rozwój bezserwerowego modelu obliczeniowego i usług na nim opartych. Usługi takie prawdopodobnie w przyszłości będą dominującym modelem tworzenia aplikacji chmurowych.

W przedstawionej pracy Autor skupia się na trzech problemach badawczych. Pierwszy z nich dotyczy modelu aplikacji bezserwerowych, w którym funkcje są łączone w grafy lub łańcuchy wywołań. Wiedza o strukturze aplikacji może być wykorzystana w algorytmach zarządzania zasobami, np. po to aby z wyprzedzeniem przygotować środowisko wykonawcze dla funkcji i przez to zredukować wystąpienia tzw. zimnych startów. Drugi problem badawczy dotyczy optymalizacji zarządzania zasobami w środowisku FaaS w skali centrum danych. Głównym celem optymalizacji jest redukcja całkowitej liczby maszyn przeznaczonej do przetwarzania zadanego obciążenia bez obniżenia wskaźników jakości tego przetwarzania. Trzecim problemem badawczym jest efektywne zarządzanie zasobami na potrzeby obliczeń bezserwerowych w ramach pojedynczej maszyny. Głównym celem jest szeregowanie wywołań funkcji, tak aby zminimalizować parametry związane z czasem odpowiedzi.

2. Analiza i ocena treści pracy

Przedłożona rozprawa doktorska mgra Pawła Żuka składa się z siedmiu rozdziałów i obejmuje 128 stron wraz z bibliografią, która liczy 141 pozycji.

Rozdział 1 zawiera krótkie wprowadzeniem przestawiające ogólne tło pracy; zwięzły, ale szeroki przegląd przegląd literatury podsumowujący obecny stan badań w dziedzinie obliczeń bezserwerowych; a także przewodnik po strukturze pracy oraz dane bibliograficzne pięciu artykułów naukowych, w których opublikowano wyniki przedstawione w poszczególnych rozdziałach pracy.

Rozdział 2 jest bardzo krótkim wprowadzeniem podstawowych pojęć modelu FaaS oraz platformy Apache OpenWhisk, która jest podstawą zarówno opracowanych w pracy modeli jak i badań eksperymentalnych.

Rozdziały 3-6 stanowią trzon dysertacji, w którym przedstawiono główne wyniki badań. Zasadniczo mają one strukturę artykułów naukowych, np. zawierają przegląd badań pokrewnych związanych ściśle z tematyką danego rozdziału. Warto zaznaczyć, że rozdziały 3-6 obejmują zdecydowaną większość zawartości pracy (s. 17-110).

Przedmiotem rozdziału 3 jest szeregowanie *kompozycji funkcji*, w których funkcje mogą wywoływać kolejne funkcje tworząc łańcuch (gdy wszystkie wywołania są synchroniczne) bądź też graf (gdy przynajmniej część jest asynchroniczna) wywołań. Istniejące systemy nie biorą pod uwagę struktury wywołań w szeregowaniu funkcji. W rozdziale 3 zaproponowano metodę, która wypełnia tę lukę: model szeregowania kompozycji funkcji, który łączy problem plecakowy z wieloma plecakami reprezentującymi dostępne maszyny o określonej pojemności obliczeniowej i pamięciowej, oraz szeregowaniem zadań z zależnościami i czasem przygotowania. Ponieważ problem jest NP-trudny, w dalszej części rozdziału zaproponowane są heurystyki dla trzech aspektów tego problemu. Ewaluacja algorytmów wykorzystuje zbiór danych Microsoft Azure Trace i jest wykonana przy pomocy symulatora, który jest walidowany z wykorzystaniem systemu OpenWhisk.

Wykonana ewaluacja jest obszerna — zawiera 1440 przypadków testowych — i bada wpływ szeregu czynników na jakość szeregowania, porównując różne warianty heurystyk. Mimo tego, że zaproponowanych algorytmów nie zbadano w rzeczywistym systemie, wartość poznawczą wyników oceniam bardzo wysoko.

W rozdziale 4 zaproponowano metodę optymalizacji wykorzystania klastra obliczeniowego, tj. minimalizację liczby maszyn przetwarzających określone obciążenie w systemie FaaS. Zaproponowany został model obciążenia, którego podstawą są *aplikacje* (ich odpowiednikiem w systemie FaaS są funkcje). Przyjęto założenie, że aplikacje charakteryzuje się określonymi (stabilnymi) wymaganiami pamięciowymi i obliczeniowymi (obciążenie CPU). Zaproponowana nowa metoda alokacji zasobów dla aplikacji polega na dzieleniu aplikacji na wiele instancji ulokowanych na różnych maszynach i rozdzielenie obciążenia procesora pomiędzy te instancje. W ten sposób można zmieniać "rozmiar" instancji, zatem rozlokowanie wielu instancji na jak najmniejszej liczbie maszyn sprowadza się do problemu pakowania. Model ten został nazwany modelem "multi-instanced" w odróżnieniu od bazowego modelu "single-instanced". Drugim celem optymalizacji jest minimalizacja maksymalnego obciążenia CPU pomiędzy wszystkimi maszynami w klastrze. Ponieważ podobnie jak w rozdziale 3 obydwa postawione problemy optymalizacyjne (problem pakowania i minimalizacji maksymalnego obciążenia maszyn) są (poza specjalnymi przypadkami) NP-trudne,

zaproponowano szereg heurystyk przydziału maszyn do aplikacji, które zostały eksperymentalnie zbadane przy użyciu symulacji i zbioru danych Azure Public Dataset V2. Wykonane przypadki testowe różnią się konfiguracją maszyn (liczbą rdzeni i ilością pamięci). Uzyskane wyniki wskazują na to, że model "multi-instanced" poprawia wykorzystanie klastra w porównaniu z bazowym modelem "single-instanced".

Rozdziały 5 i 6 poświęcone są problemowi optymalizacji alokacji zasobów w systemie FaaS w ramach pojedynczej maszyny. Celem optymalizacji jest minimalizacja czasu odpowiedzi i spowolnienia (stretch/slowdown). W rozdziale 5 zaproponowano teoretyczny model szeregowania funkcji w obrębie pojedynczego węzła, który sprowadza się do podjęcia decyzji o kolejności wykonania oczekujących w kolejce wywołań funkcji. Wykorzystując symulację eksperymentalnie porównano wiele strategii takiego szeregowania — w tym bazowe FIFO i round-robin oraz nowo zaproponowane SEPT, SERPT i Fair Choice. Wyniki wskazują, że nowo zaproponowane strategie poprawiają (w niektórych przypadkach wielokrotnie) czas odpowiedzi i spowolnienie. W rozdziale 6 część algorytmów zbadanych w rozdziale 5 została zaimplementowana w systemie Apache OpenWhisk i zbadana eksperymentalnie w rzeczywistym systemie z wykorzystaniem funkcji z benchmarku SeBS. Wprowadzono również dwie nowe strategie szeregowania zapobiegające zagłodzeniu. Badania eksperymentalne potwierdzają, że zaproponowane strategie znacznie poprawiają zarządzanie zasobami, szczególnie w miarę wzrostu intensywności obciążenia. Ostatni eksperyment demonstruje przetwarzanie na 4-węzłowym klastrze, a wyniki robią szczególne wrażenie: zaproponowane strategie szeregowania umożliwiają przetworzenie tego samego obciążenia na trzech węzłach co strategie bazowe na czterech, poprawiając jednocześnie znacznie czasy odpowiedzi.

Rozdział 7 zawiera krótkie podsumowanie wyników.

Podsumowując, poziom merytoryczny rozprawy oceniam bardzo wysoko. Proponowane nowe algorytmy i strategie przydziału zasobów stanowią oryginalne rozwiązania istotnych problemów naukowych. Przedstawione wyniki posiadają dużą wartość poznawczą, a potencjalnie również praktyczną. Praca wyróżnia się też bardzo wysokim poziomem redakcyjnym.

4. Dyskusja

Bardzo proszę autora o ustosunkowanie się do następujących szczegółowych kwestii, które nasunęły mi się w trakcie lektury pracy.

Rozdział 3:

- Zaproponowany model oparty jest na dość mocnych założeniach: homogeniczność maszyn i przewidywalność czasu wykonania funkcji (clairvoyance). Proszę o szersze skomentowanie realistyczności tych założeń w odniesieniu do rzeczywistych systemów.

Rozdział 4:

- Pojęcie obciążenia (load) wydaje się być używane w dwóch znaczeniach: jako zbiór/strumień wywołań funkcji (s. 44-45) oraz jako obciążenie procesora przez aplikację wyrażone w vCPU (s. 47). Nie jest to duży problem, ale ta dwuznaczność była nieco myląca w trakcie lektury.
- Zaproponowany model z wieloma instancjami i równoważeniem obciążenia CPU wydaje się zachowywać całkowitą przepustowość aplikacji kosztem prędkości przetwarzania (czasów

odpowiedzi) poszczególnych wywołań. Ten aspekt nie jest omówiony w pracy. Przykładowo, czy przydział z lewej strony rysunku 4.2 nie jest (pod pewnym względem) "lepszy" od przydziału z prawej strony, bo jedna instancja działająca "z pełną prędkością" jest lepsza od dwóch działających z "połową prędkości"?

- Na s. 47, czy koniec zdania: "our aim is to assign applications to machines in such a way that all the incoming requests can be processed" nie powinien brzmieć: "...processed without sacrificing throughput"?
- Proszę szerzej skomentować istotność drugiego celu optymalizacji wyrażonego warunkiem (4.4) na s. 48 (min-max obciążenia CPU)
- Czy spodziewane jest, że rezultaty są skalowalne, tzn. utrzymają się w skali rzeczywistego centrum danych, gdzie będzie o wiele więcej aplikacji i maszyn?

Rozdziały 5 i 6:

- Czy możliwe byłoby połączenie zaproponowanych algorytmów z tymi, które przedstawiono w Rozdziale 4? Czy nie zachodzi tu sprzeczność celów optymalizacji:
  - w ramach jednej maszyny optymalizujemy ze względu na czas odpowiedzi i stretch/slowdown
  - w skali klastra (centrum danych) minimalizujemy liczbę węzłów, ale kosztem w/w parametrów (efekt uboczny metody "multi-instanced application")?
- W rozdziale 6 zastosowano podejście minimalizujące przełączenia kontekstu na poziomie systemu operacyjnego, co jest generalnie słuszne ze względu na duży koszt takich przełączeń. Jednakże może to prowadzić do bezczynności rdzeni w przypadku funkcji wykonujących operacje I/O. Na s. 96 znajduje się uwaga: "for I/O-intensive actions, some CPU cores may stay idle, although they could execute another function. As in the SeBS benchmark [54] we find both CPU- and I/O-intensive functions, we will verify the impact of that experimentally." Niestety nie znalazłem rozwinięcia tej uwagi w dalszej części rozdziału.

5. Konkluzja

Konkludując, przedłożona praca doktorska zawiera bardzo bogatą treść naukową dotyczącą ważnych i aktualnych problemów Informatyki. Problem naukowy przydziału zasobów w bezserwerowych platformach przetwarzania w chmurze jest potraktowany w sposób szeroki i kompleksowy. W mojej opinii praca spełnia wymogi stawiane rozprawom doktorskim, w odniesieniu do dyscypliny Informatyka, określone art. 187 ust. 1 i 2 Ustawy z dn. 20 lipca 2018 roku Prawo o szkolnictwie wyższym i nauce (tekst jednolity: Dz. U. z 2023 poz. 742). **Wnoszę zatem o jej przyjęcie i dopuszczenie mgra Pawła Żuka do dalszych etapów postępowania w sprawie nadania stopnia doktora.**

Ponadto, ze względu na wysoki poziom merytoryczny i redakcyjny, oraz dużą wartość poznawczą i praktyczną, **wnoszę o wyróżnienie rozprawy**.

Sacha Sala