Politechnika Wrocławska
Wydział Informatyki i Telekomunikacji

Kierunek: **Informatyka Stosowana (IST)**
Specjalność: **Projektowanie systemów informatycznych (PSI)**

PRACA DYPLOMOWA MAGISTERSKA

Metody optymalizacji wydajności rozwiązań ekosystemu Java w ramach AWS Lambda

Piotr Puchała

Opiekun pracy
dr inż. Michał Szczepanik

Słowa kluczowe: AWS Lambda, optymalizacja wydajności, Java, Kotlin, Kotlin Multiplatform, GraalVM

WROCŁAW 2025

Streszczenie

Dodaj streszczenie pracy w języku polskim. Staraj się uwzględnić wymienione na stronie tytułowej słowa kluczowe. Uwaga przedstawiony rekomendowany szablon dotyczy pracy dyplomowej pisanej w języku angielskim. W przeciwnym wypadku, student powinien samodzielnie zmienić nazwy „Chapter" na „Rodział" itp stosując odpowiednie pakiety systemu LaTeXoraz ustawienia w pliku *latex-settings.tex*.

Abstract

Streszczenie w języku angielskim.

Spis treści

Wstęp

TODO: Do napisania na końcu

Problem badawczy

Usługa AWS Lambda jest jednym z kluczowych serwisów oferowanych przez chmurę Amazon Web Services (AWS) w architekturze bezserwerowej. Wraz z rosnącą popularnością tego rodzaju architektur, pojawia się potrzeba ciągłej poprawy ich działania. Jednym z pierwotnie dostępnych języków programowania w AWS Lambda jest Java (oraz inne języki oparte o Java Virtual Machine), która jednak ze względu na swoją specyfikę (w tym wpływ na responsywność funkcji) nie jest najczęściej wybieraną opcją implementacji w tym serwisie.

Ważnym elementem pracy z AWS Lambda jest wydajność tworzonych funkcji, która przekłada się bezpośrednio na koszt usługi. Wpływ na wydajność mają takie czynniki jak na przykład tzw. zimne i ciepłe starty, czy czas działania samej funkcji, a niewystarczająco szybkie mogą powodować trudności dla programistów.

Mimo rozwoju wielu różnych technologii, ekosystem Java dalej cieszy się dużą popularnością wśród zespołów programistycznych. Poprawa wydajności funkcji AWS Lambda w tym ekosystemie ułatwi programistom decyzję o wyborze tego rozwiązania oraz pozwoli na lepszą pracę z już znanym językiem. Z tego powodu istnieje potrzeba analizy i zaproponowania metod optymalizacji wydajności dla funkcji AWS Lambda w ekosystemie Java.

Cel pracy

Celem pracy jest zaproponowanie nowych metod poprawy wydajności funkcji AWS Lambda w ekosystemie Java oraz analiza ich wpływu na czas działania funkcji i inne wybrane czynniki, które mogą wpłynąć na jakość pracy programistów.

Pytania badawcze

- PB1: Które metody optymalizacji pozwalają na najlepszą poprawę czasu wykonania funkcji AWS Lambda w ekosystemie Java?
- PB2: W jakim stopniu wybrane metody optymalizacji redukują czas zimnego startu funkcji Java w AWS Lambda?
- PB3: Jakie kompromisy w procesie rozwoju oprogramowania wiążą się implementacją poszczególnych metod optymalizacji wydajności funkcji Java w AWS Lambda?

Zakres pracy

Cel pracy zostanie zrealizowany poprzez następujące działania stanowiące zasadniczy wkład pracy:

1. Systematyczny przegląd literatury
2. Identyfikacja i zaproponowanie metod
3. Analiza wpływu

Struktura pracy

TODO: Do napisania na końcu

Słownik pojęć

Function as a Service (FaaS) -
Serverless -
AWS -
AWS Lambda -

1. Opis działania modelu FaaS

1.1. Model serverless

Jednym z dynamicznie rozwijających się obszarów chmur obliczeniowych są usługi serverless. W 2025 roku rynek usług opartych o architekturę bezserwerową jest wart 17,88 miliarda dolarów amerykańskich, a według prognoz jego wartość wzrośnie do 41,14 miliarda w roku 2029 [3]. Badania wykonane na otwartoźródłowych projektach serverless wykazały, że architektura ta używana jest ze względu na niższe koszty, uproszczenie procesów operacyjnych (jak wdrażanie, skalowanie i monitorowanie) oraz bardzo wysoką skalowalność [1]. Cechy te są osiągalne ze względu na wyjątkowe założenia tego modelu.

Przetwarzanie bezserwerowe możemy zdefiniować jako „formę przetwarzania w chmurze, która umożliwia użytkownikom uruchamianie aplikacji sterowanych zdarzeniami i rozliczanych granularnie bez konieczności zarządzania logiką operacyjną" [7]. W definicji tej znajdują się dwa ważne aspekty działania modelu bezserwerowego:

1. „Przetwarzania w chmurze, która umożliwia użytkownikom uruchamianie aplikacji (...) bez konieczności zarządzania logiką operacyjną."

2. „Aplikacji sterowanych zdarzeniami i rozliczanych granularnie."

Pierwszy punkt odnosi się do zwiększenia zakresu odpowiedzialności dostawcy chmurowego w porównaniu do klasycznych usług (np. Amazon Elastic Compute Cloud). W usługach tych fizyczne serwery są utrzymywane przez dostawcę chmurowego, a użytkownik jedynie wynajmuje jednostki obliczeniowe. Posiada on dalej kontrolę nad konfiguracją wielu aspektów infrastruktury, co pozwala na większą wydajność rozwoju oprogramowania w porównaniu z środowiskami niechmurowymi. Mimo to, dalej wymaga to poświęcenia czasu i środków na skonfigurowanie oraz zabezpieczenie aplikacji. Architektury bezserwerowe mają na celu uproszczenie tych procesów. Podczas tworzenia aplikacji w usługach bezserwerowych zespoły programistyczne nie muszą zarządzać wdrożeniem, a następnie utrzymaniem serwerów (nawet w formie jednostek jak AWS EC2). Rolą twórcy oprogramowania jest dostarczenie kodu aplikacji lub obrazu Docker [6] [5], które zostaną uruchomione w utrzymywanym przez dostawcę chmurowego środowisku. Dzięki temu inżynierowie mogą skupić się w większym stopniu na logice aplikacji. Pozwala to na zmniejszenie liczby obowiązków, a co za tym idzie kosztów zespołu [8].

Drugi punkt skupia się na charakterystycznym modelu płatności oraz sposobie działania architektur bezserwerowych, który umożliwia taki rodzaj rozliczeń. W przypadku klasycznych usług jak AWS EC2 płatność dokonywana jest za czas działania instancji, niezależnie od tego czy jest ona używana [4]. Model ten

2. Przegląd literatury

2.1. Wyniki przeglądu

W ramach przeglądu wybrano X prac badawczych. Na bazie prac rozpatrzono postawione pytania badawcze. Odpowiedzi na nie zostały zawarte w kolejnych podrozdziałach.

2.1.1. Czynniki wpływające na wydajność funkcji

Pierwszym pytaniem badawczym postawionym do przeglądu literatury jest: „Jakie są główne czynniki wpływające na wydajność funkcji AWS Lambda?". Identyfikacja czynników wpływających na wydajność jest kluczowa w kontekście jej optymalizacji. Pozwoli to następnie na zrozumienie na które z czynników ma także wpływ twórca funkcji AWS Lambda, co ułatwi dalszą analizę metod poprawy ich wydajności.

Wielkość pamięci funkcji

Kelly et al. [2] zauważa, że wielkość pamięci funkcji oprócz bezpośredniego wpływu na czas działania funkcji, wywiera także wpływ na inne czynniki jak użycie procesora czy wydajność I/O dysku. W ramach badania wykonano pomiary dla funkcji bezserwerowych oferowanych przez wielu dostawców chmurowych, w tym Amazon Web Services, w celu zrozumienia infrastruktury i jej zarządzania, co domyślnie jest ukryte dla użytkownika funkcji jak AWS Lambda. Poprzez analizę maszyn wirtualnych, w ramach których uruchamiany jest kod funkcji, możliwe było otrzymanie wartości parametrów, które nie są domyślnie konfigurowalne podczas wdrażania funkcji przez programistę.

Pomiary pokrywały wiele parametrów funkcji, m. in. łączny czas wykonania, czas inicjalizacji, użycie procesora, wydajność I/O dysku oraz liczbę utworzonych maszyn wirtualnych (co wpływa na częstość zimnych startów). W badaniu uwzględniono predefiniowane wielkości pamięci, które mogą być wybrane przez programistę (128MB, 256MB, 512MB, 1024MB oraz 2048MB). Wraz z wzrostem pamięci funkcji, parametry te poprawiały się.

Autorzy podkreślili ważność odpowiedniego doboru wielkości pamięci podczas tworzenia funkcji. Dodatkowo, wykazali, że platforma AWS nie używa ponownie maszyn wirtualnych w celu wykonania kolejnych zapytań funkcji, co powoduje częstsze zimne starty. Wykazano zatem, że te dwa czynniki (wielkość pamięci i zimne starty) znacząco wpływają na ogólną wydajność funkcji AWS Lambda.

Bibliografia

[1] S. Eismann, J. Scheuner, E. van Eyk, M. Schwinger, J. Grohmann, N. Herbst, C. L. Abad, and A. Iosup. Serverless applications: Why, when, and how? *IEEE Software*, 38(1):32–39, 2021.

[2] D. Kelly, F. Glavin, and E. Barrett. Serverless computing: Behind the scenes of major platforms. In *2020 IEEE 13th International Conference on Cloud Computing (CLOUD)*, pages 304–312, 2020.

[3] Research and Markets. Serverless architecture market report 2025, (2025). Dostęp 20.03.2025 z https://www.researchandmarkets.com/reports/5953253/serverless-architecture-market-report.

[4] A. W. Services. Amazon ec2 documentation, 2025. Dostęp 21.03.2025 z https://docs.aws.amazon.com/ec2/.

[5] A. W. Services. Amazon ecs developer guide, 2025. Dostęp 21.03.2025 z https://docs.aws.amazon.com/AmazonECS/latest/developerguide.

[6] A. W. Services. Aws lambda documentation, 2025. Dostęp 21.03.2025 z https://docs.aws.amazon.com/lambda/latest/dg/.

[7] E. van Eyk, A. Iosup, C. L. Abad, J. Grohmann, and S. Eismann. A spec rg cloud group's vision on the performance challenges of faas cloud architectures. In *Companion of the 2018 ACM/SPEC International Conference on Performance Engineering*, page 21–24, New York, NY, USA, 2018. Association for Computing Machinery.

[8] J. Wen, Z. Chen, X. Jin, and X. Liu. Rise of the planet of serverless computing: A systematic review. *ACM Trans. Softw. Eng. Methodol.*, 32(5), 2023.