Oscar Teeninga

1. Cel projektu

Stworzenie klastra Kubernetes'owego zarządzanego przez Kubeless w taki sposób, by rozwiązywał jeden z przykładowych problemów:

- Przetwarzanie dużego zestawu danych:
  - szeregów czasowych
  - dużych danych tekstowych
  - tagowanie obrazów
  - rozpoznawanie obrazów
- Łamanie haseł (np. hashcat)
- Rozproszone renderowanie np. povray, blender
- Metody numeryczne, np. całkowanie MC
- Konwersja/prosta analiza dużej ilości obrazów, np.: imagenet

Dodatkowo celem będzie zapoznanie się z całym narzędziem Kubeless oraz zwiększenie swoich kompetencji w zakresie Kubernetesa.

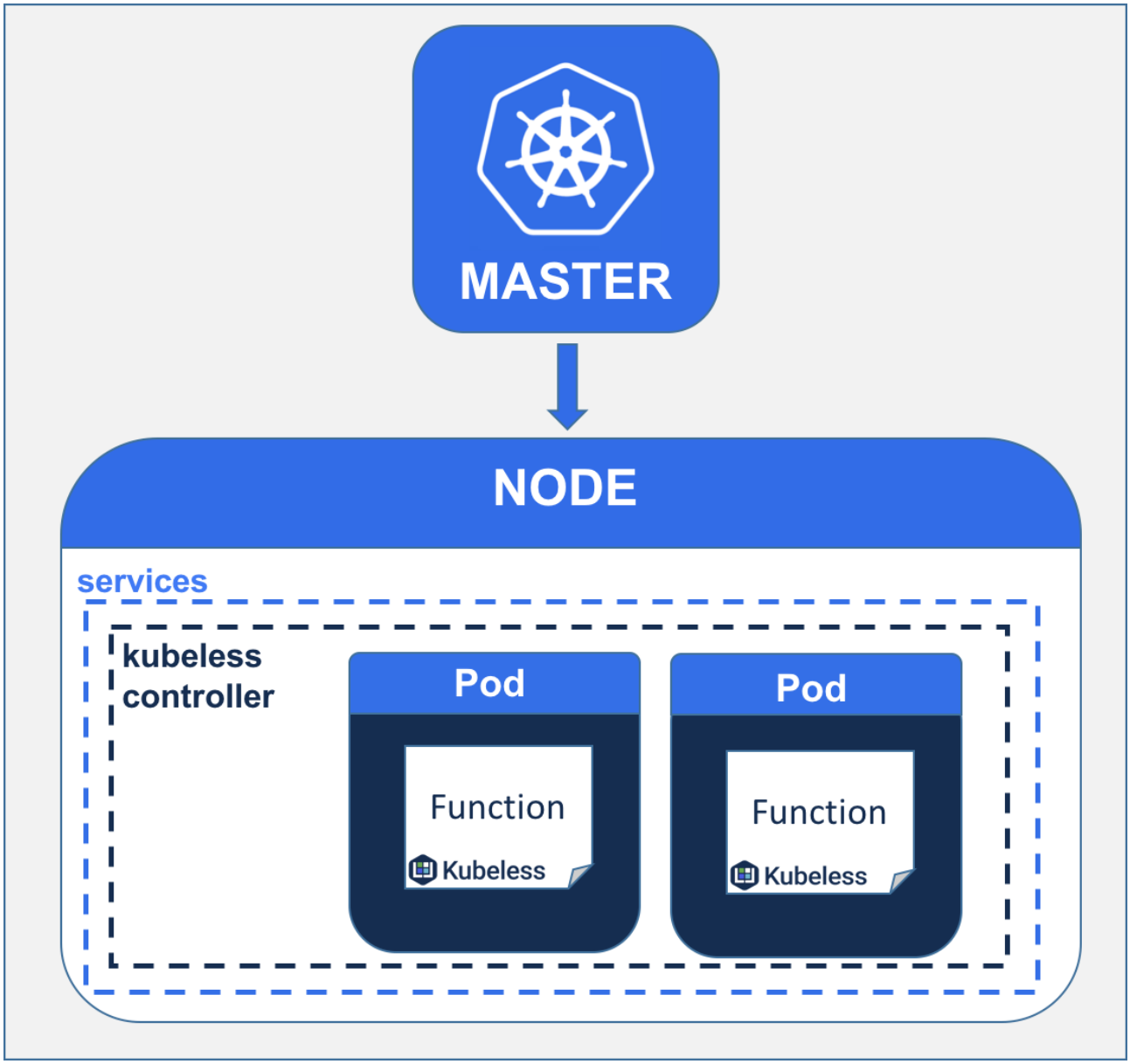
2. Wymagania

Wymagane jest, by posiadać komputer z system operacyjnym Linux, MacOS lub Windows. Wszystko opiera się na Kubernetes, a więc jest on również wymagany. Wspierane języki programowania to Python, Node.js, Ruby, PHP, Golang, .NET oraz Ballerina. Zdecyduje się na Pythona.

Problematyczną kwestią może być fakt, że Kubeless stracił swoje wsparcie - przez co musiałem skorzystać z starszej wersji minikube oraz kubernetesem w wersji 1.16.

3. Architektura

Głównym zadaniem Kubelessa jest stworzenie klastra zarządzanym przez mastera. Wszystko opierać będzie się na Kuberetesie stworzonym przy pomocy minikube oraz Dockera. Funkcje będą wykonywane przez pody, które będą klastrami w Kubernetesie. Ewaluacja funkcji w przypadku Kubelessa jest bardzo proste.



4. Testy

Pomysł Kubelessa polega na stworzeniu dodatkowego menadżera nad kubernetesem, dzięki któremu możliwe będzie wykonywanie dowolnych funkcji w pomyśle serverless. Można to robić z poziomu GUI'a.
Będę sprawdzał czy wykonywane funkcje dają poprawne efekty oraz zbadam możliwości skalowalności rozwiązania.
Dodatkowo mogę sprawdzić ile względem normalnego rozwiązania trwa wykonanie zadania.
Wybrałem problem MonteCarlo i obliczenie całki oznaczonej z f(x) = 2x^3 + 3x^2 + 8x - 12 dla granic całkowania [0, 2]. Wykorzystałem 1 mln próbek. Dało to wynik 19.40512834732532. Porównałem czasy dla kubelessa i normalnego uruchomienia:
Kubeless: 4.999589204788208 s
Normal: 4.5659730434417725 s
Ograniczenie wydajności: ~ 91.32%
Jak widzimy, rozwiązanie Kubeless ze względu na konteneryzację powoduje spadek wydajności względem normalnego uruchomienia.

5. Rozwiązanie

Samo rozwiązanie jest bardzo proste, ponieważ Kubeless ma za zadanie ułatwić wiele rzeczy, a więc samo postawienie i uruchomienie funkcji z jego poziomu nie powinno sprawić problemu. W końcu jest to przykład obliczeń serverless, które bazować mają na prostym użyciu w pewnych sytuacjach. Ciekawy zagadnieniem na pewno będzie wydajność i zasobożerność tego rozwiązania (względem normalnego uruchomienia skryptu). Spróbuje również uruchomić funkcję z poziomu innego komputera wysyłając żądanie przez internet (tak jak ma to miejsce w przypadku AWS).