K8s 환경에서 최소 리소스를 가진 컨테이너 할당 알고리즘

김두현

Overview

- 프로젝트 소개
- 개발 환경 및 도구 정의
- 할당 알고리즘 소개
- 결론

프로젝트 소개

클라우드 환경에서 컨테이너(Docker, Kubernetes)를 운영할 때 자원(cpu, memory)을 **최적화**하여 비용 절감을 이루는 자동화된 알고리즘 개발

최종 목표

서비스 요청(n회)을 정해진 시간(n초) 이내로 모두 처리하는 **최소 리소스**를 가진 컨테이너 추천 알고리즘 개발

Why?

현대 클라우드 환경에서 컨테이너는 애플리케이션 배포의 핵심이 되었지만, 리소스 할당이 최적화되지 않으면 초기 컨테이너 할당시 불필요한 비용이 발생

개발 환경 및 도구 정의

컨테이너 관리

- Docker
- K8s

부하 테스트

- k6 부하 테스트 도구
- 사전에 정의된 기준(TPS, etc ...) 내에 처리할 수 있는지 검증

모니터링

- Grafana
- 컨테니어 cpu 사용률, 메모리 사용량, 응답 시간 등 지표 수집, 시각화

이차원 이진 탐색

가정: (n, k) 리소스를 가진 컨테이너가 응답 성공했으면, (n+1, k) or ((n, k+1) 리소스를 가진 컨테이너 또한 통과

- CPU 수, 메모리 크기로 구성된 2차원 배열 정의
- 리소스 크기를 기준으로 오름차순으로 정렬

01

이진 탐색 기반 접근

- 각 CPU 값에 대해 최소 메모리 크기를 이진 탐색으로 검색
- OOM Killed이 발생하지 않으며, 기준 내 요청이 완료된 조합
- 이진 탐색은 O(log n) 시간 복잡도로 효율적인 탐색이 가능

02

탐색 범위 최적화

- 이전 CPU 값에서 찾은 최소 메모리 인덱스를 다음 CPU 값의 탐색 범위로 제한

EX)
(cpu core 2, mem 1024MB)가 저번 탐색에서 최소 값이였으면 (cpu core 3, mem 1024MB) 또한 통과할거라고 기대 가능하기 때문

03

최종 선택 기준

- 각 CPU 값에 대해 요청을 처리할 수 있는 최소 메모리 크기를 확인
- CPU와 메모리에 가중치를 부여한 점수를 계산하여 가장 낮은 점수를 가진 조합을 최종 선택

알고리즘 동작 예시 - 1

CPU ↓ \ Memory →	128MB	256MB	512MB	1024MB	2048MB	4096MB
1.0	X	X	X	X	X	O
2.0	X	X	X	O	O	O
3.0	X	X	O	O	O	O
4.0	X	X	O	O	O	O
5.0	X	X	O	O	O	O
6.0	X	O	O	O	O	O

- X: 요청 처리 실패 또는 OOM Killed
- O: 기준 내 요청 처리 성공

결론

장점

- 완전 탐색은 리소스에 대한 모든 조합(n:cpu 수, k: 메모리 수)에 대해 컨테이너를 생성해 많은 비용
- 해당 알고리즘은 O(nlogk)의 시간 복잡도로 비용 절감 가능
  -> 테스트 조합 수가 많을 수록 효과적!

단점

- 이전 컨테이너에 대한 응답 결과가 필요해 동시에 여러 컨테이너를 테스트하기 어려움
  -> 컨테니어 테스트 시간이 많이 소요