[양식 4] 과제 요약보고서

[캡스톤디자인 과제 요약보고서]

과 제 명	Microservice Architecture에서 L/B, autoscaling algorithm에 따른 성능 평가

1. 과제 개요

가. 과제 선정 배경 및 필요성

Microservice Architecture로 서버를 개발할 때, 다양한 load balancing, autoscaling 방법들이 존재한다. 이때, 프로그램의 연산량, 동기/비동기 처리, I/O, 동시 처리 등 프로그램의 특성에 따라 효율이 프로그래밍 언어나, L/B 방법에 따라 다르다. 따라서, 본 연구를 통해 Microservice Architecture로 서버를 개발할 때 L/B 방법 및 프로그래밍 언어 선택에 있어 선택의 기준을 제시하도록 한다.

나. 과제 주요내용

- Variables (server): I/O 연산량, 동시 요청 수, 컨테이너의 크기, microservice의 수
- Variables (L/B): thread isolation, L/B rule, programming language

위 내용들에 대해 지연 시간, 자원 사용량, 최대 동시 처리 가능 요청 수들을 측정한다.

다. 최종결과물의 목표

상황에 따른 가장 최적화된 프로그래밍 언어 및 L/B 설정 값들을 알 수 있다.

2. 과제 수행방법

동시 사용자수: 1000명
측정 기준: 응답 시간이 100ms를 초과하면 실패

실험A: Kubernetes에 서비스를 LoadBalancer로 배포하고, 서비스에 직접 접속하도록 실험
실험B: Kubernetes에 서비스 및 Zuul, Eureka를 배포하여 Zuul에 접속하도록 실험
실험C: 실험B에서 Load balancer를 직접 상황에 맞게 프로그래밍 하여 실험

각각 Google Cloud Platform의 Kubernetes Engine에 배포하여 Apache Jemeter로 트랜잭션과 응답 시간을 측정

3. 수행결과

가. 과제수행 결과

실험A: 서비스만 등록한 경우
Avg. response time: 90ms
Error rate: 16.21%
Transaction: 1,185.4/sec

실험 B-1: Load Balancer로 Rond Robin을 사용
Avg. response time: 81ms
Error rate: 8.74%
Transaction: 1,206.4/sec

실험B-2: Load Balancer로 ResponseTimeWeighted를 사용
Avg. response time: 80ms
Error rate: 5.89%

Transaction: 1,227.1/sec

실험C-1: Response time으로 Server List Filtering
Avg. response time: 84ms
Error: 2.66%
Transaction: 1,178.4/sec

실험C-2: Recent Response time으로 Server List Filtering
Avg. response time: 73ms
Error rate: 1.04%
Transaction: 1,334.4/sec

실험C-3: Recent Response time으로 Server List Filtering, Compare Server
Avg. response time: 73ms
Error rate: 0.92%
Transaction: 1,334.4/sec

실험C-4: Recent Response time으로 Server List Filtering, 최저 응답 시간의 서버로 Load Balancing
Avg. response time: 73ms
Error Rate: 0.83 %
Transaction: 1,340.0/sec

나. 최종결과물 주요특징 및 설명

실험A: Kubernetes만 사용해서 성능이 가장 좋지 않았다.
실험B-1: Round Robin에 가중치가 없기 때문에 새로 생성된 서버로 제대로 분산이 되지 않았다.
실험B-2: Weight가 존재하여 B-1보다 더 개선된 성능을 얻을 수 있었다.
실험 C-1: 성능이 개선되었지만, 사용 가능한 서버의 수가 줄어드는 때가 있기 때문에 서비스가 원할히 공급되지 않는 구간이 존재한다.
실험 C-2: 평균 응답 시간을 최근 1초의 데이터만 사용하도록 하여 실험 C-1 대비 개선된 성능을 보였다.
실험 C-3: L/B시 연산 작업의 수를 확률적으로 감소시켜 성능이 개선되었다.
실험 C-4: L/B시 서버 목록을 확인하지 않도록 수정하여 성능이 더 개선되었다.

Kubernetes만 사용할 경우, 응답 실패 비율이 16.21%였는데, 수정된 결과 0.83%까지 개선됨.

4. 기대효과 및 활용방안
가. 기대효과
- L/B 전략 선택에 있어서의 지표 제공

나. 활용방안
- Microservices architecture로 서비스를 제공할 때 보다 안정적인 서비스 제공을 위한 L/B 전략 수립 및 알고리즘 작성에 있어 참고 자료로 활용 가능

5. 결론 및 제언

성능: Kubernetes < Ribbon (default) <Customized Load Balancer
L/B 성능은 서버 상황에 맞도록 프로그래밍 하는 것이 중요하며, 그렇지 않더라도 Kubernetes Load Balancer보다 Netflix Zuul에 Ribbon에서 제공하는 Load Balancer를 사용하는 것이 성능이 월등히 좋다.

※ 본 양식은 요약보고서이며, 최종결과물을 필히 추가 제출하여야 함.

팀 학생대표 성명 : 차 수환 (인)