서버리스 플랫폼에서의 정확한 요금 산정을 위한 초 저지연 리소스 미터링 모듈

최상훈*, 이병용*, 박건호*, 박기웅†

*세종대학교 시스템 보안 연구실, † 세종대학교 정보보호학과

Low Latency Resource Metering Module for Fair Pricing in Serverless Platform

Sang-Hoon Choi, Byeong-Yong Yi, Keon-Ho Park, Ki-Woong Park†

*SysCore Lab., Sejong University.

† Department of Computer and Information Security, Sejong University

요 약

최근 클라우드 컴퓨팅 기술이 발달하면서 함수 단위의 서비스 제공 플랫폼인 서버리스 플랫폼이 등장하게 되었다. 서버리스 플랫폼에선 기본적으로 리소스 모니터링 주기를 기반으로 API 사용에 따른 과금이 이루어진다. 따라서 정확한 과금을 위해선, 리소스 사용량을 측정할 수 있는 리소스 모니터링 도구의 성능이 중요하다. 본 논문에선, 기존에 서버리스 환경에서 사용하는 리소스 미터링 도구 한계점을 해결 하기위해 Linux Kernel 모듈 기반의 초 저지연 리소스 미터링 방안을 제안한다. 본 논문에서 제안하는 모듈은 기존 컨테이너 모니터링 도구인 cAdvisor에 비교하여 연산 오버헤드가 적고, 빠른 주기로 리소스 메트릭 수집이 가능하다. 본 논문에서 제안한 기술은 서버리스 서비스 사용량에 따른 정확한 요금 산정을 위한 기반 기술로 활용이 가능하다.

I. 서론

클라우드 컴퓨팅 기술의 발달로 인해 클라우드의 장점을 활용한 다양한 서비스들이 등장하고 있다. 클라우드 컴퓨팅은 기존의 레거시 시스템과 비교하여 비용 측면의 절감 효과가 있다는 장점이 있어 그에 따른 수요가 증가하고 있다. 요금 관련 트렌드에 대해 가장 잘반영한 기술은 FaaS(Function as a Service)가 있다. FaaS는 사용자가 원하는 기능(로직)을 함수로 작성하면 컨테이너를 통해 해당 함수를 수행할 수 있도록 지원하는 서비스이다. 따라서 사용자는 함수 수행 요청에 대해, 실질적으로 컨테이너에서 함수 실행된 시간에 대해서만 요금만 지불하면 된다는 특징이 있다. 하지만, 기본적으로 함수의 실행 시간 측정은 리소스 모니터링 기술이 기반이 되므로, 서버리스 플랫폼에서 세밀한 요금 계산을 위해선 리소스 모니터링 도구의 성능이 중요하다. 본 논문에서는 기존 서버리스 플랫폼에서 사용하는 모니터링 도구의 성능 개선을 위해 Linux Kernel 모듈을 활용한 초 저지연 리소스 미터링 모듈을 제안한다. 제안하는 미터링 모듈은 기존 컨테이너 환경에서 사용하는 cAdvisor[1]에 비해 낮은 오버헤드를 발생시키며, 더 빠른 주기로 리소스 미터링이 가능하다. 본 논문의 구성은 다음과 같다. 2장에서는 본 논문의 기반이 되는 배경지식에 대해서 설명한다. 3장에서는 본 논문에서 제시하는 초 저지연 리소스 미터링 모듈을 제안한다. 4장에서는 기존의 컨테이너 시스템에서 사용하는 미터링 도구인 cAdvisor와 본 논문에서 제시하는 미터링 방안에 대한 성능 비교를 수행하고 5장에서는 결론 및 향후 연구에 대해서 기술한다.

II. 배경지식

본 장에서는 상용 서버리스 플랫폼에 대해 설명하고, 미터링에 사용되는 cAdvisor 등의 컨테이너 모니터링 도구의 특징에 대해서 설명한

다. 또한, 기존 서버리스 플랫폼의 과금 체계에 대해 기술하고, 과금체계의 공통적인 한계점에 대해서 기술한다.

2.1 서버리스 플랫폼 서비스

기존의 서버리스 플랫폼에는 Amazon 사의 AWS Lambda[2] Google 사의 Google Cloud Functions[3] 이 있다.

AWS Lambda: AWS Lambda는 Amazon 사에서 제공하는 상용 서버리스 플랫폼으로서 2014년에 서비스를 공개하였다. 현재 Python, Go, Java, C# 등의 언어를 지원하고 있으며, 요청한 API의 구동을 위해 할당 가능한 최대 시간은 900초이다. 플랫폼 내부의 성능 및 이벤트 모니터링, 로깅 기능을 수행하기 위해 CloudWatch[4]를 사용한다. API 구동에 할당한 시간에 따라 100ms 단위로 요금이 부과된다는 점이 있다. 즉, 구동 시간 관련하여 부과되는 최소 요금은 100ms부터 부과된다는 것을 알 수 있다.

Cloud Functions: Cloud Funtions는 Google 사에서 제공하는 상용 서버리스 플랫폼으로서, 2018년에 서비스를 공개하였다. 현재 지원 가능한 언어는 Python, Go, Node.js 등의 언어이며, 요청한 API의 구동을 위해 할당 가능한 시간 최대 540초이다. 플랫폼 내부의 로깅 및 모니터링 도구로서 Stackdriver[5]를 사용한다. Cloud Funtions 또한, API 구동에 할당한 시간에 따라서 100ms 단위로 요금을 부과한다.

2.2 서버리스 플랫폼의 과금 정책에서 리소스 미터링 도구의 역할

cAdvisor는 Google 사에서 오픈소스로 배포하고 있는 도커 컨테이너 리소스 사용량 도구이다. cAdvisor는 리눅스 커널에서 프로세스의 리소스 사용의 관리를 수행하는 cgroups 기반으로 개발되었으며, 컨테이너 워크로드 관리를 위한 플랫폼인 쿠버네티스 환경에서 리소스 모니터링 도구로서 사용되고 있다.

서버리스 플랫폼에서 기본적인 과금관련 정책은 리소스 사용량에 따른 과금이다. cAdvisor 등의 모니터링 도구를 통해 리소스 사용량 측정을 수행한 뒤, 리소스 사용량과 사용 시간에 비례하여 과금을 수행하므로 서버리스 플랫폼에서 과금 서비스의 기반 역할을 수행한다. 따라서, 최대 측정 가능한 주기 및 모니터링 도구 자체의 연산 오버헤드는 과금 서비스의 기본적인 요소가 된다.

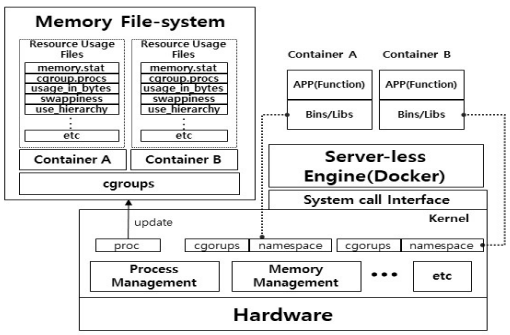
2.3 기존 서버리스 플랫폼의 과금 관련 모니터링 도구와의 연관성 및 한계점

AWS Lambda, Google Clound Funtions 등의 서버리스 플랫폼은 기본 과금 단위가 100ms 단위로 반올림하여 정산된다는 특징이 있다. 이처럼 100ms 단위의 요금 정책을 선정한 이유는 미시적 관점에서 100ms 보다 낮은 단위의 리소스 사용량 측정에 한계가 있기 때문이다. 서버리스 플랫폼에서 리소스 사용량 미터링을 위해 사용하는 cAdvisor등의 리소스 모니터링 도구의 최대 수집주기가 곧 서버리스 플랫폼 모니터링 도구의 성능이 이어짐을 의미한다. 따라서, 악의적인 사용자는 Public API를 Function Container를 통해 사용자에게 제공할 경우 발생할 수 있는 과요금 문제를 해결하기 위해선, 기존 모니터링 도구의 리소스 수집 주기에서 개선이 필요하다.

III. 초 저지연 리소스 미터링

본 장에서는 기존 리소스 사용량 수집 방법의 성능적인 문제점을 분석하고, 이를 해결 한 초 저지연 리소스 미터링 모듈을 설명한다.

3.1 cAdvisor 분석을 통한 문제점 도출



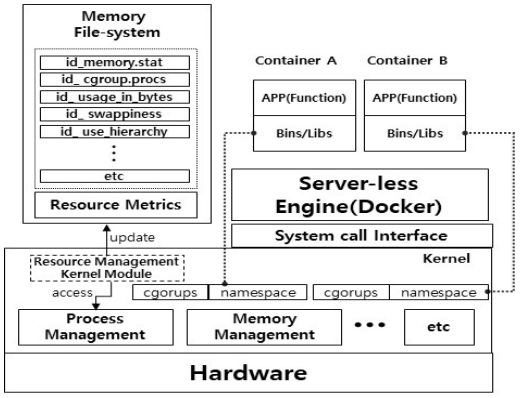
[그림 1] 기존 리소스 사용량 수집 방법

본 연구에서는 기존 컨테이너 플랫폼에서 리

소스 사용량 측정을 위해 사용되는 cAdvisor가 리소스 메트릭 정보를 빠른 주기로 수집하지 못하는 원인을 도출하기 위해서 소스코드 레벨에서 분석을 진행하였다.

분석을 통해 도출 된 원인을 다음과 같이 도출 할 수 있었다. 기존의 cAdvisor의 경우 proc 파일 시스템 내부에 생성되는 프로세스 리소스 정보를 파일로 접근하여 읽기를 통해 사용량 정보를 수집한다. 하지만, 리소스 사용량에 대한 정보는 컨테이너(프로세스) 마다 proc 파일 시스템 내부 별도의 영역에 존재하며, 리소스 종류에 따라 별도의 파일(memory.stat, memory.usage_in_bytes 등)로 관리 된다. 이와 같은 과정에서 프로세스 리소스 정보의 파일 위치를 추적하기 위한 연산과 다양한 파일에 대한 반복적인 읽기 연산으로 인해 Context Switch가 빈번하게 발생한다.

3.2 초 저지연 리소스 미터링 설계 및 구현



[그림 2] 리소스 미터링 커널 모듈 구조

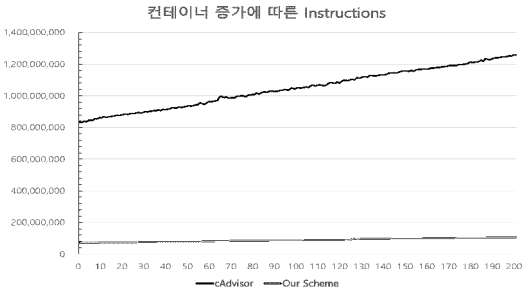
cAdvisor 분석을 통해 컨테이너 리소스 사용량 측정에서 발생하는 불필요한 연산 오버헤드의 문제점을 파악하였다. 이와 같은 문제를 해결하기 위해 [그림 2]과 같이 커널 모듈을 통해 컨테이너의 리소스 사용량 정보를 효율적으로 수집 할 수 있도록 리소스 미터링 커널 모듈을 설계 및 구현 하였다.

본 논문에서 제안한 리소스 미터링 커널 모듈은 도커 엔진을 통해 구동중인 컨테이너 pid 추적을 통해 프로세스 구조체에 직접 접근하여, 프로세스 구조체 정보(리소스 관련 정보)가 업데이트 될 때마다 리소스 사용량 정보를 ReousrceMetric 이라는 proc 파일시스템 영역에 리소스 메트릭 정보를 업데이트 한다. 따라서, 단 한번의 proc 파일 시스템 영역 읽기연산을 수행함으로써 시스템 내 구동중인 모든 컨테이너들의 리소스 사용량 메트릭 정보를 수집할 수 있다.

IV. 성능평가

4.1 연산 오버헤드 성능평가

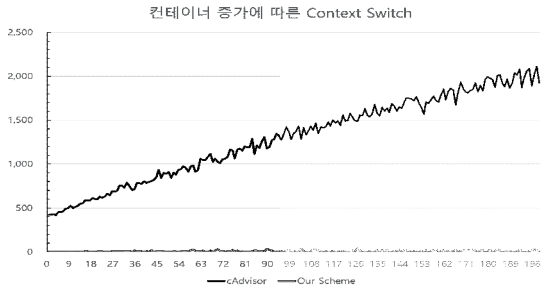
본 연구에서 제안한 초 저지연 리소스 미터링 커널 모듈의 오버헤드 성능 측정하기 위해, 컨테이너 리소스 자원 수집에 활용되는 Google 사의 cAdvisor와 성능을 비교하였다. 실험 방법은 Linux에서 제공하는 벤치마크 도구인 Perf를 사용하여 수행하였으며, Perf가 지원하는 성능관련 지표 중 Context Switch와 Instrcutions를 기준으로 성능평가를 수행하였다.



[그림 3] 컨테이너 증가에 따른 Instructions 변화

cAdvisor가 컨테이너 리소스 자원 수집할 때 컨테이너의 cgroups 파일 추적, 파일 읽기, 리소스 관련 정보가 포함된 구조체 접근 및 읽기가 발생한다. 제안한 초 저지연 리소스 미터링 커널 모듈의 경우 리소스 관련 정보가 포함된 구조체에 직접 접근하기 때문에 컨테이너의 cgroups 파일 추적 과정이 생략되고, 파일 읽기가 한번만 수행된다. Context Switch는 Task와 비례하여 발생한다. 따라서 컨테이너의 수만큼 파일 접근 및 읽기가 발생하는 cAdvisor에 비해, 컨테이너 수와 관계없이 한번의 파일 읽기가 발생하는 초 저지연 리소스 미터링 커널 모

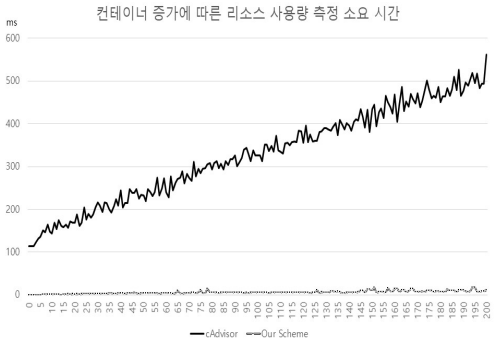
듈이 적은 Context Switch를 유발한다.



[그림 4] 컨테이너 증가에 따른 Context Switch 변화

Instructions에 대한 성능 평가는 [그림 4]와 같다. Instructions는 리소스 자원 수집을 위해 수행되는 과정에 비례하며, 파일 접근 및 읽기, 구조체 접근 및 읽기 등 여러 단계를 거쳐 정보를 수집하는 cAdvisor와 비교하여 본 논문에서 제안하는 모듈은 구조체에 직접 접근하기 때문에 최소의 명령어로 자원 사용량 수집이 가능하다.

4.2 리소스 수집 소요시간 성능평가



[그림 5] 리소스 수집 소요 시간 비교

리소스 수집 소요 시간은 서버리스 환경에서 정확한 리소스 미터링(요금산정)을 미세하게 측정하기 위한 가장 중요한 요소이다. 본 장에서는 제안한 초 저지연 리소스 미터링 모듈과 CAdvisor의 리소스 수집 소요 시간에 대한 성능을 비교한다. 결과는 [그림 5]와 같다. 동일한 환경에서 리소스 수집를 수행할 경우, 리소스 수집 소요 시간은 한 번의 리소스 수집 연산에 비례한다. 따라서 연산 오버헤드 성능 평가를 통해 적은 연산 오버헤드를 발생시키는 초 저지연 리소스 미터링 모듈이 적은 수집 소요 시간을 보이며, 이에 따라 더 빠른 주기로 수집가능하다는 것을 볼 수 있다.

V. 결론

본 논문에서 수행한 실험 결과에서는 제안한 초 저지연 리소스 미터링 모듈이 cAdvisor에 비해 낮은 연산 오버헤드와 빠른 리소스 수집 주기를 보였다. 연산 오버헤드 측면에서는 Context Switch와 Instructions을 기준으로 연산 오버헤드를 측정하였다. Context Switch의 경우, Task와 비례하여 발생하기 때문에, 파일 읽기가 컨테이너의 수만큼 발생하는 cAdviosr와 비교하여 한 번의 파일 읽기가 수행되는 초 지연 리소스 미터링 모듈이 더 적은 Context Switch를 유발하였다. 또한 Instructions은 cAdvisor는 리소스 사용량 측정을 위해 컨테이너의 cgroups 파일 추적과 파일 읽기, 리소스 사용량 측정을 위한 구조체 접근 등 여러 단계를 거치며 이에 대한 명령어가 수행되는 반면, 제안한 초지연 리소스 미터링 모듈은 cgroups 파일 추적 과정이 생략되고, 파일 읽기가 한번만 수행되며, 리소스 사용량 측정을 위한 구조체에 직접 접근하기 때문에 연산에 필요한 명령어 수가 최소화 된다. 또한 하나의 메모리 영역(파일)에 대해 접근하기 때문에 Context Switch가 최소화 된다. 본 논문에서 제안한 기술은 서버리스 서비스 사용량에 따른 정확한 요금 산정을 위한 기반 기술로 활용할 수 있다.

[참고문헌]

[1] Casalicchio, Emiliano, and Vanessa Perciballi. "Measuring docker performance: What a mess!!!." Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering Companion. ACM, 2017.

[2] Villamizar, Mario, et al. "Infrastructure cost comparison of running web applications in the cloud using AWS

lambda and monolithic and microservice architectures." 2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid). IEEE, 2016.

[3] Google Cloud Function,

https://cloud.google.com/functions

[4] Cloud watch,

https://aws.amazon.com/cloudwatch/

[5] Statck Drvier,

https://cloud.google.com/stackdriver