멀티쓰레드 자바스크립트 어플리케이션 실행을 위한 SGX library OS 최적화

이철민 [1], 이병영 [2]
[1] 서울대학교 전기정보공학부 석박통합과정
[2] 서울대학교 전기정보공학부 교수

2harry@snu.ac.kr, byoungyoung@snu.ac.kr

Optimization of SGX library OS for executing multi-thread JavaScript application

Chul-Min Lee[1], Byoung-Young Lee[2]
[1]Dept. of Electrical and Computer Engineering, Seoul National University
[2]Dept. of Electrical and Computer Engineering, Seoul National University

요 약

자바스크립트는 어플리케이션 및 서비스를 개발하는 주요한 언어 중 하나이다. 자바스크립트는 현재 서버 측에서도 널리 사용되고 있으며 서버에서 계산 집약적인 어플리케이션을 수행하기 위해 멀티쓰레드 기능도 최근 추가되었다. 이 논문에서는 Intel SGX 를 활용하여 클라우드 환경에서 자바스크립트 어플리케이션에 대한 보안을 강화하려는 시도를 다룬다. 그 과정에서 SGX 의 library OS 가 자바스크립트 어플리케이션을 수행하는 데에 있어서 발생하는 성능 저하를 다루며, 이를 최적화하여 극복하는 방안을 제시한다. 또한, 실험 결과를 통해 제안된 최적화 기법이 어떻게 성능을 향상시키는지를 확인한다.

1. 서론

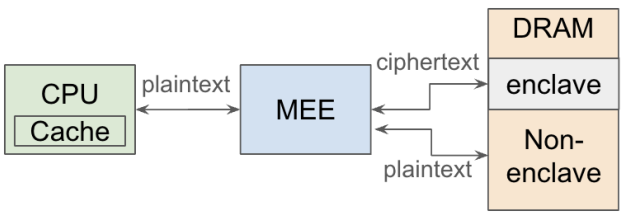
자바스크립트는 어플리케이션이나 서비스 개발에서 주로 쓰이는 프로그램 언어 중 하나이다. 자바스크립트는 웹 브라우저에서 사용자가 더 다양한 로직을 경험할 수 있게 도와줄 뿐만 아니라, 서버 측에서도 널리 사용되고 있다. 개발자는 자바스크립트를 이용해 어플리케이션을 개발하고 이를 Node.js[1]와 같은 환경을 이용해 서버 측에서 어플리케이션을 실행하여 클라이언트에게 서비스를 제공한다.

자바스크립트의 멀티쓰레드 기능은 서버 어플리케이션을 실행하는 데에 있어서 자바스크립트가 가지고 있는 단점을 보완하기 위해 새롭게 등장했다. 초기 자바스크립트는 단일 쓰레드 모델을 채택했기 때문에 연산 집약적인 서비스를 제공하는 데에 있어서 어려움이 있었다. 이를 보완하기 위해 자바스크립트는 Worker[2]라는 새로운 기능을 통해 멀티쓰레드 형태의 자바스크립트 어플리케이션을 개발할 수 있도록 했다.

Intel SGX[3]는 클라우드 환경에서 서버 어플리케이션을 실행할 때 클라우드 유저의 데이터나 로직을 보호하기 위해 사용된다. SGX 의 단점 중 하나는 SGX 를 사용하기 위해 사용자 코드를 재구성해야할 필요가 있다는 것인데, SGX 의 library OS, 그 중 가장 대표적으로 사용되는 gramine [4]는 그런 수고를 덜어준다.

이 논문에서는 서버에서 멀티쓰레드 자바스크립트 어플리케이션을 SGX 를 이용해 실행할 때, SGX library OS 가 일으키는 성능 저하를 분석하고, 이를 해결하기 위한 optimization 기능을 제시하며 그 영향을 실험을 통해 보여준다.

2. 배경 지식

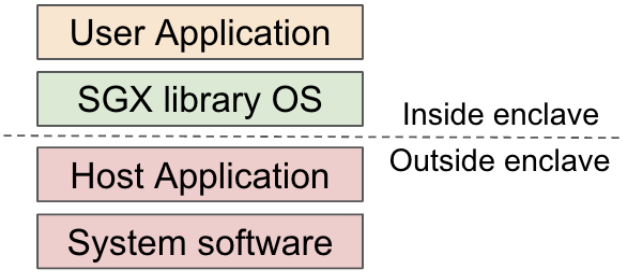


<그림 1> Intel SGX 하드웨어 구조

2.1. Intel SGX

Intel SGX 는 운영체제나 하이퍼바이저 등의 시스템 소프트웨어, 그리고 SGX 를 이용하기 위한 host application 으로부터 유저 레벨의 어플리케이션을 보호하기 위한 하드웨어 기능이다. Intel SGX 를 지원하는 기기에서는 enclave 라는 보호된 메모리 영역을 구성하고 어플리케이션이 이 enclave 메모리를 사용하도록 한다. <그림 1>과 같이 enclave 영역의 plaintext 는 CPU 만이 접근하여 사용할 수 있으며, DRAM 에 저장될 때는 MEE 라고 하는 암호화 엔진 하드웨어에 의해 암호화되어 저장된다. 이를 통해 CPU 이외의 모든 시스템 소프트웨어는 enclave 의 메모리 영역에 직접 접근해도 암호화된 값만 볼 수 있게 하여 유저 어플리케이션을 시스템 소프트웨어로부터 보호할 수 있다.

최근 SGX 는 SGX version 2 [5](이하 SGX2 로 기술함)을 발표하였는데, 이 기술은 enclave 메모리 영역의 크기를 동적으로 조정할 수 있도록 하는 기능이다. 새롭게 추가된 명령어인 EAUG, EACCEPT 명령어를 이용해 동적으로 메모리 사이즈를 확장할 수 있다. 자세히 설명하자면, enclave 내부에서 특정 메모리 주소를 인자로 하여 EACCEPT 명령어를 실행하면 page fault 가 발생하면서 커널이 EAUG 명령어를 실행하여 새로운 enclave page 를 할당하고, 다시 enclave 내부 코드가 EACCEPT 를 실행해 새롭게 할당한 page 를 accept 하는 작업을 함으로써 메모리 크기를 확장한다.



<그림 2> 소프트웨어 스택

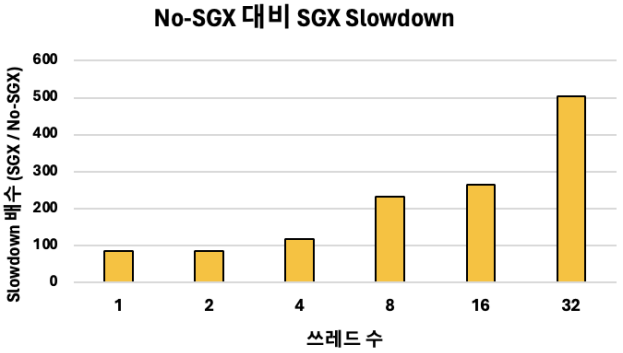
2.2 SGX library OS

<그림 2>에서 볼 수 있듯이 SGX library OS 는 enclave 메모리 내부에서 유저 어플리케이션을 보호하는 중간 레이어 소프트웨어이다. SGX library OS 가 없는 상황에서는 유저 어플리케이션에서 enclave 내부로 들어가게 하는 함수와 외부로 빠져나오게 하는 함수를 각각 전부 명시해야 하며, 그 과정에서 많은 코드 수정이 필요하다. SGX library OS 는 그런 과정이 필요 없이 원래의 어플리케이션을 library OS 위에서 그대로 실행시킬 수 있게 한다. 또한, enclave 외부의 시스템 소프트웨어로부터 악의적인 시스템 콜 등에 대한 validation 을 진행해 유저 어플리케이션을 보호해주기도 한다.

3. 문제점

자바스크립트를 이용한 서버 어플리케이션 중에는 유저의 데이터를 활용하는 어플리케이션이 많다. 대표적으로 웹 서버나 서버리스 함수 등은 유저의 아이디, 패스워드, 생체 정보, 파일 등이 입력값으로 사용된다. 이 때문에 이 정보가 클라우드로부터 보호될 수 있도록 SGX 로 해당 어플리케이션을 보호해야 할 필요가 있다. 이러한 어플리케이션들은 자바스크립트의 단일 쓰레드 모델에서의 성능 한계를 극복하기 위해 멀티쓰레드 방식을 많이 사용한다.

SGX library OS 로 가장 많이 쓰이는 프로그램 중 하나인 gramine 을 사용할 때, 자바스크립트 멀티쓰레드 어플리케이션을 실행하는 과정에서 굉장히 많은 시간이 소요되는 것을 확인했다. <그림 3>에서 보여지듯이 동시에 시작하는 쓰레드 수가 많을수록 각 쓰레드가 시작하는 데에 걸리는 시간이 느려졌고, SGX 를 사용하지 않고 실행시킬 때 걸리는 시간대비 느려지는 비율이 매우 증가하였다. 이에 대한 자세한 실험 결과는 <그림 4>에 실험 결과와 함께 표시했다.



<그림 3> 쓰레드 수에 따른 성능 오버헤드

SGX library OS 와 Node.js 에 대한 코드 분석을 통해 그 원인을 분석해본 결과 성능 오버헤드를 야기하는 지점을 찾을 수 있었다.

자세히 말하자면, Node.js 에서 Worker 를 통해 새로운 쓰레드를 만들 때 반복적으로 큰 메모리 영역을 할당 후 해제하는 작업이 진행되는 것이 확인되었다. 새로운 쓰레드를 만들 때 16-64 page 만큼의 메모리 영역을 반복적으로 할당 후 해제하는데, 새로운 쓰레드를 위한 heap 메모리를 할당하는 과정에서 각 메모리를 안전하게 사용하도록 permission 을 조정하는 데에 이러한 로직이 추가되어 있는 것으로 확인된다. 이는 SGX library OS 에서 동적 메모리 사이즈 변경이 단일 page 단위로만 진행된다는 점을 고려할 때 큰 오버헤드를 야기한다. 여러 page 사이즈만큼을 확장할 때 여러 번의 EAUG 와 EACCEPT 명령어 수행이 일어나며, 각 명령어 수행마다 enclave 내부에서 외부로, 그리고 다시 외부에서 내부로 control 이 바뀌는 과정이 일어나며 overhead 가 발생한다.

4. 해결 방안

3 에서 언급한 문제에 대한 해결 방안으로 다음과 같이 우리는 hybrid 방식의 메모리 할당이라는 해결

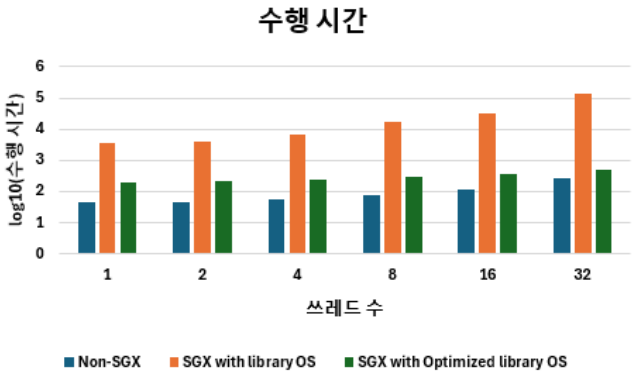
방안을 구현했다. 각 자바스크립트 쓰레드가 시작되며 heap 메모리가 할당될 때, 3 에서 언급한 반복적인 로직에 대한 힙 메모리 할당은 enclave 가 시작할 때 미리 할당한 메모리 영역에서 진행한다. 즉, 해당 영역에 대해서는 추가적으로 메모리 할당 및 해제가 필요 없게 만드는 것이다. 기존의 SGX library OS 는 enclave size 를 어플리케이션 시작 전에 정해놓고 그 사이즈만큼 미리 메모리를 전부 할당해놓거나, 혹은 최소한의 메모리만 할당 후 나머지 모든 메모리를 동적으로 할당 혹은 해제하는 방식을 사용했다. 우리는 이 두 가지의 중간 지점인 hybrid 방식을 통해 일부 메모리만 미리 할당한 후 나머지 메모리에 대해서는 동적 할당 및 해제를 할 수 있도록 했다. 이를 통해 성능 오버헤드를 줄이면서 메모리 사용량 또한 너무 높아지지 않게끔 만들었다.

5. 구현

4 에서 언급한 해결 방안을 구현하기 위해 gramine 이라는 대표적인 SGX library OS 를 수정하였다. 총 214 줄의 코드를 수정하였다.

6. 실험 결과

실험은 SGX2 가 가능한 Intel Xeon Gold 6348 CPU 머신에서 진행했다. 모든 실험은 160GB 의 메모리와 48GB 의 EPC 메모리가 할당된 QEMU VM 내부에서 진행되었다.
실험 결과는 아래 <그림 4>와 같다.



<그림 4> 수행 시간 비교

기존의 SGX library OS 를 이용해 멀티쓰레드 자바스크립트 어플리케이션을 수행할 때보다 약 84-498 배 정도의 성능 향상을 보일 수 있었다. 최적화한 library OS 를 통해 실행할 때는 SGX 없이 실행할 때보다 약 10 배 느린 수치를 보였다. 미리 메모리를 할당한 것이 기존보다 훨씬 좋은 성능을 이끌어 낸 것이라고 이해할 수 있다. 미리 할당하는 메모리의 크기는 쓰레드 하나당 약 400MB 정도가 필요한 것으로 확인되었다.

7. 관련 연구

멀티쓰레드 환경에서 SGX 어플리케이션의 성능 향상을 위한 연구는 SGX 가 발표된 이후로 꾸준히 이어져왔다. SCONE[6]은 각각의 enclave 쓰레드의 시스템 콜로 인한 오버헤드를 줄이기 위해 asynchronous 시스템 콜 메커니즘을 이용해 enclave 바깥의 쓰레드가 enclave 쓰레드의 시스템 콜을 처리하고 그 결과를 dynamic 하게 전달하는 메커니즘을 구현했다. Occlum[7]은 멀티프로세스 어플리케이션을 enclave 내부에서 멀티쓰레딩처럼 처리하여 프로세스 생성 오버헤드를 없애는 library OS 를 구현했다.

8. 결론

기존의 SGX 의 library OS 는 멀티쓰레드 자바스크립트 어플리케이션을 수행하는 데에 있어서 성능 저하를 야기하는 문제가 있었다. 이 논문에서는 hybrid 방식으로 미리 메모리를 일부 할당하는 것을 통해 성능 향상을 이끌어낼 수 있음을 보인다.

Acknowledgement

이 논문은 2024 년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No.2020-0-01840, 스마트폰의 내부데이터 접근 및 보호 기술 분석)

참고문헌

[1] Node.js. https://github.com/nodejs
[2] Node.js Worker threads.
https://nodejs.org/api/worker_threads.html
[3] Victor Costan and Srinivas Devadas. Intel sgx explained. Cryptology ePrint Archive, 2016.
[4] Chia-Che Tsai, Donald E Porter, and Mona Vij. Graphene-sgx: A practical library os for unmodified applications on sgx. In USENIX Annual Technical Conference, Santa Clara, 2017. p. 645–658,.
[5] Frank McKeen, Ilya Alexandrovich, Ittai Anati. Dror Caspi, Simon Johnson, Rebekah Leslie-Hurd, and Carlos Rozas. Intel Software Guard Extensions Support for Dynamic Memory Management Inside an Enclave. In HASP, New York, 2016.
[6] Sergei Arnautov, Bohdan Trach, Franz Gregor, Thomas Knauth, and Andre Martin, Christian Priebe, Joshua Lind, Divya Muthukumaran, Dan O'Keeffe, and Mark L Stillwell, David Goltzsche, Dave Eyers, Rüdiger Kapitza, Peter Pietzuch, Christof Fetzer. SCONE: Secure Linux Containers with Intel SGX. In Proceedings of the 12trh USENIX Symposium on Operating System Design and Implementation, Savannah, 2016.
[7] Youren Shen, Hongliang Tian, Yu Chen, Kang Chen, Runji Wang, Yi Xu, Yubin Xia, Shoumeng Yan. Occlum: Secure and Efficient Multitasking Inside a Single Enclave of Intel SGX. In International Conference on Architectural Support for Programming Languages and Operating Systems, Lausanne, 2020.