

현대엔지비 특강

Rust는 왜 주목 받고 있는가

Chris Ohk

utilForever@gmail.com

발표자 소개

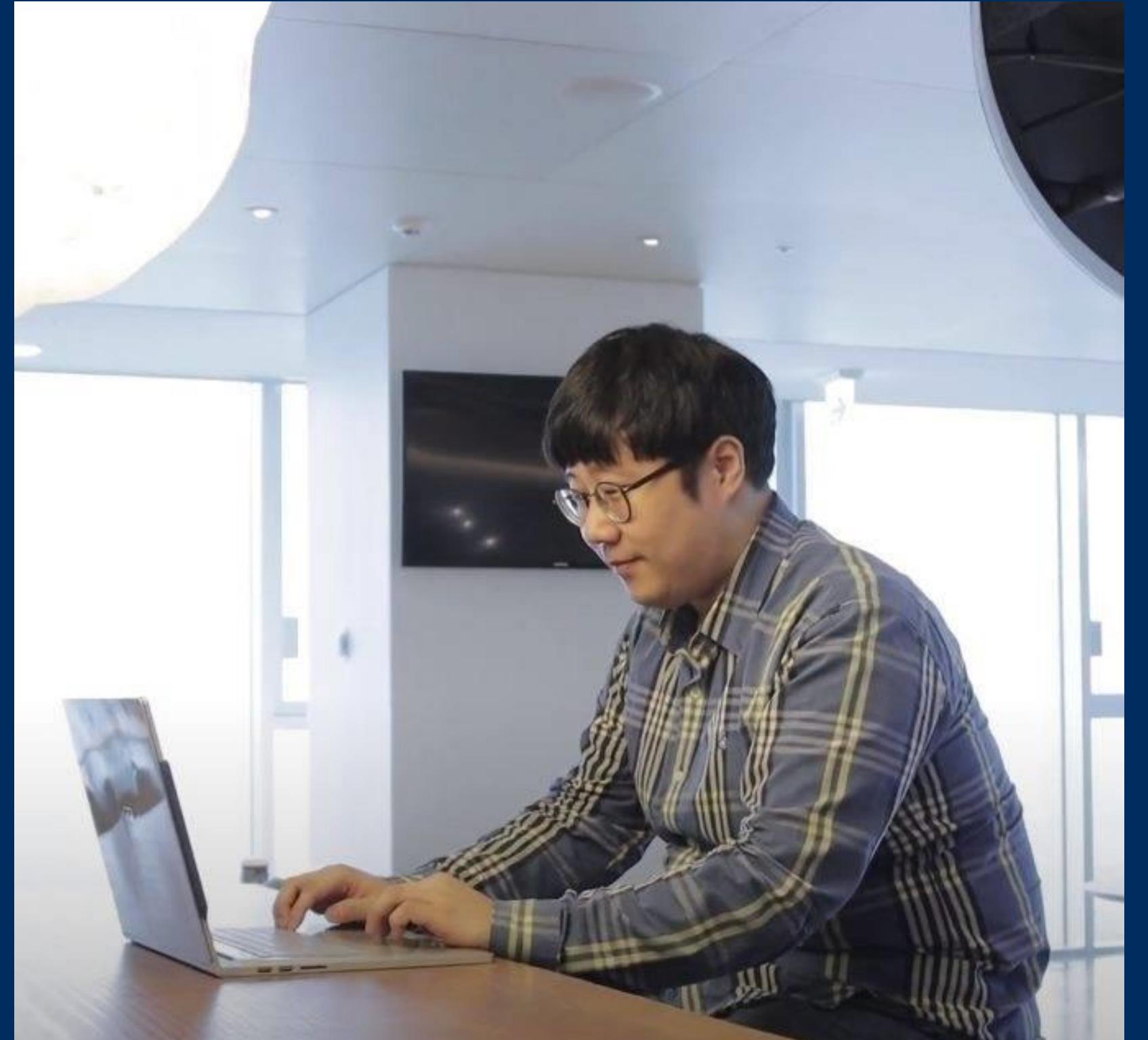
현대엔지비 특강
Rust는 왜 주목 받고 있는가

- 옥찬호 (Chris Ohk)
- (현) EJN Tech Lead
- (전) Momenti Engine Engineer
- (전) Nexon Korea Game Programmer
- Microsoft Developer Technologies MVP
- C++ Korea Founder & Administrator
- Reinforcement Learning KR Administrator
- IT 전문서 집필 및 번역 다수
- 대학교 Rust 특강 및 강의 다수

utilForever@gmail.com



utilForever



목차

현대엔지비 특강
Rust는 왜 주목 받고 있는가

- 들어가며
- Rust 소개 및 특징
- Rust는 왜 주목 받고 있는가
- Rust 사용 사례

들어가며

현대엔지비 특강
Rust는 왜 주목 받고 있는가

- Rust의 현재



- Rust의 현재

Linux 6.1 Officially Adds Support for Rust in the Kernel

LIKEDISCUSSPRINTBOOKMARK

DEC 20, 2022 • 1 MIN READ

by  Sergio De Simone FOLLOW

Write for InfoQ

Join a community of experts.

Increase your visibility.
Grow your career.
[Learn more](#)

After over two years in development, support for using Rust for kernel development has [entered a stable Linux release](#), Linux 6.1, which became [available](#) a couple of weeks ago. Previous to its official release, Rust support has been available in linux-next, the git tree resulting from merging all of the developers and maintainers trees, for over a year. With the stable release, Rust has become the second language officially accepted for Linux kernel development, along with C.

Initial Rust support is just the absolute minimum to get Rust code building in the kernel, say Rust for Linux maintainers. This possibly means that Rust support is not ready yet for prime-time development and that a number of changes at the infrastructure level are to be expected in coming releases. Still, there has been quite some work going on on a few actual drivers that should become available in the next future. These include a Rust [nvme driver](#), a [9p server](#), and [Apple Silicon GPU drivers](#).

Rust for Linux is only available on the architectures supported by LLVM/Clang (<https://github.com/Rust> for Linux/linux/blob/rust/Documentation/rust/arch-support.rst), which is required to compile Rust. Thus, LLVM/Clang must be used to build the whole Linux kernel instead of the more traditional GNU toolchain. This limits supported architecture to a handful, including arm, arm64, x86, powerpc, mips, and others. For detailed instructions about building Linux with the appropriate flag for each supported platform, check the official documentation (<https://github.com/Rust> for Linux/linux/blob/rust/Documentation/kbuild/llvm.rst).

- Rust의 현재

바이든 행정부가 버퍼 오버플로 및 기타 메모리 액세스 취약성을 유발하는 프로그래밍 언어에서 벗어날 것을 촉구했다.



Image Credit : Getty Images Bank

소프트웨어 개발자들이 C와 C++와 같은 취약한 프로그래밍 언어의 사용을 중단하라고 조 바이든 미국 대통령의 행정부가 촉구했다. 대신 메모리 안전 프로그래밍 언어(memory-safe programming languages)를 사용하라는 권고다.

백악관 국가사이버책임자실(ONCD)은 [26일 발표한 보고서](#)에서 개발자들에게 메모리 안전 취약점이 없는 프로그래밍 언어를 사용하여 사이버 공격의 위험을 줄여야 한다고 주문했다. 백악관은 보도자료를 통해 기술 기업들이 메모리 안전 프로그래밍 언어를 채택함으로써 "모든 종류의 취약점이 디지털 생태계에 유입되는 것을 방지할 수 있다"라고 밝혔다.

메모리 안전 프로그래밍 언어는 버퍼 오버플로, 범위를 벗어난 읽기, 메모리 누수 등 메모리 액세스와 관련된 소프트웨어 버그 및 취약성으로부터 안전한 프로그래밍 언어를 의미한다. 마이크로소프트와 구글의 최근 연구에 따르면 전체 보안 취약점의 약 70%가 메모리 안전 문제로 인해 발생하는 것으로 나타났다.

Rust란

현대엔지비 특강
Rust는 왜 주목 받고 있는가

- 모질라 재단에서 2010년 7월 7일 처음 발표
- 2015년 5월 15일 Stable 1.0 정식 발표
- 2021년 2월부터는 Rust 재단으로 분리



Rust 공식 로고



Rust 비공식 마스코트 “Ferris”

Rust 언어의 특징

현대엔지비 특강
Rust는 왜 주목 받고 있는가

- 안전한 메모리 관리
- 철저한 예외와 오류 관리
- 특이한 열거 시스템
- 트레잇
- 하이지닉 매크로
- 비동기 프로그래밍
- 제네릭

Rust는 왜 주목 받고 있는가

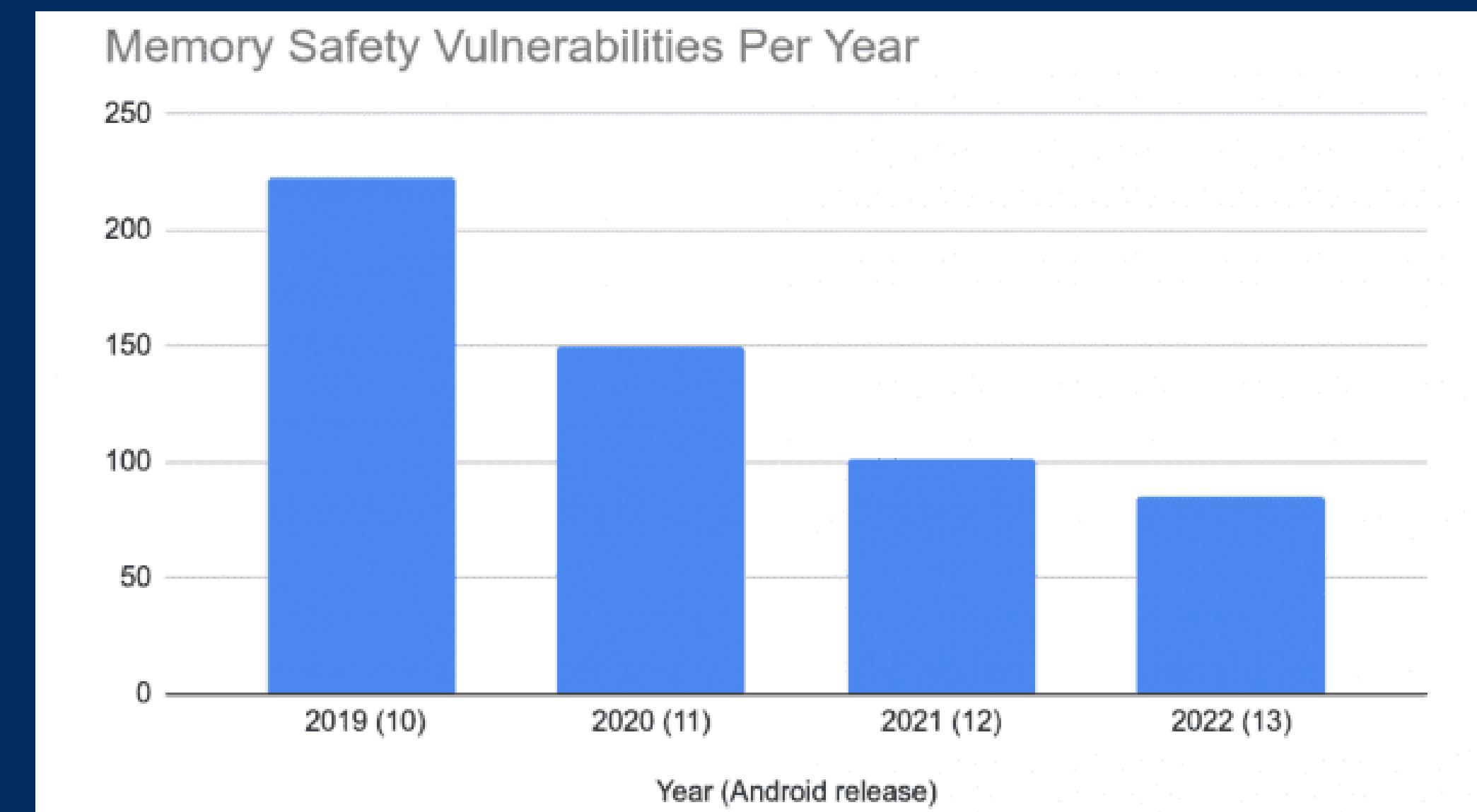
현대엔지비 특강
Rust는 왜 주목 받고 있는가

- 메모리 안전성
- 성능
- 동시성
- 웹어셈블리
- 지속 가능성

메모리 안전성 - 안드로이드 OS

현대엔지비 특강
Rust는 왜 주목 받고 있는가

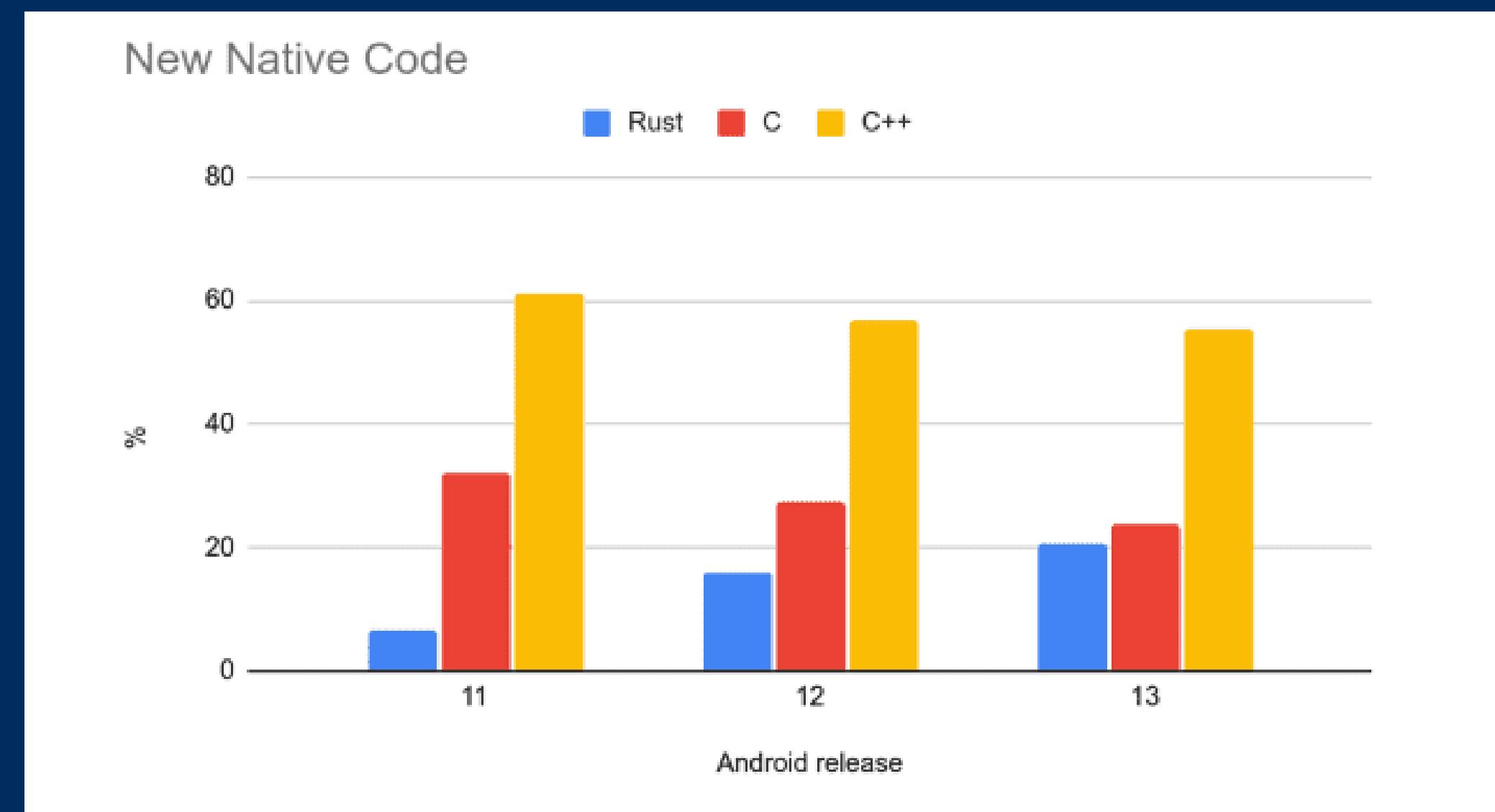
- 기존 안드로이드 OS의 전체 취약점 중 메모리 안전 취약점이 76%를 차지
- 구글은 2019년부터 Rust를 안드로이드 OS에 통합해오면서 4년의 작업을 통해 안드로이드 OS의 보안 취약성을 줄이고 성능을 개선하고 있음
- 그 결과 76% 비중을 차지하던 메모리 안전 취약점은 35%로 떨어지게 됨



메모리 안전성 - 안드로이드 OS

현대엔지비 특강
Rust는 왜 주목 받고 있는가

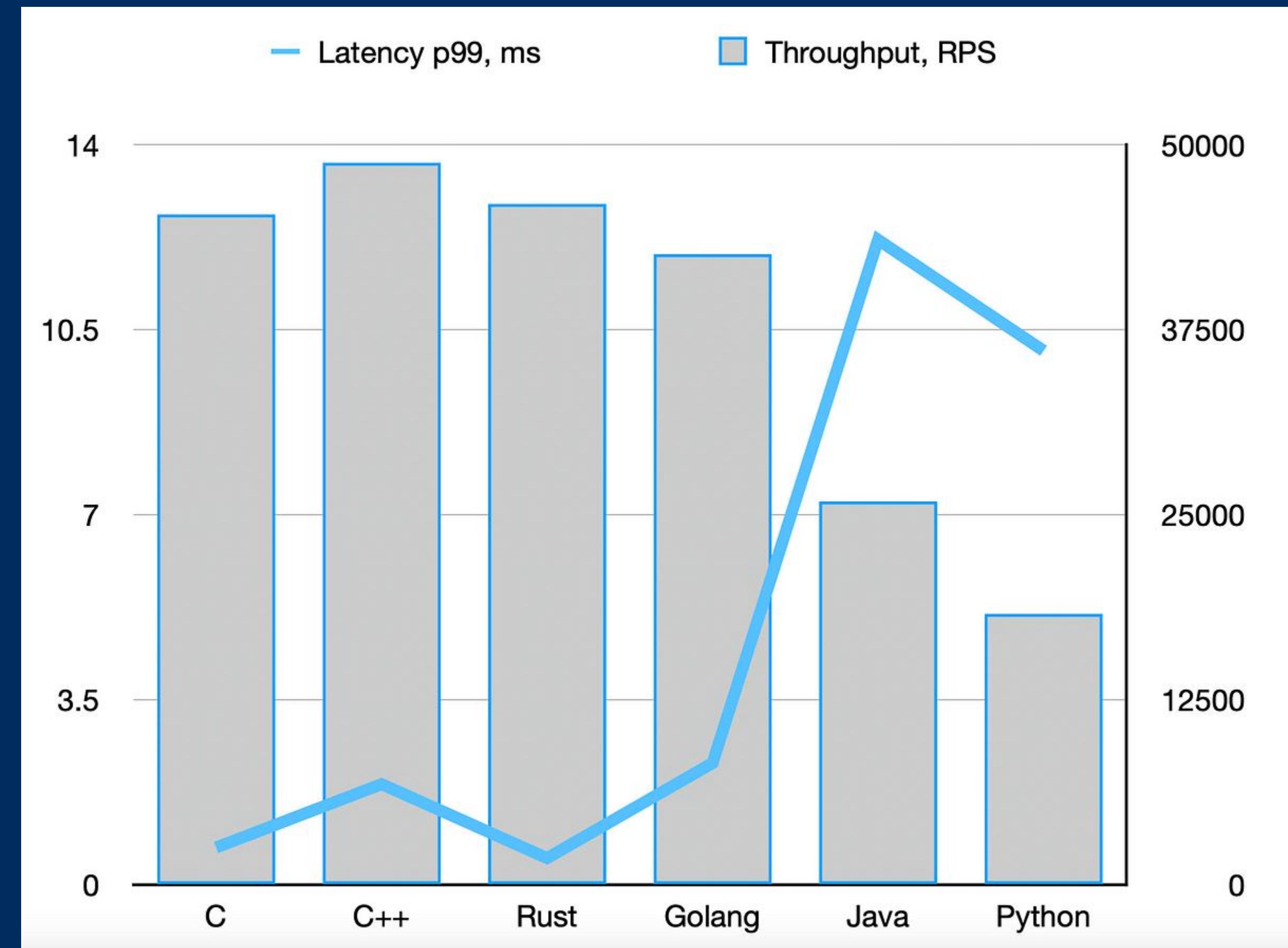
- 기존 C/C++ 언어를 Rust로 대체하는 대신, 신규 코드를 Rust로 작성하는 식으로 대체
- 그 결과, 안드로이드의 네이티브 코드에서 C/C++의 비율은 아주 조금씩 감소 추세고, Rust는 급증하고 있다고 함 (안드로이드 13 기준으로 새로운 네이티브 코드 중 21% 차지)
- 안드로이드의 Rust 코드에서 메모리 보안 취약점이 전혀 발견되지 않았다고 밝힘



성능 - 로우 레벨 I/O 벤치마크

현대엔지비 특강
Rust는 왜 주목 받고 있는가

- C, C++, Rust, Golang, Java, Python



동시성 - 겁이 없음

현대엔지비 특강
Rust는 왜 주목 받고 있는가

- 동시성 프로그래밍을 할 때 가장 걱정하는 것들
 - 경쟁 조건(Race Condition)
 - 데드락(Deadlock)
 - 재현하기 힘들고 수정하기 힘든 버그
- Rust는 소유권(Ownership) 시스템과 빌림(Borrowing) 검사기를 통해 동시성 프로그래밍에서 발생할 수 있는 문제들을 미리 감지함
(런타임 에러가 컴파일 타임 에러가 됨)
- 그 결과, 안전한 동시성 패턴을 제공해 멀티코어 시스템에서의 프로그래밍을 간소화함

웹어셈블리 - 시너지 효과

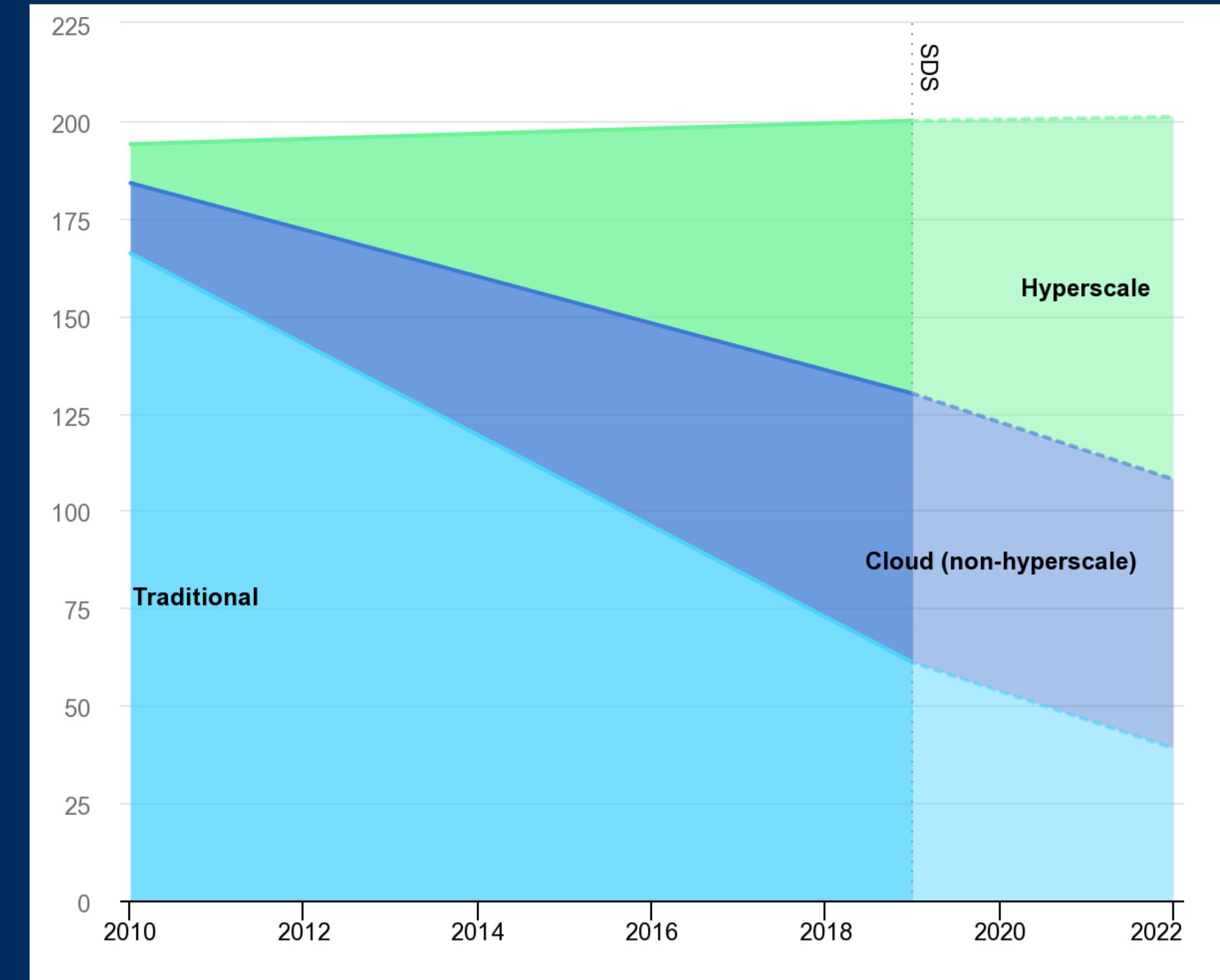
현대엔지비 특강
Rust는 왜 주목 받고 있는가

- Rust의 장점 : 메모리 안전성, 성능, 동시성, 풍부한 생태계
- 웹어셈블리의 장점 : 성능, 보안, 이식성, 상호 운용성
- Rust와 웹어셈블리의 시너지 효과
 - 최적화된 성능 : Rust는 비용이 들지 않는 추상화에 중점을 두어 WebAssembly의 모듈을 간결하고 빠르게 만듬
 - 안전성 및 보안 : Rust와 WebAssembly 모두 안전한 실행을 우선시해 웹 애플리케이션의 안전성을 높임
 - 통합 용이성 : Rust는 WebAssembly에 대한 최고 수준의 지원을 제공하므로 Rust 코드를 wasm으로 컴파일하고 이를 웹 애플리케이션에 통합하는 것이 간단함

지속 가능성 - 에너지 사용 in 클라우드

현대엔지비 특강
Rust는 왜 주목 받고 있는가

- 전세계적으로 연간 약 200 테라와트의 전력을 소비
(지구에서 소비되는 모든 에너지의 약 1%를 차지)
- 기존 데이터 센터의 비율은 줄고,
클라우드/하이퍼스케일 데이터 센터의 비율은 증가
- 하지만 전체 에너지 사용량은 거의 비슷
- 이는 클라우드/하이퍼스케일 데이터 센터의
에너지 효율성 개선을 꾸준히 한 결과



지속 가능성 - 에너지 효율성

현대엔지비 특강
Rust는 왜 주목 받고 있는가

- 여러 프로그래밍 언어들의 에너지 소비, 성능, 메모리 사용 사이의 상관관계를 조사
- C와 Rust가 다른 언어보다 더 효율적이라는 결과

Table 4. Normalized global results for Energy, Time, and Memory

| Total | | | | |
|----------------|--------|-------|------|-----------------|
| | Energy | Time | Mb | |
| (c) C | 1.00 | 1.00 | 1.00 | |
| (c) Rust | 1.03 | 1.04 | 1.05 | |
| (c) C++ | 1.34 | 1.56 | 1.17 | |
| (c) Ada | 1.70 | 1.85 | 1.24 | |
| (v) Java | 1.98 | 1.89 | 1.34 | |
| (c) Pascal | 2.14 | 2.14 | 1.47 | |
| (c) Chapel | 2.18 | 2.83 | 1.54 | |
| (v) Lisp | 2.27 | 3.02 | 1.92 | |
| (c) Ocaml | 2.40 | 3.09 | 2.45 | |
| (c) Fortran | 2.52 | 3.14 | 2.57 | |
| (c) Swift | 2.79 | 3.40 | 2.71 | |
| (c) Haskell | 3.10 | 3.55 | 2.80 | |
| (v) C# | 3.14 | 4.20 | 2.82 | |
| (c) Go | 3.23 | 4.20 | 2.85 | |
| (i) Dart | 3.83 | 6.30 | 3.34 | |
| (v) F# | 4.13 | 6.52 | 3.52 | |
| (i) JavaScript | 4.45 | 6.67 | 3.97 | |
| (v) Racket | 7.91 | 11.27 | 4.00 | |
| (i) TypeScript | 21.50 | 26.99 | 4.25 | |
| (i) Hack | 24.02 | 27.64 | 4.59 | |
| (i) PHP | 29.30 | 36.71 | 4.69 | |
| (v) Erlang | 42.23 | 43.44 | 4.69 | |
| (i) Lua | 45.98 | 46.20 | 6.01 | |
| (i) Jruby | 46.54 | 59.34 | 6.62 | |
| (i) Ruby | 69.91 | 65.79 | 6.72 | |
| (i) Python | 75.88 | 71.90 | 7.20 | |
| (i) Perl | 79.58 | 82.91 | 8.64 | |
| | | | | (i) Jruby 19.84 |

Rust 사용 사례

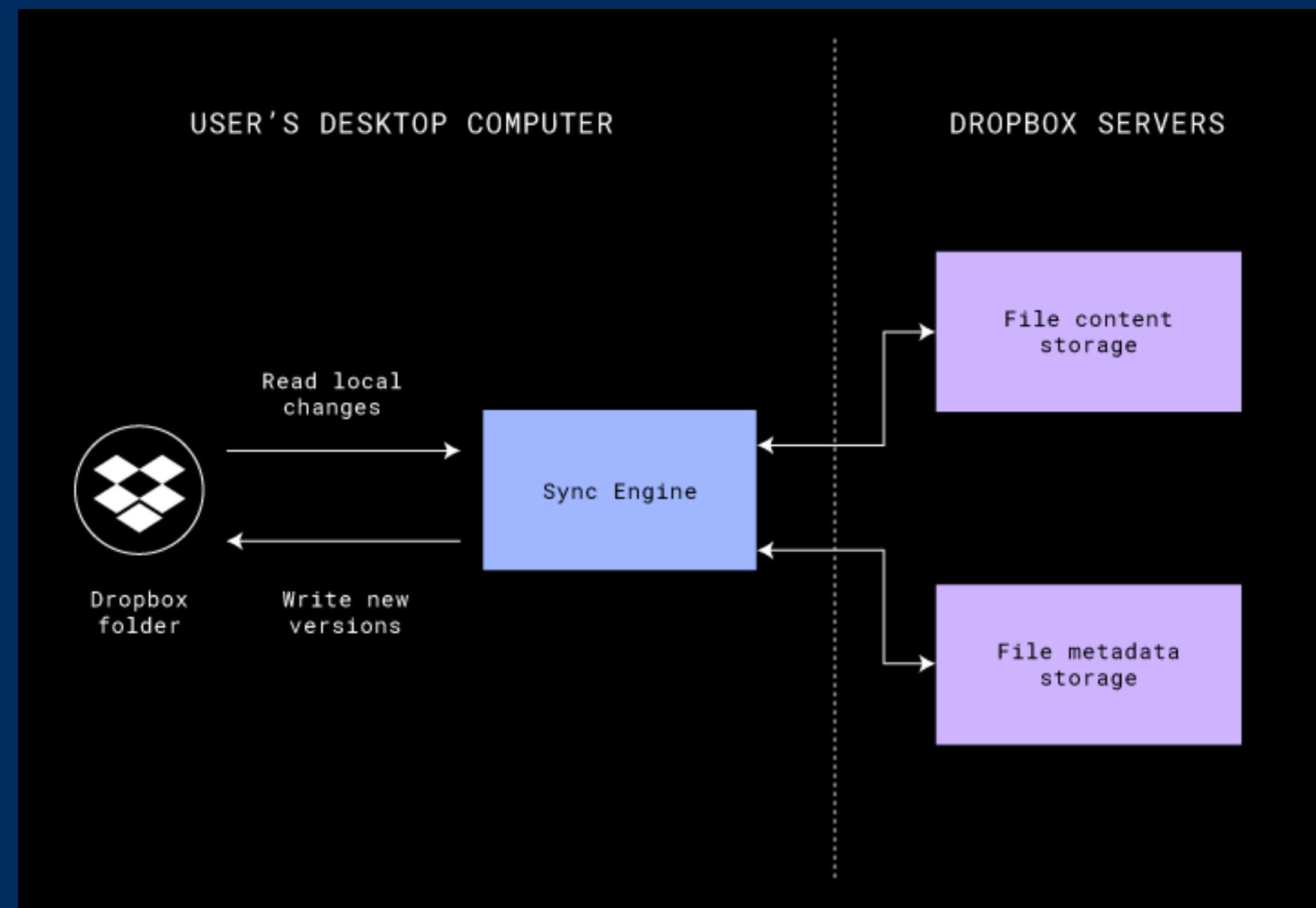
현대엔지비 특강
Rust는 왜 주목 받고 있는가

- 드롭박스
- 피그마
- 디스코드
- Tenable
- Volvo
- 카카오
- 그 외

드롭박스

현대엔지비 특강
Rust는 왜 주목 받고 있는가

- 클라우드 저장소 서비스를 운영하는 드롭박스의 가장 핵심 기능 중 하나는 로컬 컴퓨터에 있는 데이터를 원격 클라우드에 빠르게 동기화하는 것
- C++로 작성되어 있었던 동기화 로직이 시간이 지남에 따라 문제가 많이 생김
- 동기화 로직을 Rust로 재작성했고, 만족할만한 성과를 거둠

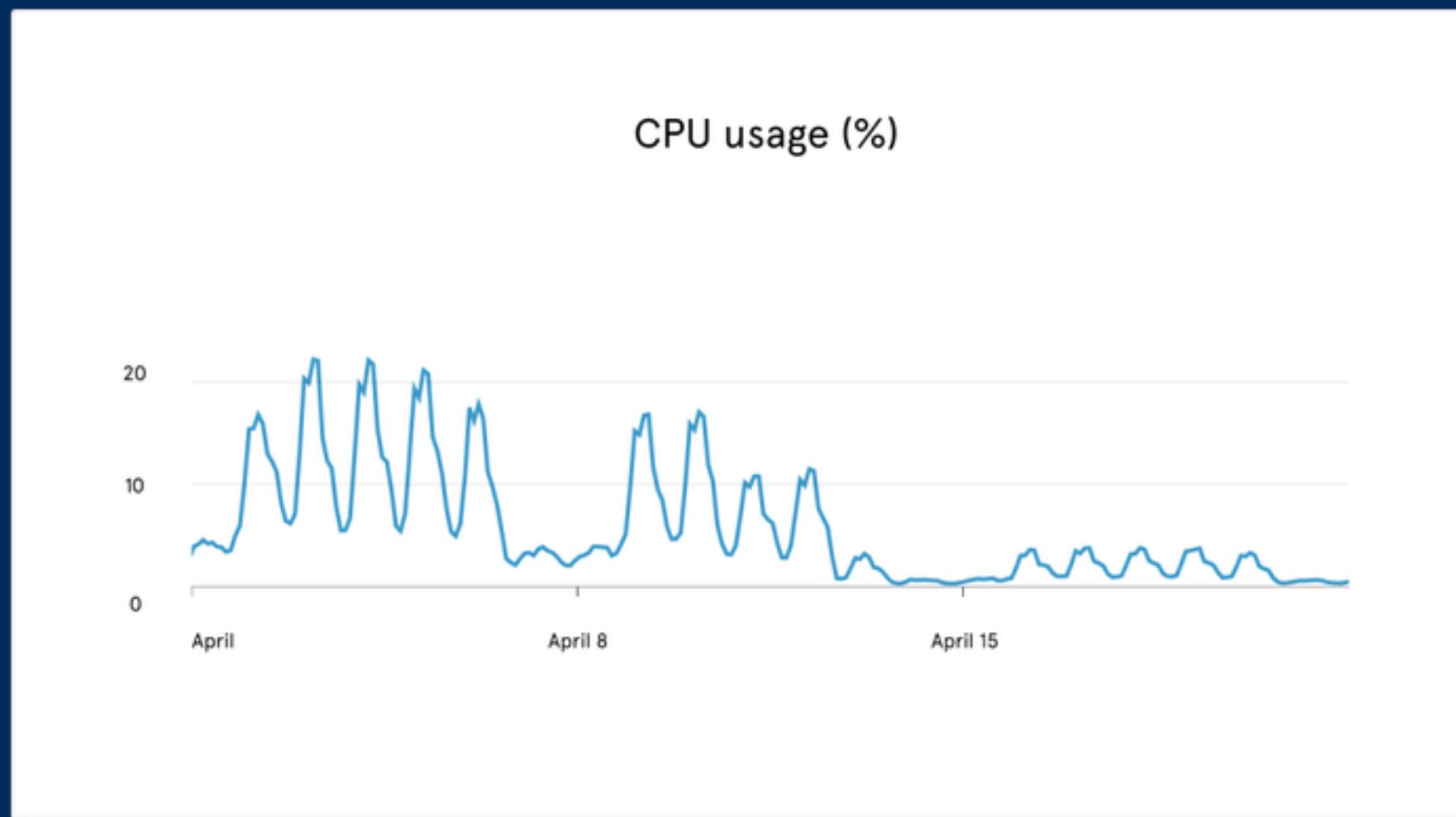
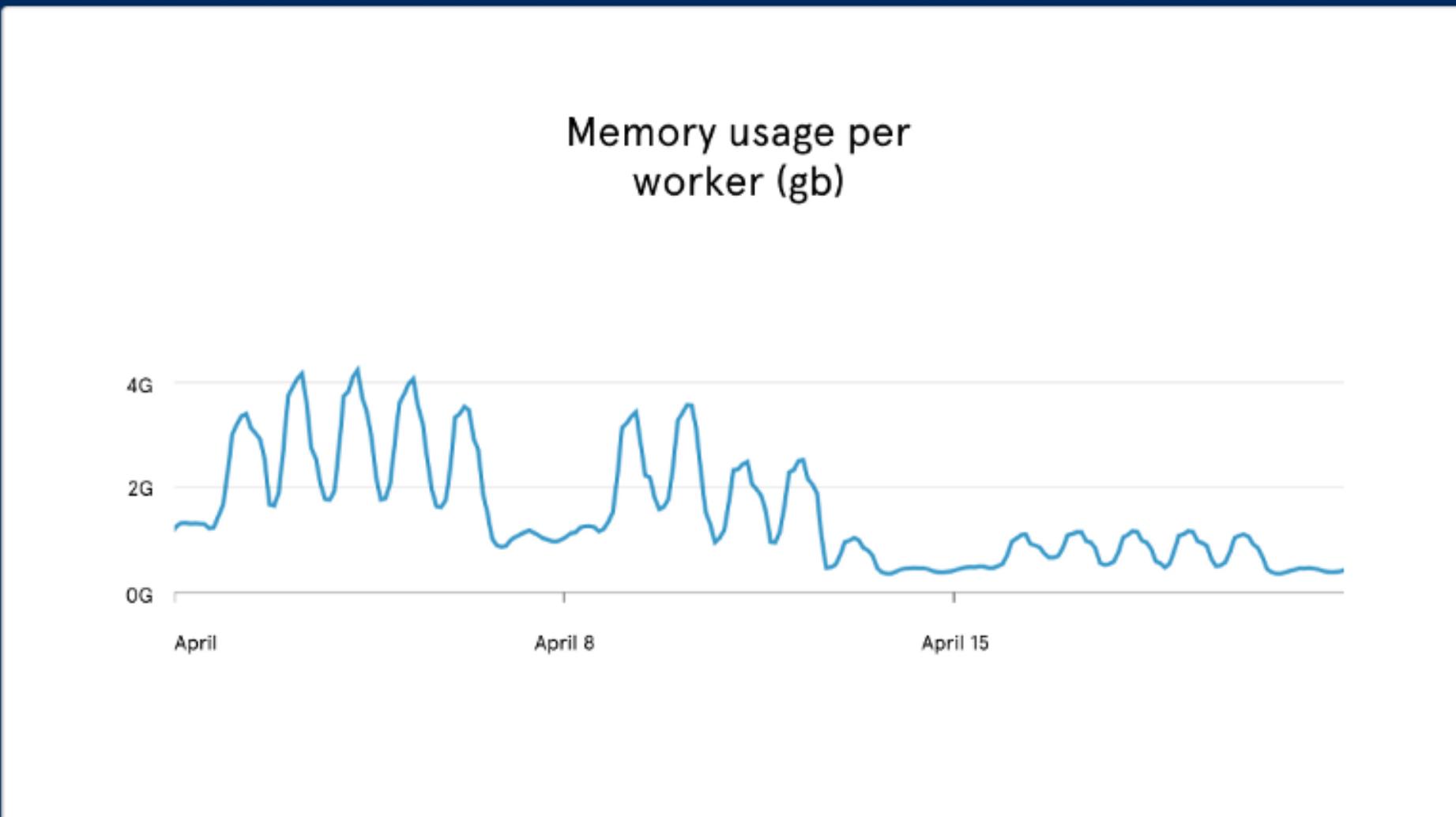


- 피그마는 UI 프로토타입을 제작할 수 있는 도구로 웹 기반으로 동작 따라서 화면에 결과를 빠르게 보여주는 것이 중요
- 기존 서버는 TypeScript로 작성되었는데 동기화 중 예측할 수 없는 자연 시간 스파이크 문제로 고생을 하고 있었음
- 싱글 스레드로 동작하고 있어서 동작들을 병렬로 처리할 수 없었음
- 이 문제를 해결하기 위해 Rust로 재작성하기로 결정

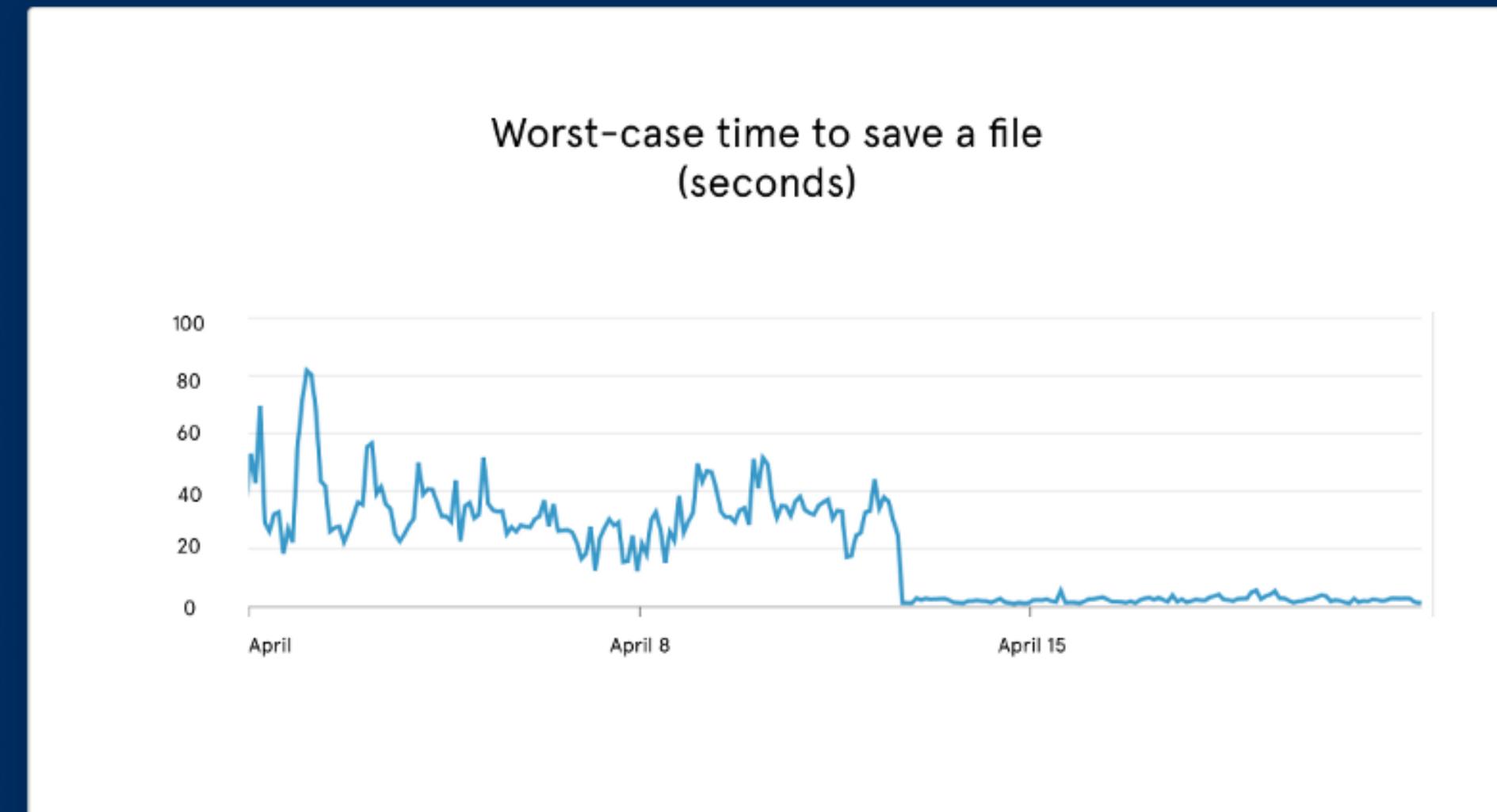
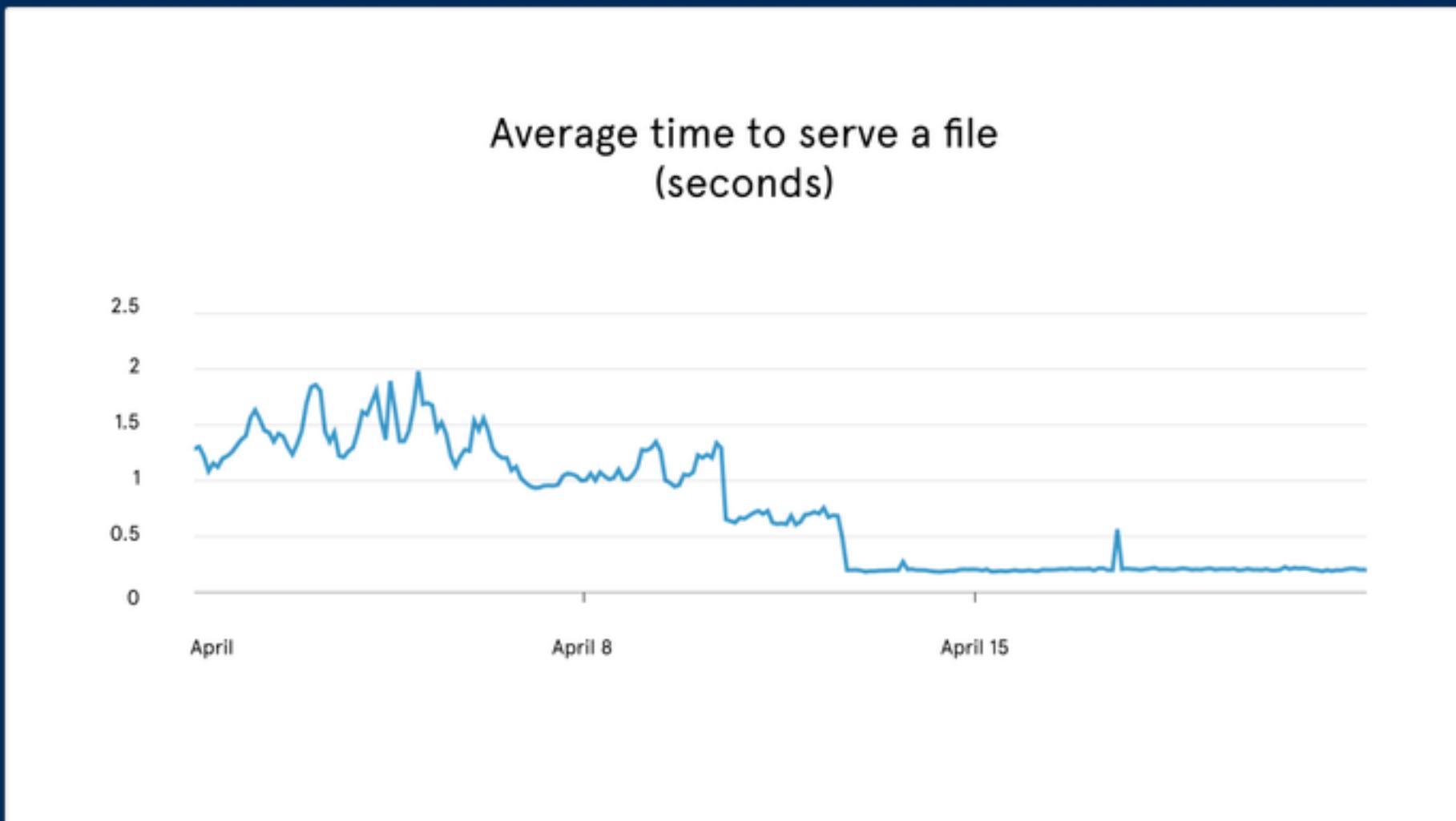
피그마

현대엔지비 특강
Rust는 왜 주목 받고 있는가

- 그 결과 이전 서버와 비교하여 지표들이 전반적으로 개선됨



- 그 결과 이전 서버와 비교하여 지표들이 전반적으로 개선됨



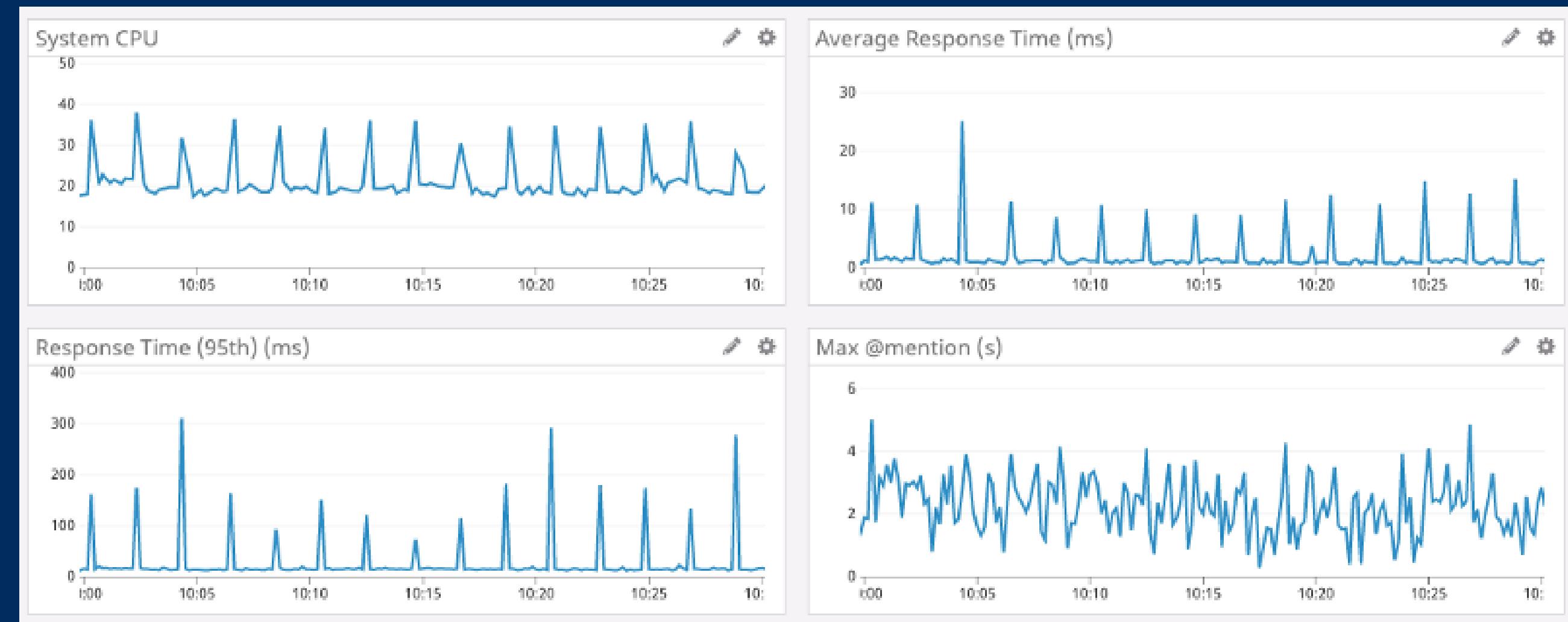
- 그 결과 이전 서버와 비교하여 지표들이 전반적으로 개선됨

| Metric | Old server | New server | Improvement |
|--------------------------------------|------------|------------|--------------|
| Peak average per-worker memory usage | 4.2gb | → 1.1gb | 3.8x smaller |
| Peak average per-machine CPU usage | 24% | → 4% | 6x smaller |
| Peak average file serve time | 2s | → 0.2s | 10x faster |
| Peak worst-case save time | 82s | → 5s | 16.4x faster |

디스코드

현대엔지비 특강
Rust는 왜 주목 받고 있는가

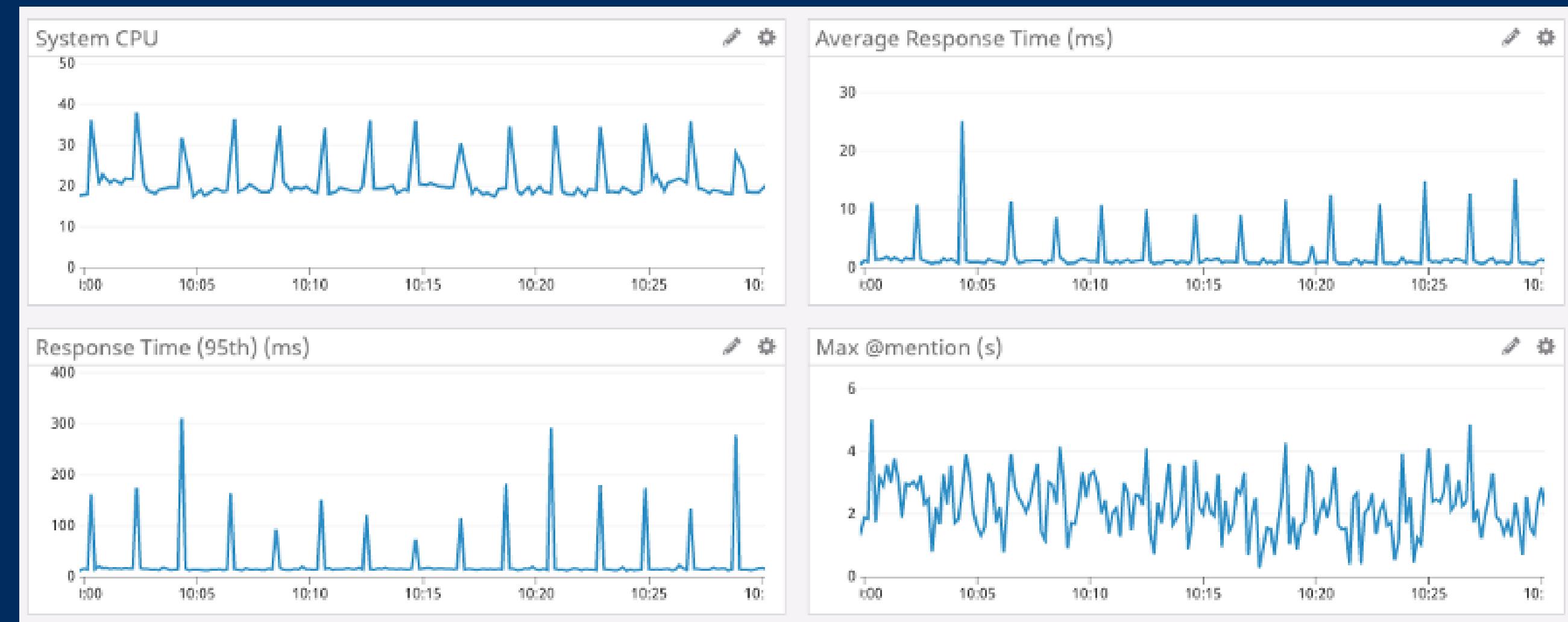
- Read State 서비스는 사용자가 읽은 채널과 메시지를 추적, Go로 구현
- 리소스 모니터를 관찰한 결과, 2분마다 대기 시간과 CPU 스파이크가 발생하는 걸 발견



디스코드

현대엔지비 특강
Rust는 왜 주목 받고 있는가

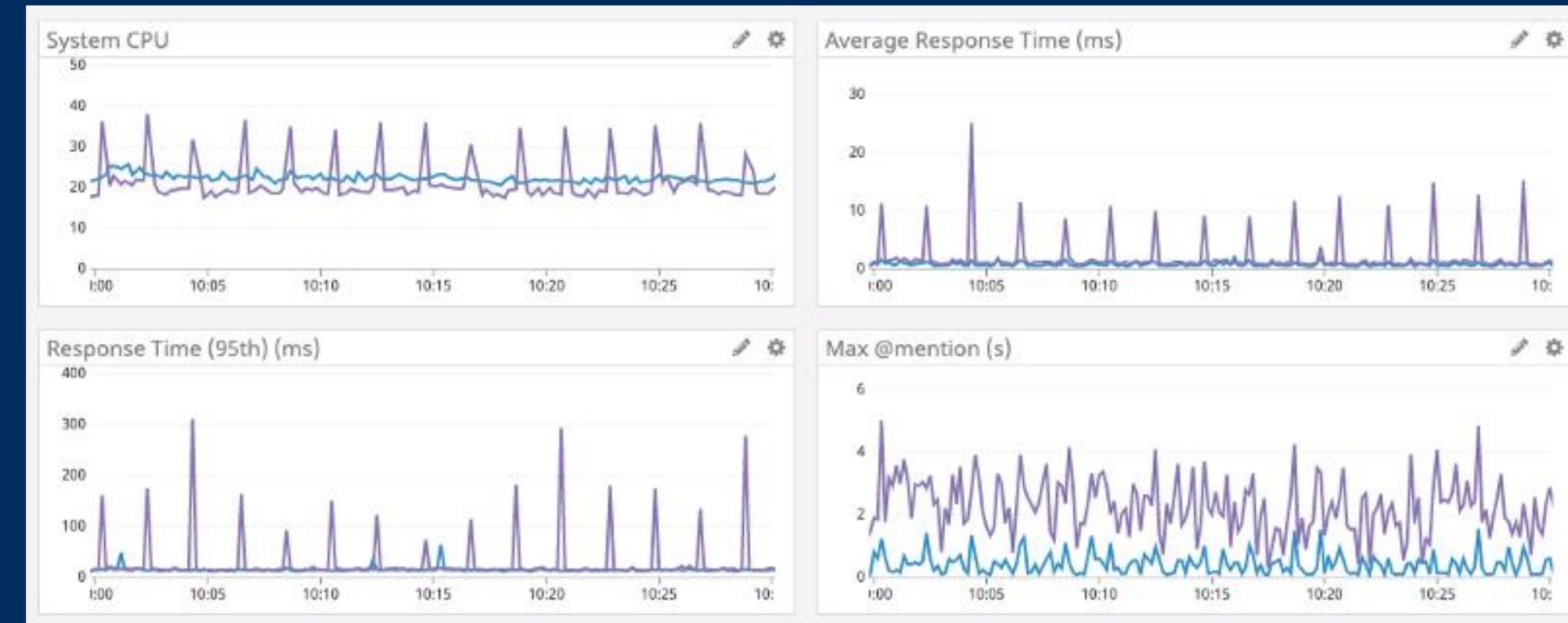
- 원인을 분석한 결과, Go가 최소 2분마다 가비지 컬렉션을 강제로 실행한다는 걸 알게 됨
- LRU 캐시가 작을수록 가비지 컬렉션으로 인한 스파이크가 더 작아진다는 사실을 알아냈지만, 트레이드오프로 인해 대기 시간이 더 길어짐 (캐시 히트율이 줄어들기 때문)



디스코드

현대엔지비 특강
Rust는 왜 주목 받고 있는가

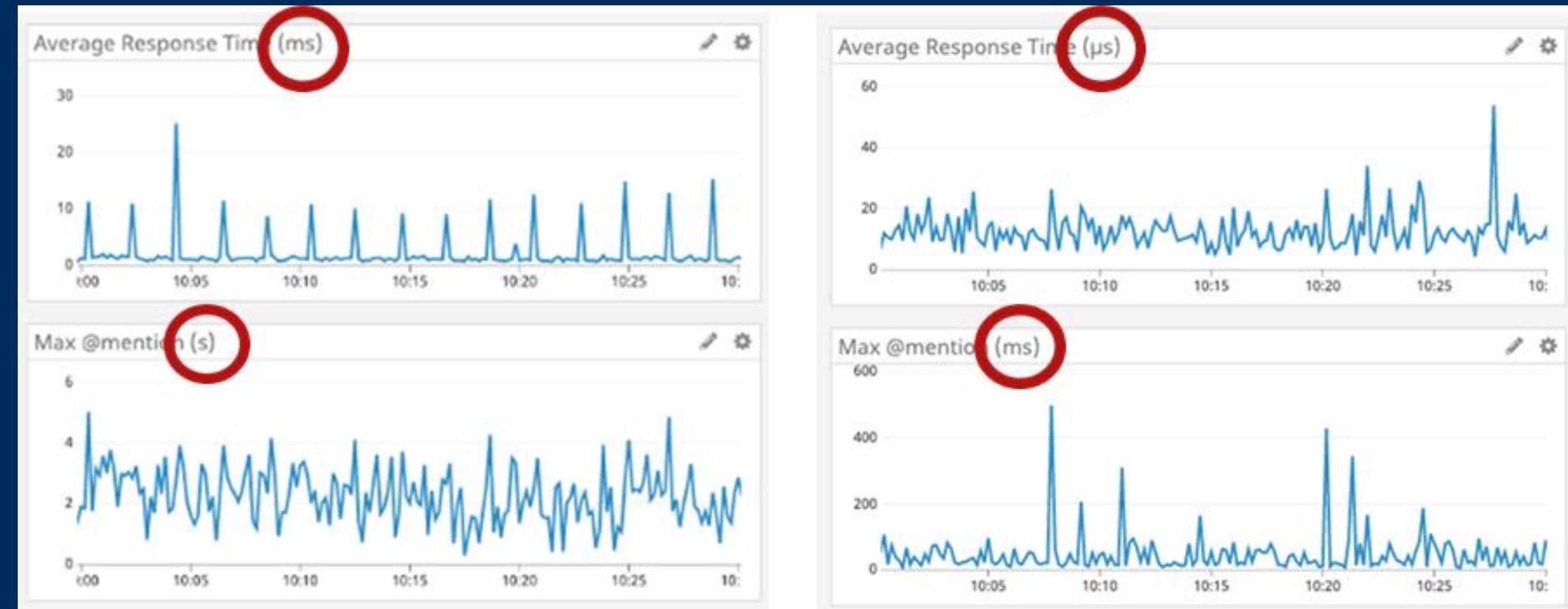
- 문제를 해결하기 위해 Read State 서비스를 Go에서 Rust로 전환
 - Go = 보라색
 - Rust = 파란색



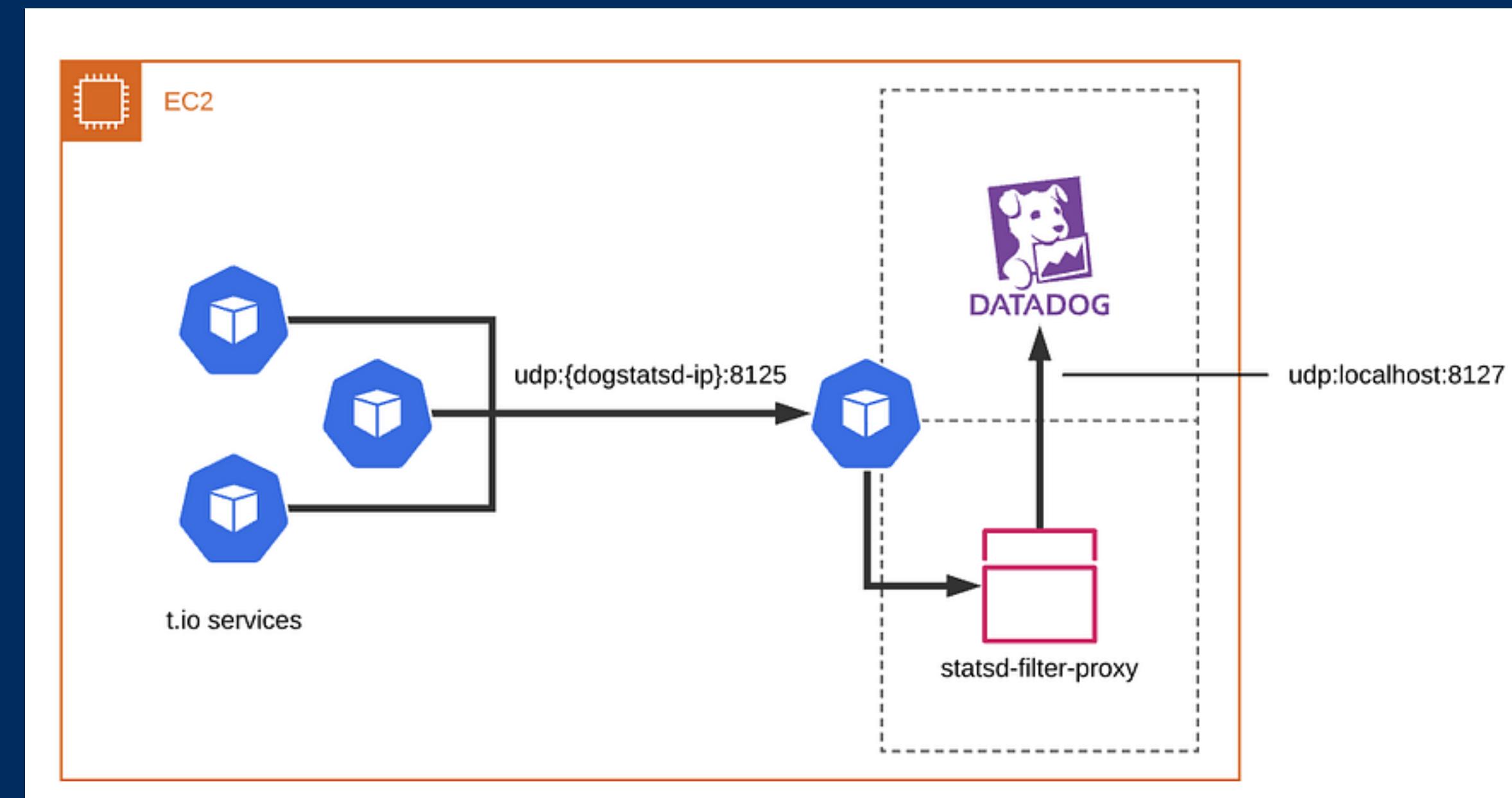
디스코드

현대엔지비 특강
Rust는 왜 주목 받고 있는가

- 문제를 해결하기 위해 Read State 서비스를 Go에서 Rust로 전환
 - Go = 왼쪽
 - Rust = 오른쪽



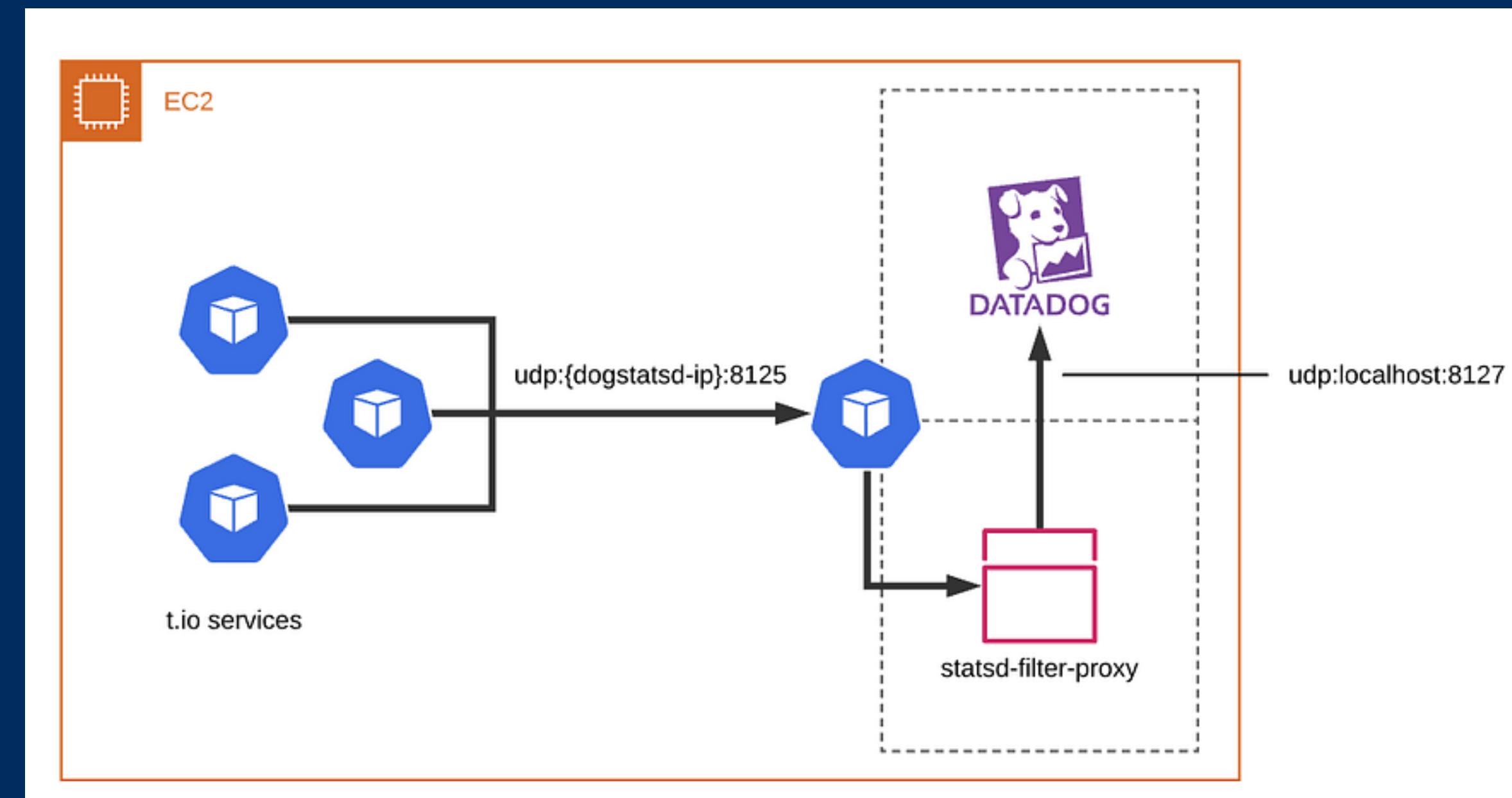
- Datadog의 커스텀 메트릭을 사용하기 위해 수백만 개의 지표를 전송
- 플랫폼이 성장함에 따라 레거시 앱에서 전송한 수많은 메트릭들이 더이상 사용되지 않는다는 사실을 발견
- 이 문제를 해결하기 위해 불필요한 측정 항목을 필터링하기로 함



Tenable

현대엔지비 특강
Rust는 왜 주목 받고 있는가

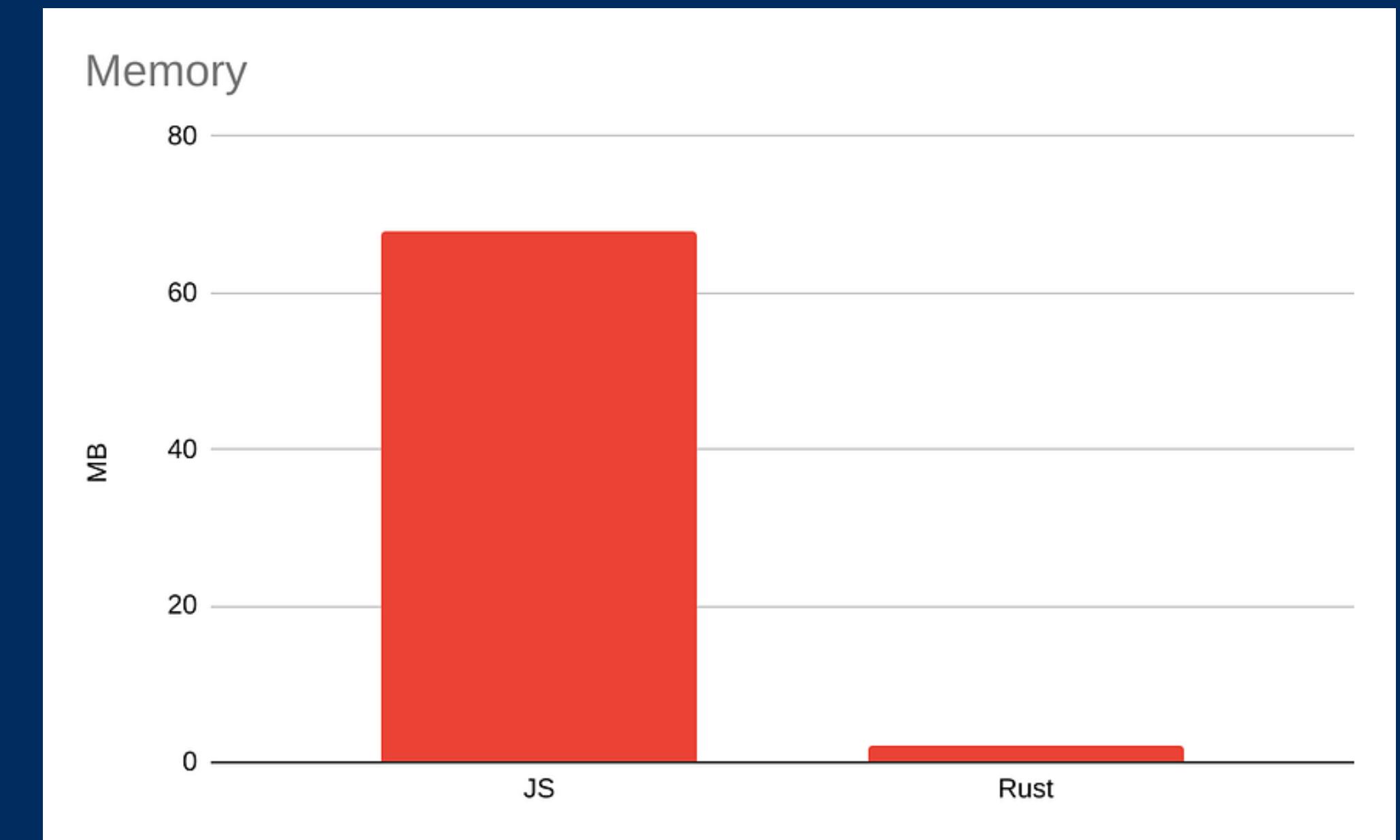
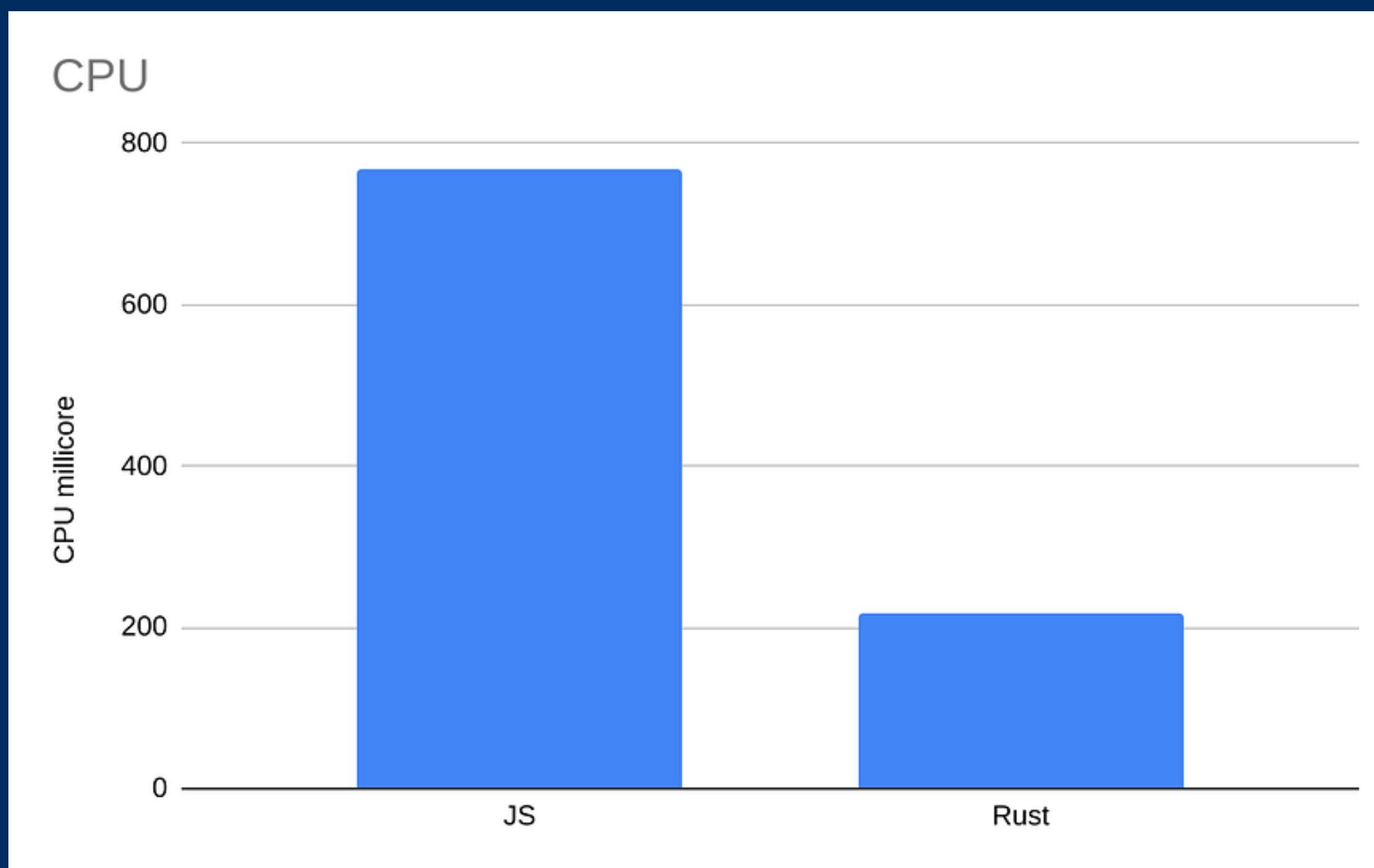
- 필터는 Node.js로 작성되었는데,
처음에는 괜찮았지만 시간이 지나면서 성능 문제가 발생하기 시작
- 성능 지표를 분석하고 필터를 보다 효율적인 언어, 즉 Rust로 다시 작성하기로 결정



Tenable

현대엔지비 특강
Rust는 왜 주목 받고 있는가

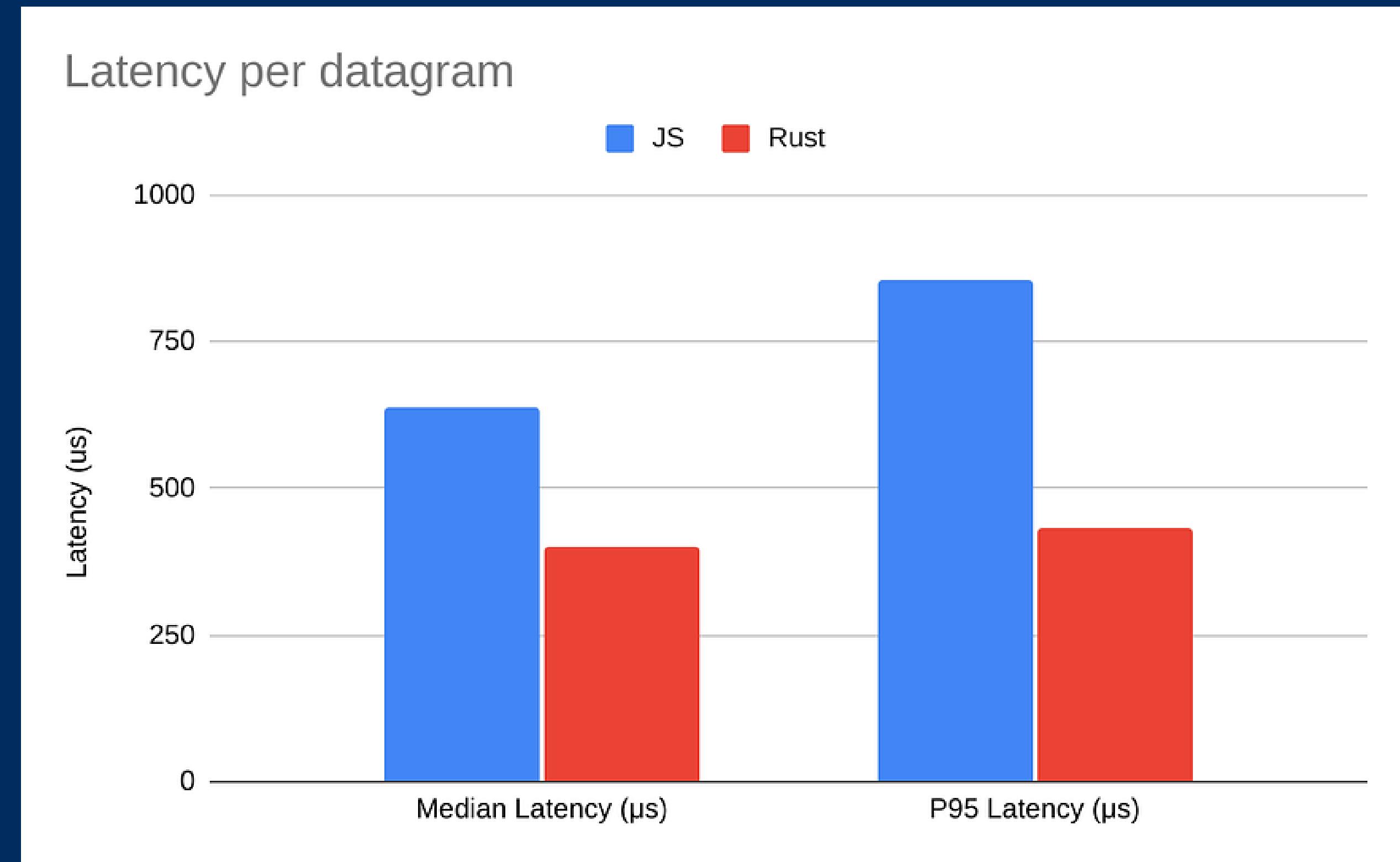
- 그 결과, 프로덕션 환경에서 CPU 사용량이 75%, 메모리 사용량이 95% 감소
(CPU : 800ms → 200ms, Memory : 70MB → 5MB)



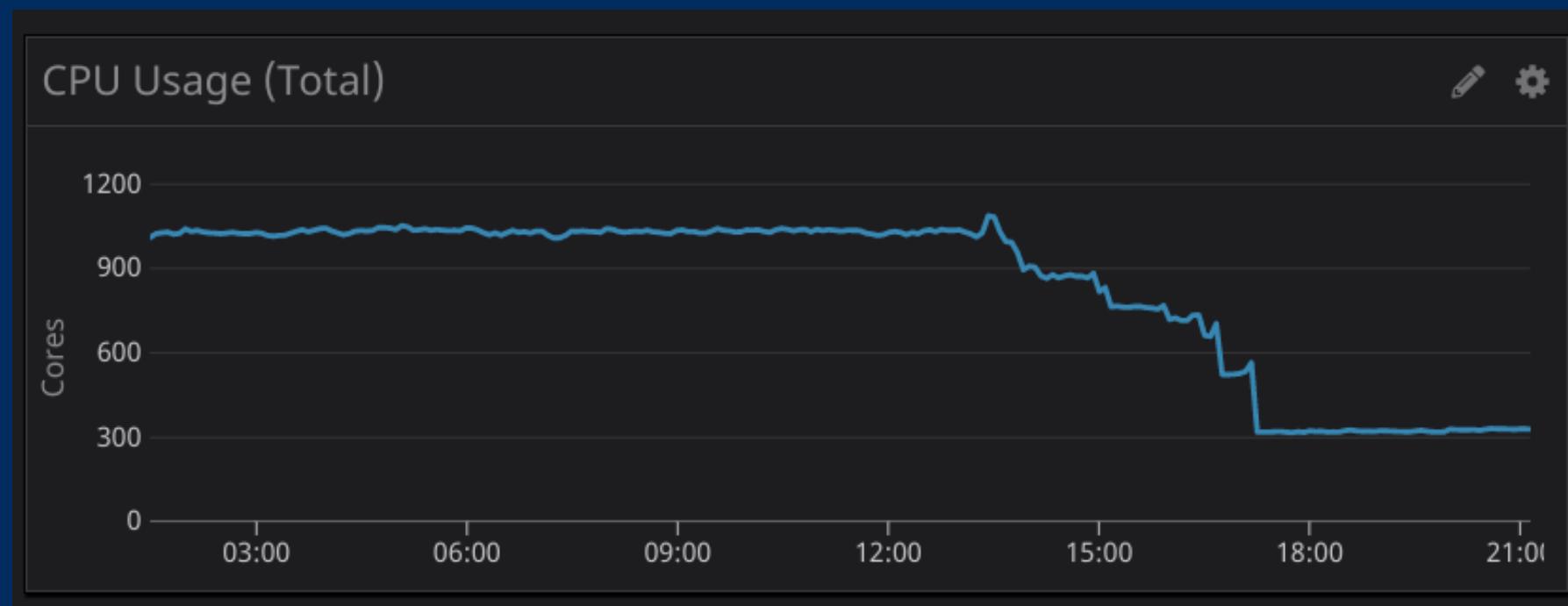
Tenable

현대엔지비 특강
Rust는 왜 주목 받고 있는가

- 또한 패킷 당 대기 시간이 50% 이상 감소



- 새로운 필터가 배포된 후 CPU/메모리 사용량이 급격히 감소되었음을 확인



- Julius Gustavsson은 Volvo Cars Corporation의 기술 전문가로, 볼보에 입사할 당시 Rust에 마음이 사로잡혀 있었다고 함
 - 안전성이 중요한 소프트웨어를 개발할 때 원하는 것과 동일한 이념을 구현하기 때문에 볼보 자동차에도 유용할 것이라고 생각했다고 함
 - 모호하고 찾기 어려운 오류는 자주 발생하지는 않지만, 발생하면 고통스러움
- 볼보에 입사하고 나서 간단한 프로젝트부터 Rust로 구현하기 시작함
- 결과에 꽤 만족하고 있고, 볼보 자동차에서 Rust를 확장하고 싶다는 계획을 가지고 있음
 - 경합 상태, 메모리 손상, 메모리 안전에 대해서 생각할 필요가 없다는 게 큰 장점

- 사용자 지표 정보를 처리하는 애플리케이션 개발 필요
- 핵심 기능을 포함하는 프로토타입을 Python과 Rust로 작성해 비교

| 언어 | 평균 시간 (나노초) | 최대 시간 (나노초) | 최소 시간 (나노초) | 평균 메모리 사용량 (바이트) |
|-----|-------------|-------------|-------------|------------------|
| 파이썬 | 3,852,418 | 7,380,500 | 931,875 | 51,053 |
| 러스트 | 2,014,239 | 5,465,708 | 1,072,875 | 11,397 |

- 작업 처리 속도는 Rust가 Python보다 1.9배 정도 더 빠름
- 메모리 사용량은 Python이 Rust보다 4.5배 정도 더 많음
- CPU 사용량은 Python이 Rust보다 최대 3배, 평균 2배 정도 더 많음
- 비동기 처리가 동일하게 구현된 단일 스레드 환경에서는 Rust의 메시지 처리 속도가 Python보다 10배 정도 빠름

- Meta에서는 백엔드 서버를 작성하는 언어 중 하나로 Rust를 채택
- Mozilla에서 개발하는 Firefox 브라우저의 엔진(Servo)는 Rust로 작성
- Next.js의 컴파일 엔진은 Rust로 재작성
- AWS의 Lambda에서 컨테이너는 FireCracker라는 Rust 툴 위에서 실행
- Sentry는 Python의 낮은 성능 지표를 Rust를 도입해 해결함

감사합니다.

utilForever@gmail.com

<https://github.com/utilForever>

Facebook, Twitter: @utilForever