

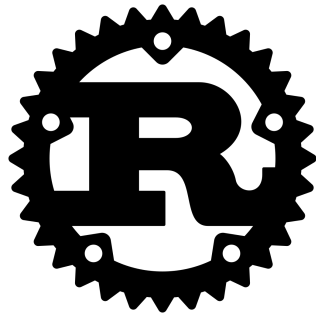
Finding ICE Bugs in Rust Compilers

변지석, 신해소진, 오학주

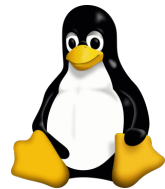
고려대학교 소프트웨어분석연구실

소프트웨어재난연구센터 2024년 여름 정기워크샵, 2024-Jul-9

Rust 프로그래밍 언어 & 컴파일러



- 쓰레기 수집기 없이도 메모리 안전성 보장
- 강력한 타입 시스템
- C/C++ 과 비슷한 성능




The Rust experiment

The Rust support was merged in v6.1 into mainline in order to help in determining whether Rust as a language was suitable for the kernel, i.e. worth the tradeoffs.



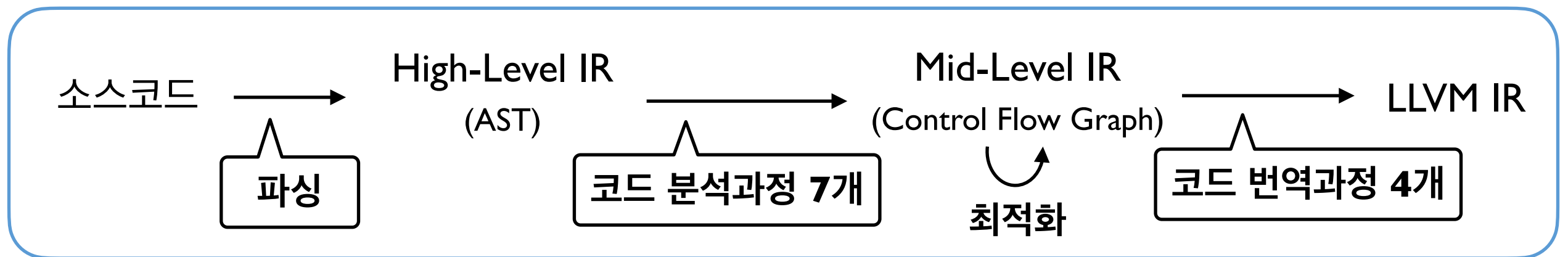
AWS has started upgrading the software behind S3 storage cloud

Sharding system coded in 40,000+ lines of Rust is changing the way cloud colossus ensures data durability

 [Simon Sharwood, APAC Editor](#)

Thu 28 Oct 2021 // 02:59 UTC

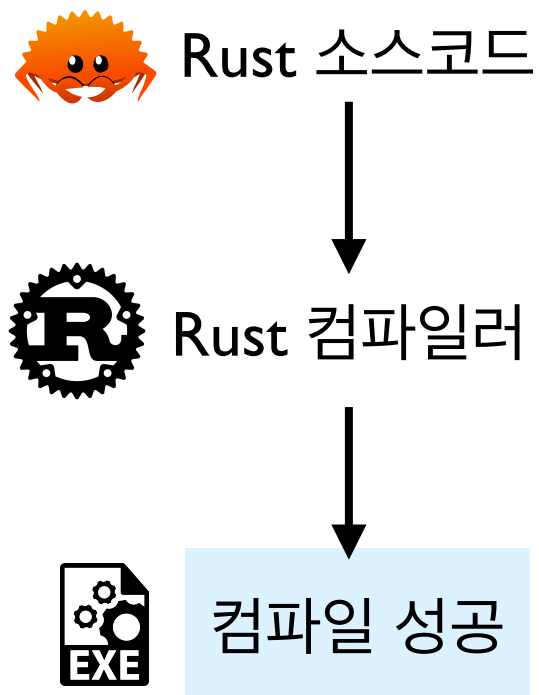
Rust 컴파일러



- 구현체: 72개 패키지, 530,000 LOC

Rust 컴파일러 테스트

Rust 컴파일러 테스트 성공 확인방법: **Internal Compiler Error (ICE)** 발생여부



Error
Warning

컴파일 실패: Rust 코드의 잘못된 지점을 짚어주는 에러 메시지가 출력됨

```
error[E0601]: `main` function not found in crate `117100`
--> 117100.rs:8:2
   |
8  | }
   | ^ consider adding a `main` function to `117100.rs`
error: aborting due to 2 previous errors; 5 warnings emitted
```



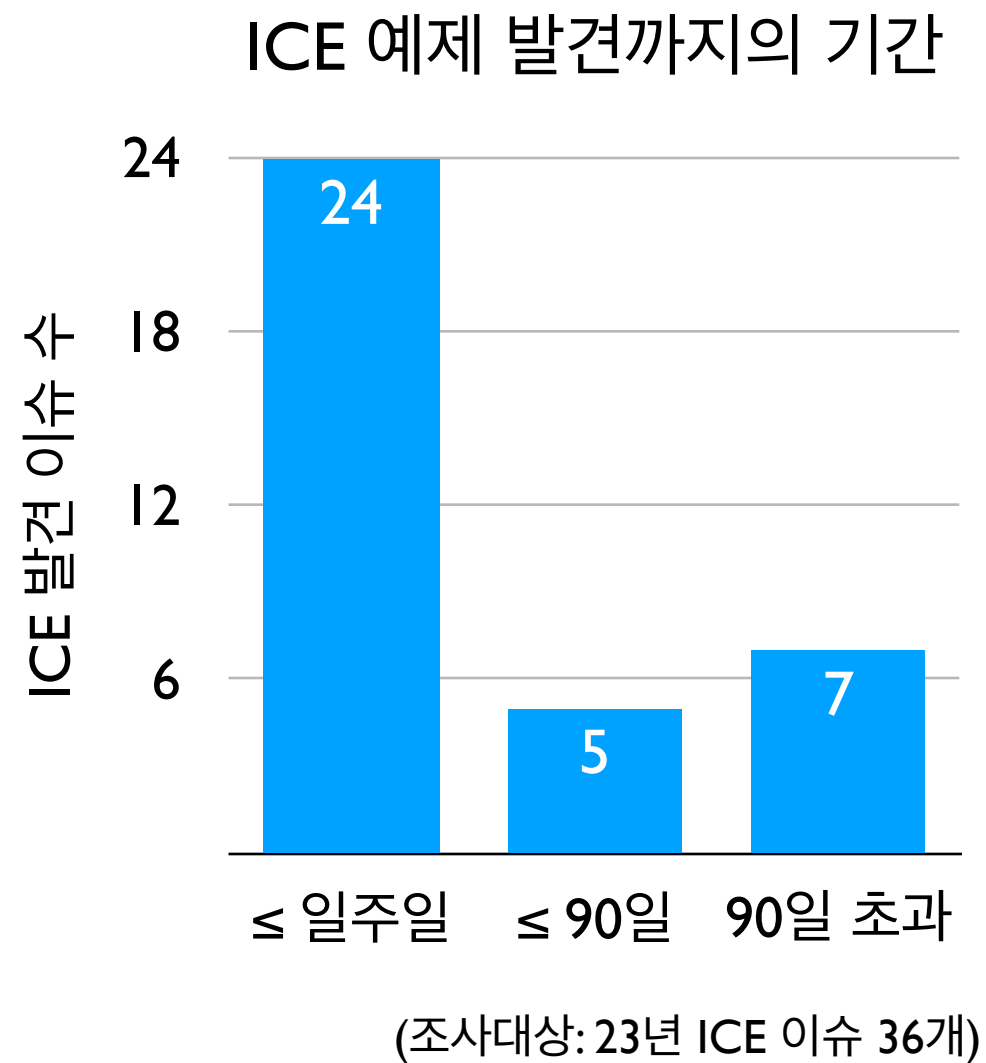
ICE 발생: 크래시 메시지와 함께 버그 리포트 작성을 부탁함

```
thread 'rustc' panicked at compiler/rustc_hir_analysis/src/collect/generics_of.rs:88:71:
no entry found for key
error: the compiler unexpectedly panicked. this is a bug.

note: we would appreciate a bug report: https://github.com/rust-lang/rust/issues/new?labels=C-bug%2C+I-ICE%2C+T-compiler&template=ice.md
```

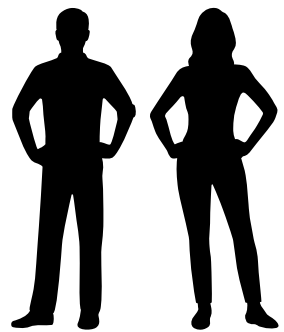
더 많은 Rust 컴파일러 테스트 필요

발견 년도	# 해결된 ICE 이슈
2020	375
2021	339
2022	425
2023	472
2024	> 250



Rust 컴파일러는 테스트하기 어렵다

어떤 코드가 버그를 일으킬 수 있는지 컴파일러 개발자도 쉽게 떠올리지 못한다.



개발자들

(24년 2월) 가진 테스트로는 오류상황에 도달 못함

rust-lang/rust #121208



(24년 3월) 오류상황에 도달 가능해짐

rust-lang/rust #121154



(24년 6월) 테스트 발견

rust-lang/rust #126670

3달동안 발견하지 못한 테스트

```
1  pub trait A {}
2
3  pub trait Mirror {
4      type Assoc: ?Sized;
5  }
6  impl<T: ?Sized> Mirror for dyn A {
7      type Assoc = T;
8  }
9
10 pub trait First {
11     async fn first() -> <dyn A + 'static as Mirror>::Assoc;
12 }
13
14 impl First for dyn A {
15     async fn first() {}
16 }
17
18 fn main() {}
19
```

오류) 유추할 수 없는 타입

rust-lang/rust #126670

Rust ICE 를 찾기 위한 방법 (wip)

```
1 pub trait A {}
2
3 pub trait Mirror {
4     type Assoc: ?Sized;
5 }
6 impl<T: ?Sized> Mirror for A {
7     type Assoc = T;
8 }
9 pub fn foo<'a>(x: &'a <dyn A + 'static as Mirror>::Assoc) {}
```

Seed 1

I. 타입을 바꿔 적고 (tree-splice)

```
11 struct A;
12 pub trait First {
13-     async fn first(self) -> A;
13+     async fn first(self) -> <dyn A + 'static as Mirror>::Assoc;
14 }
15 impl First for A {
16     async fn first(self) -> A {}
17 }
18
19 fn main() {}
```

Seed 2

Rust ICE 를 찾기 위한 방법 (wip)

```
1 pub trait A {}
2
3 pub trait Mirror {
4     type Assoc: ?Sized;
5 }
6 impl<T: ?Sized> Mirror for A {
7     type Assoc = T;
8 }
9 pub fn foo<'a>(x: &'a <dyn A + 'static as Mirror>::Assoc) {}
```

Seed 1

1. 타입을 바꿔 적고 (tree-splice)



2. 바뀐 타입 관련 코드를 덧붙였다

```
11 struct A;
12 pub trait First {
13-     async fn first(self) -> A;
13+     async fn first(self) -> <dyn A + 'static as Mirror>::Assoc;
14 }
15 impl First for A {
16     async fn first(self) -> A {}
17 }
18
19 fn main() {}
```

Seed 2

Rust ICE 를 찾기 위한 방법 (wip)

```
1 pub trait A {}
2
3 pub trait Mirror {
4     type Assoc: ?Sized;
5 }
6 impl<T: ?Sized> Mirror for A {
7     type Assoc = T;
8 }
9 pub fn foo<'a>(x: &'a <dyn A + 'static as Mirror>::Assoc) {}
```

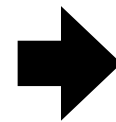
Seed 1

1. 타입을 바꿔 적고 (tree-splice)



2. 바뀐 타입 관련 코드를 덧붙였다

기존 변이방법 (Tree-splice) 사용
새로 찾은 ICE 수 : **22개** / **2개월+**



위 변이방법 사용
새로 찾은 ICE 수 : **20개** / **1개월**

20개 중 **18개**는 기존 변이방법으로 만들기 어려운 코드이다

```
19 fn main() {}
```

Seed 2

기존 Rust 컴파일러 테스트 생성방법

1. 오류 없는 Rust 테스트 코드 생성 연구

→ 제한된 문법으로 테스트를 생성

RustSmith 런타임 오류 없는 테스트 생성, ISSTA'23 (Tool Demo.)

Fuzzing the Rust Typechecker Using CLP 타입 오류 없는 테스트 생성, ASE'15

2. Rust 코드 무작위 생성도구

→ 문법은 거의 맞지만 컴파일러 테스트에 적합하지 않은 변이를 많이 생성

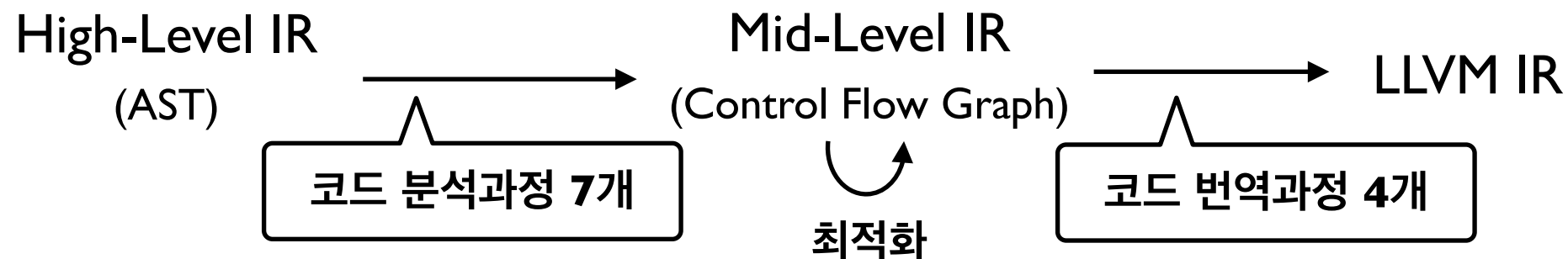
rust-tree-splicer AST 가지 접붙이기 변이

fuzz-rustc LLVM lib-fuzzer 를 이용

목표 : 컴파일러 분석/번역 과정 테스트에 적합한 변이방법 제시

Rust ICE 를 찾기 위한 방법 (wip)

타입만 잘 바뀌적었을 뿐인데 다양한 ICE 를 찾는다



ICE 가 발생하는 컴파일러 기능 위치들 (합 18개):

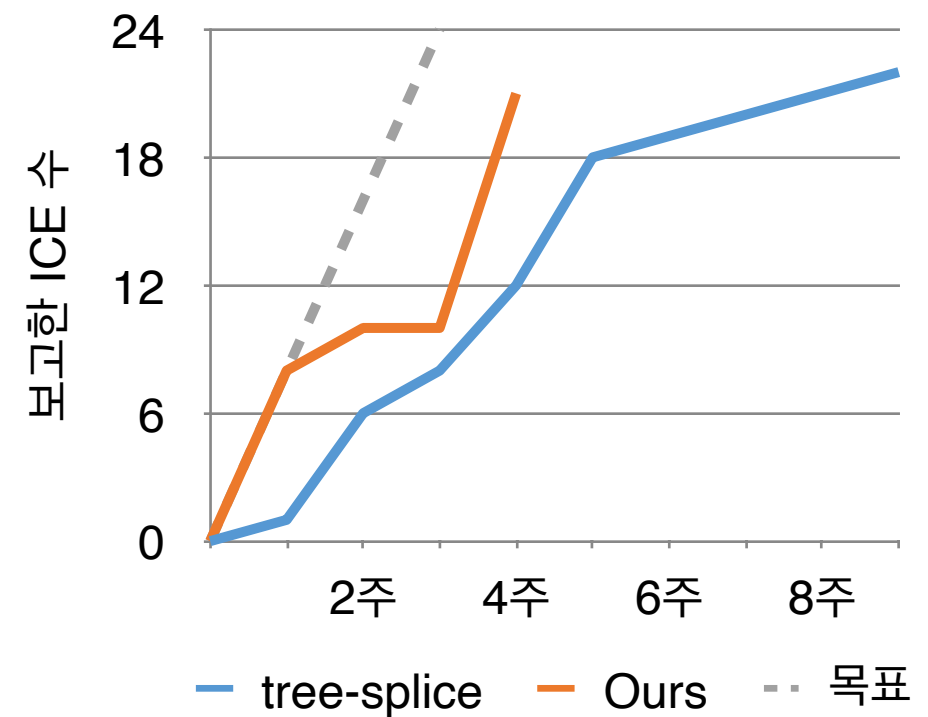
- | | | | | | |
|-----------------------------|----------|-----------------------------|----------|--------------------------------|----------|
| • <u>Type Inference</u> : | 2 | • <u>Unsafty Checking</u> : | 0 | • <u>MIR Build</u> : | 2 |
| • <u>Trait Selection</u> : | 1 | • <u>Drop Elaboration</u> : | 0 | • <u>Constant Evaluation</u> : | 4 |
| • <u>Type Checking</u> : | 5 | • <u>Borrow Checking</u> : | 0 | • <u>Monomorphization</u> : | 0 |
| • <u>Pattern Checking</u> : | 2 | • <u>MIR Optimization</u> : | 2 | • <u>Code Generation</u> : | 0 |

To-Achieve

세부목표 1) 새 ICE 50개 이상 리포트

세부목표 2) 기존 방법보다 3배 많은 ICE 발견

테스트 생성 방법	#보고됨	#수정됨
전체	43	13
우리 방법과 관련됨	18	5



Backup Slides

Rust 컴파일러는 테스트하기 어렵다

발견된 테스트 코드 :

```
1  pub trait A {}
2
3  pub trait Mirror {
4      type Assoc: ?Sized;
5  }
6  impl<T: ?Sized> Mirror for dyn A {
7      type Assoc = T;
8  }
9
10 pub trait First {
11     async fn first() -> <dyn A + 'static as Mirror>::Assoc;
12 }
13
14 impl First for dyn A {
15     async fn first() {}
16 }
17
18 fn main() {}
19
```

trait : 일종의 인터페이스

?Sized : 크기가 정해지지 않아도 되는 타입이다

A trait 을 가지는 타입에 Mirror trait 부여

오류) 유추할 수 없는 타입

first() async 함수의 타입체크 유발

rust-lang/rust #126670