Cross-platform XOffsetDatastructure: Ensuring Zero-encoding/Zero-decoding Serialization Compatibility Through **Compile-time Type Signatures**

Fanchen Su (whucssfc@gmail.com)

CppCon
The C++ Conference
2025
September 13-19
Aurora, Colorado, USA

Introduction

This poster presents a **cross-platform** solution for XOffsetDatastructure, a **zero-encode/decode** serialization library, using compile-time verification to ensure reliable performance across platforms.

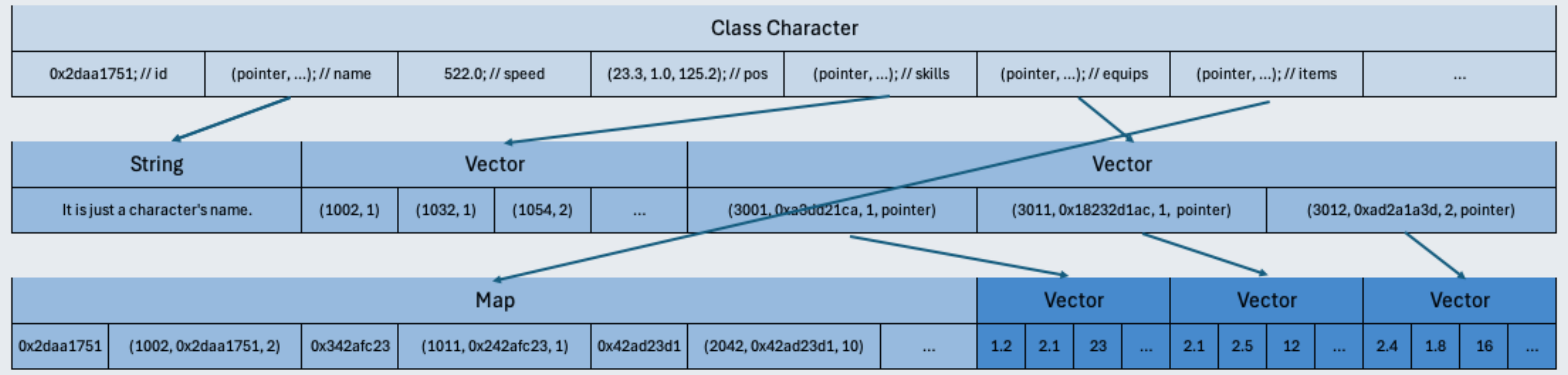
Motivation

Serialization is a major bottleneck. XOffsetDatastructure accelerates it via direct memory access, but **cross-platform** adoption is challenging due to hardware, compiler, and layout differences. Modern games target consoles, mobile, PC, and cloud.

The central question: how can zero-encode/decode work reliably across all these platforms while preserving performance?

XOffsetDatastructure Overview

https://github.com/ximicpp/XOffsetDatastructure (CppCon 2024)

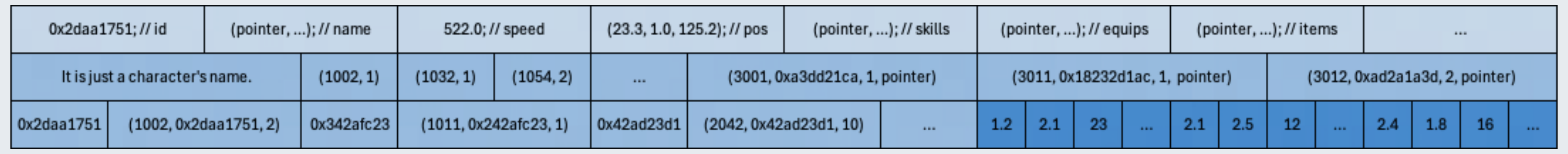


Class Character							
0x2daa1751; // id	(pointer, ...); // name	522.0; // speed	(23.3, 1.0, 125.2); // pos	(pointer, ...); // skills	(pointer, ...); // equips	(pointer, ...); // items	...

String	Vector				Vector		
It is just a character's name.	(1002, 1)	(1032, 1)	(1054, 2)	...	(3001, 0xa3dd21ca, 1, pointer)	(3011, 0x18232d1ac, 1, pointer)	(3012, 0xad2a1a3d, 2, pointer)

Map							Vector				Vector				Vector			
0x2daa1751	(1002, 0x2daa1751, 2)	0x342afc23	(1011, 0x242afc23, 1)	0x42ad23d1	(2042, 0x42ad23d1, 10)	...	1.2	2.1	23	...	2.1	2.5	12	...	2.4	1.8	16	...

Memory data structure



0x2daa1751; // id	(pointer, ...); // name	522.0; // speed	(23.3, 1.0, 125.2); // pos	(pointer, ...); // skills	(pointer, ...); // equips	(pointer, ...); // items	...

It is just a character's name.	(1002, 1)	(1032, 1)	(1054, 2)	...	(3001, 0xa3dd21ca, 1, pointer)	(3011, 0x18232d1ac, 1, pointer)	(3012, 0xad2a1a3d, 2, pointer)

0x2daa1751	(1002, 0x2daa1751, 2)	0x342afc23	(1011, 0x242afc23, 1)	0x42ad23d1	(2042, 0x42ad23d1, 10)	...	1.2	2.1	23	...	2.1	2.5	12	...	2.4	1.8	16	...

Data buffer

The memory data structure and the data buffer are equivalent. Data can be sent directly, without any additional encoding or decoding processes.

Cross-platform Compatibility Strategy

Our main contribution is a **three-tier compatibility solution**.

First Tier - Platform Targeting

Based on market analysis showing 99% of gaming platforms are **64-bit little-endian**, we constrain target platforms through compile-time verification using static assertions to ensure 64-bit architecture and little-endian byte ordering requirements.

```
static_assert(sizeof(void*) == 8,   "64-bit architecture required"  );
static_assert(sizeof(size_t) == 8,  "64-bit architecture required"  );
static_assert(IS_LITTLE_ENDIAN,  "Little-endian architecture required");
```

Second Tier - Compatible Type Subset

We define cross-platform types using Boost.Container for consistent behavior across platforms (strings, vectors, sets, maps).

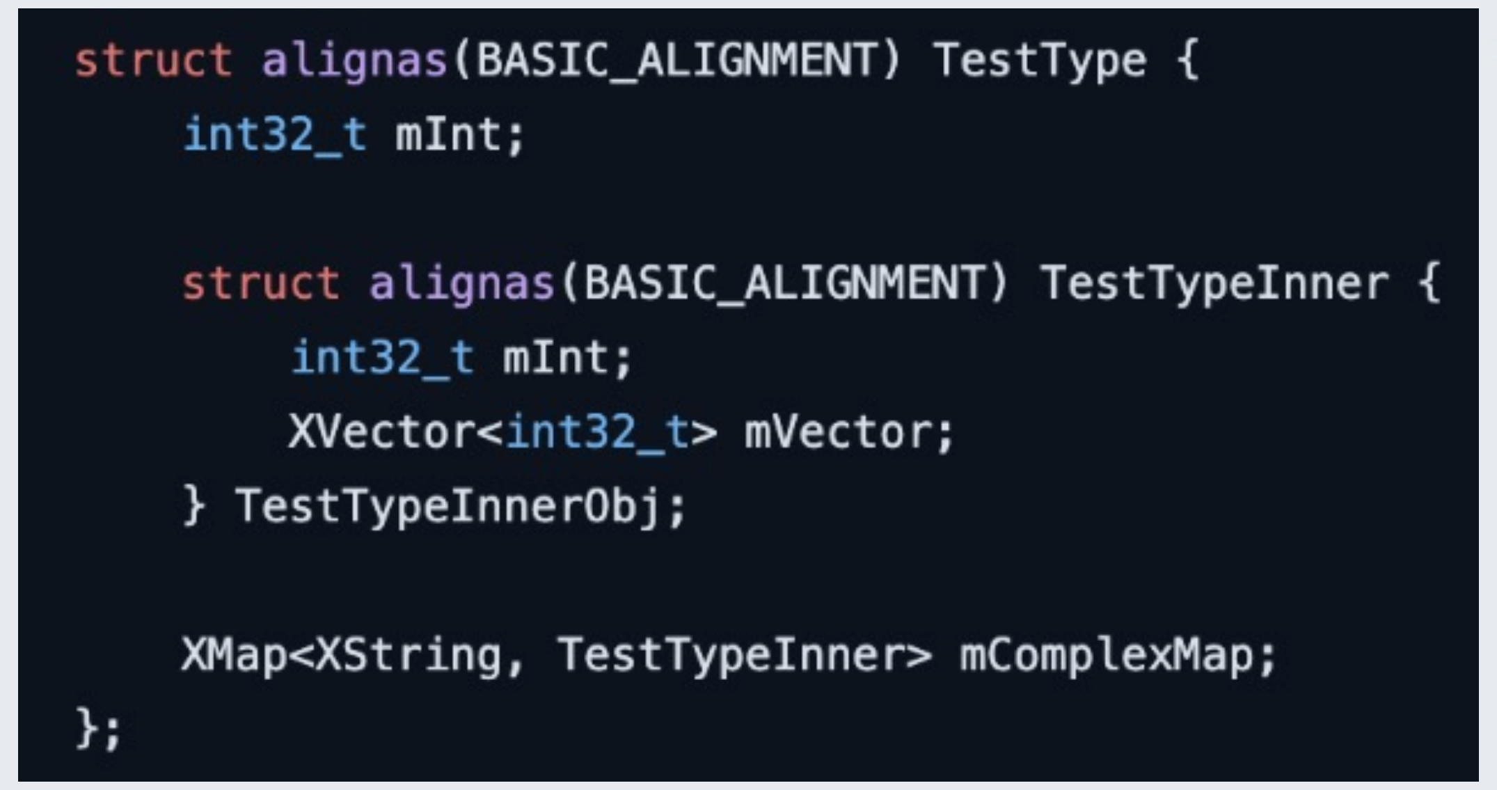
We exclude features that vary across platforms (virtual functions, multiple inheritance, RTTI) and preserve expressivity through composition and nested containers.

```
using XString = boost::container::basic_string<char>;
template <typename T> using XVector = boost::container::vector<T>;
template <typename T> using XSet = boost::container::set<T>;
template <typename K, typename V> using XMap = boost::container::map<K, V>;
```

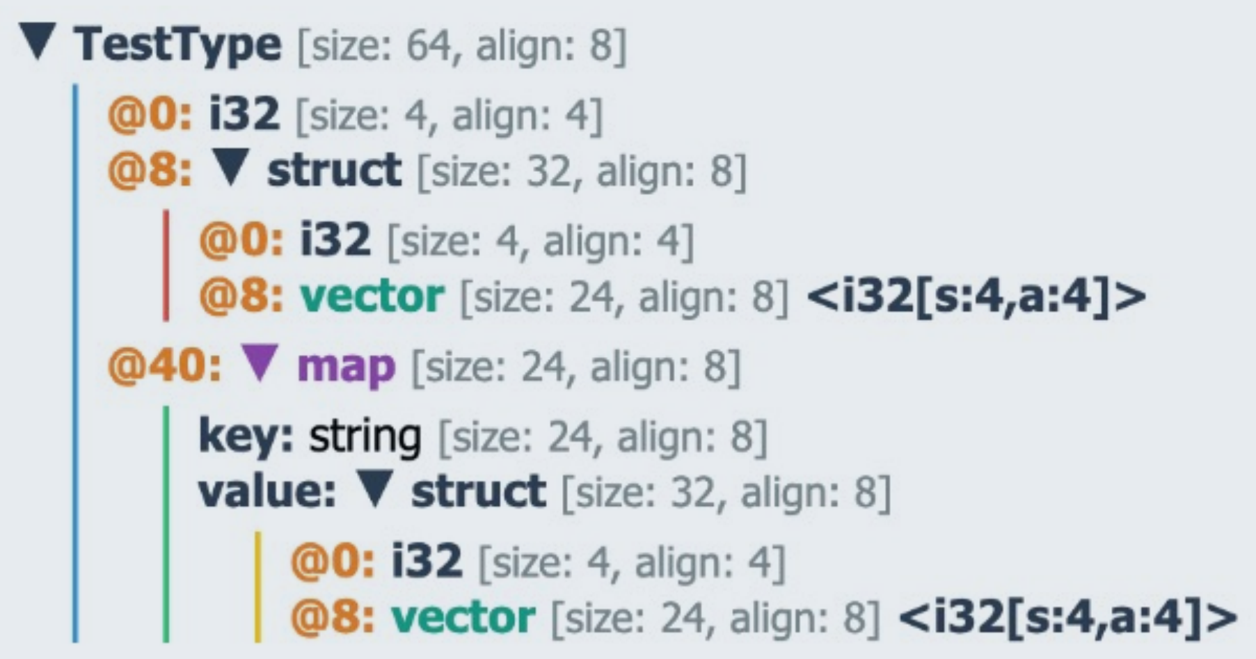
Third Tier - Compile-Time Verification

We implement a **type signature system** using modern C++17/20 features that generates unique signatures capturing complete memory layout information.

These signatures include size, alignment, and field offset information in a structured format that can be verified at compile time through **static assertions.**

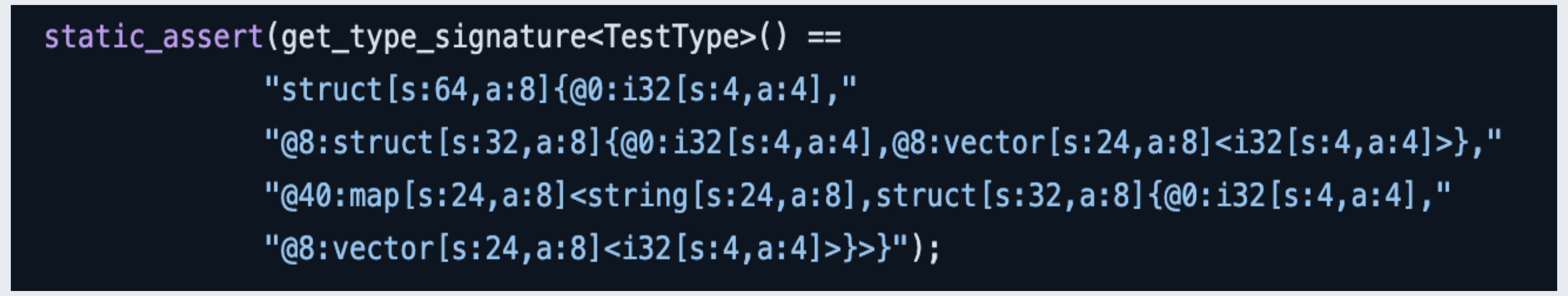


```
struct alignas(BASIC_ALIGNMENT) TestType {
    int32_t mInt;

    struct alignas(BASIC_ALIGNMENT) TestTypeInner {
        int32_t mInt;
        XVector<int32_t> mVector;
    } TestTypeInnerObj;

    XMap<XString, TestTypeInner> mComplexMap;
};
```



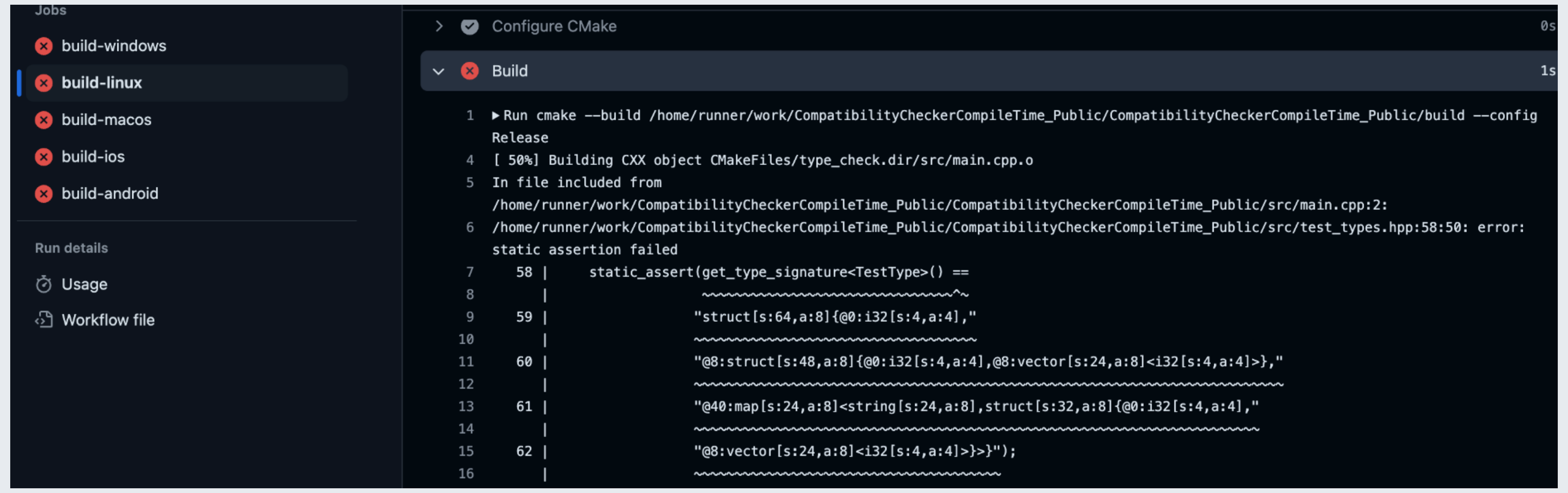
```
▼ TestType [size: 64, align: 8]
  @0: i32 [size: 4, align: 4]
  @8: ▼ struct [size: 32, align: 8]
      @0: i32 [size: 4, align: 4]
      @8: vector [size: 24, align: 8] <i32[s:4,a:4]>
  @40: ▼ map [size: 24, align: 8]
      key: string [size: 24, align: 8]
      value: ▼ struct [size: 32, align: 8]
          @0: i32 [size: 4, align: 4]
          @8: vector [size: 24, align: 8] <i32[s:4,a:4]>
```

Example aligned type with nested struct, vector, and map



```
static_assert(get_type_signature<TestType>() ==
              "struct[s:64,a:8]{@0:i32[s:4,a:4],"
              "@8:struct[s:32,a:8]{@0:i32[s:4,a:4],@8:vector[s:24,a:8]<i32[s:4,a:4]>},"
              "@40:map[s:24,a:8]<string[s:24,a:8],struct[s:32,a:8]{@0:i32[s:4,a:4],"
              "@8:vector[s:24,a:8]<i32[s:4,a:4]>}>}");
```

Compile-time type signature



```
▶Run cmake --build /home/runner/work/CompatibilityCheckerCompileTime_Public/CompatibilityCheckerCompileTime_Public/build --config
Release
[ 50%] Building CXX object CMakeFiles/type_check.dir/src/main.cpp.o
In file included from
/home/runner/work/CompatibilityCheckerCompileTime_Public/CompatibilityCheckerCompileTime_Public/src/main.cpp:2:
/home/runner/work/CompatibilityCheckerCompileTime_Public/CompatibilityCheckerCompileTime_Public/src/test_types.hpp:58:50: error:
static assertion failed
   58 |     static_assert(get_type_signature<TestType>() ==
      |                   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~^~
   59 |                 "struct[s:64,a:8]{@0:i32[s:4,a:4],"
      |                 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   60 |                 "@8:struct[s:48,a:8]{@0:i32[s:4,a:4],@8:vector[s:24,a:8]<i32[s:4,a:4]>},"
      |                 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   61 |                 "@40:map[s:24,a:8]<string[s:24,a:8],struct[s:32,a:8]{@0:i32[s:4,a:4],"
      |                 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   62 |                 "@8:vector[s:24,a:8]<i32[s:4,a:4]>}>}");
      |                 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Compile-time type signature check; on mismatch, static_assert fails and the build stops.

Results and Conclusions

Enables **cross-platform** zero-encode/decode while preserving the XOffsetDatastructure's performance.

Balances **compatibility** and **expressivity** for high-performance data exchange in games and distributed systems.