# static_any

Ultra-fast, stack-based generic container

Maciek Gajewski
David Gross

# Motivation

- Needed something like **boost::any** or **QVariant**
- Can store single item of any (copyable) type
- Faster!
- No heap allocation

# Example use

```
static_any<8> a = 7;
int x = a.get<int>(); // returns 7
```

# Example use

```
static_any<8> a = 7;
int x = a.get<int>(); // returns 7


bool hi = a.has<int>(); // returns true
bool hd = a.has<double>(); // returns false
```

# Example use

```
static_any<8> a = 7;
int x = a.get<int>(); // returns 7

bool hi = a.has<int>(); // returns true
bool hd = a.has<double>(); // returns false

double d = a.get<double>(); // throws!
```

# Example use

```
static_any<8> a = 7;
int x = a.get<int>(); // returns 7

bool hi = a.has<int>(); // returns true
bool hd = a.has<double>(); // returns false

double d = a.get<double>(); // throws!

a = std::make_pair(4.4, 5.5); // fails to compile
```

# Example use (2)

```
static_any<32> a = std::string(“Hello”);
```

# Example use (2)

```
static_any<32> a = std::string(“Hello”);
static_any<32> a2 = a; // object copied
```

# Example use (2)

```
static_any<32> a = std::string(“Hello”);
static_any<32> a2 = a; // object copied
static_any<32> a3 = std::move(a2); // object moved
```

# Example use (2)

```
static_any<32> a = std::string(“Hello”);
static_any<32> a2 = a; // object copied
static_any<32> a3 = std::move(a2); // object moved
a = 5; // object destroyed
```

# How does it work

```
template<size_t N>
class static_any
{
  // …
private:
  using fun_ptr_t = void(*)(op_t, void*, void*);

  std::array<char, N> buff_;
  fun_ptr_t           function_ = nullptr;
};

// size = N + sizeof(void*)
```

# The “Gateway function”

```
enum class op_t { copy, move, destroy, ... };


template<typename T>
static void operation(op_t op, void* ptr1, void* ptr2)
{
  switch(op)
  {
    // all operations handled here
  }
}
```

# Copy - initialization

```
template<typename T>
static_any(const T& obj)
{
  static_assert(N >= sizeof(T), "T is too big");
  new(buff_.data()) T(obj); // placement new
  function_ = &operation<T>;
}
```

# Destruction

```
void reset()
{
  if (function_) {
    function_(op_t::destroy, buff_.data(), nullptr);
    function_ = nullptr;
  }
}
~static_any() { reset(); }

// in the “gateway function”
T* this_ptr = reinterpret_cast<T*>(ptr1);
this_ptr->~T();
```

# Copy from another static_any

```
static_any(const static_any& other)
{
  if (other.function_) {
    function_ = other.function_;
    function_(op_t::copy, buff_.data(), other.buff_.data());
  }
}

// in the “gateway function”
T* this_ptr = reinterpret_cast<T*>(ptr1);
T* other_ptr = reinterpret_cast<T*>(ptr2);
new(this_ptr)_T(*other_ptr);
```

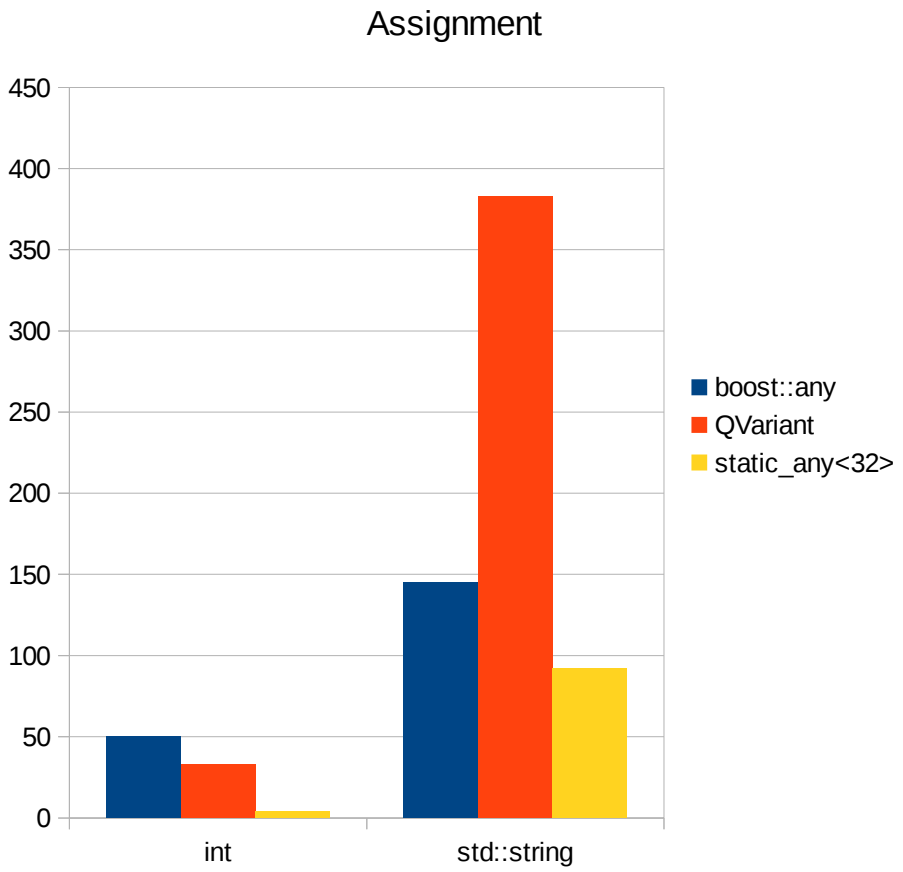
# Getting the stored object

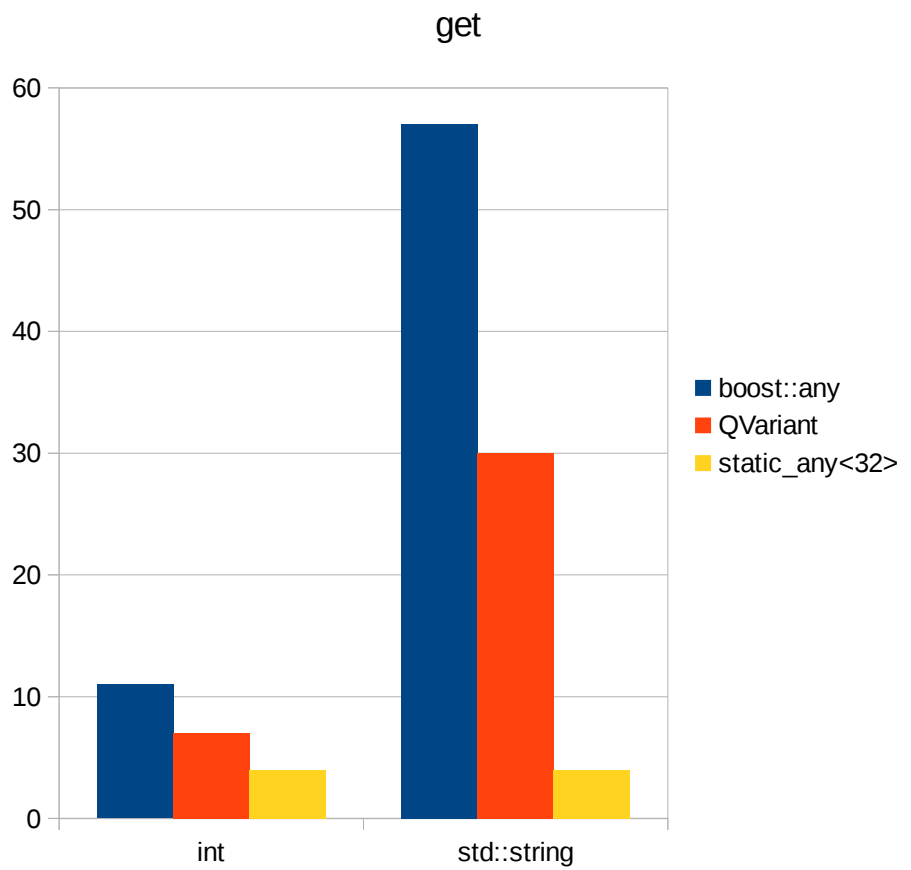
```
template<typename T>
T& get()
{
  if (has<T>())
    return *reinterpret_cast<T*>(buff_.data());
  else
    throw std::bad_cast;
}
```

# Testing type

```
template<typename T>
bool has() const
{
  if (function_ == &operation<T>)
    return true;
  else if (function_)
    return compare_type_index(); // across dll?
  else
    return false;
}
```

# Performance



# Thank you!

https://github.com/david-grs/static_any

maciej.gajewski0@gmail.com
david.a.grs@gmail.com