inplace_vector

D0843R7

Gonzalo Brito Gadeschi, Timur Doumler, Nevin Liber, David Sankel

LEWG, 2023-06-16

Review

```
inplace_vector<int, 5> v; // Storage on stack
v.push_back(1);
v.push_back(2);
```

- For systems that cannot use a dynamic allocator
- Provides useful set of performance tradeoffs

push_back interfaces*

```
constexpr T& push_back(const value_type& x);

constexpr T* try_push_back(const T& value);

constexpr T& push_back_unchecked(const T& value);
```

* also with T&& parameter and emplace_back styles

push_back

If space is not available, throw std::bad_alloc. Otherwise, return a reference to the newly created element

```
template<typename T, typename U, int N>
class pair_sequence {
  inplace_vector<T, N> ts_;
  inplace_vector<U, N> us_;
public:
  void push(T t, U u) {
    ts_.push_back(t):
    try { us_.push_back(u); }
    catch(...) { ts_.pop_back(); throw; }
  }
};
```

- push_back is a drop-in replacement for that of std::vector
- Returns a reference to the element that was added

  f(v.push_back(x));
- A straightforward extension to std::vector as well if not for ABI concerns

try_push_back

If space is not available, return 0. Otherwise, return a pointer to the newly created element

```
// Pattern 1
if( !v.try_push_back(x) ) {
  panic();
}

// Pattern 2
if( T * p = v.try_push_back(x) ) {
  f(*p);
}
```

- Exhausted inplace_vectors are a more common occurrence than exhausted std::vectors
- try_push_back is convenient for this
- try_push_back also enables this functionality for exception-free systems

push_back_unchecked

If space is not available the behavior is undefined, otherwise return a reference to the newly created element.

```
assert(v.size() < v.capacity())
v.push_back_unchecked(x);
```

- An unsafe interface (alt name unsafe_push_back?)
- May have performance benefits…

Performance Measurements

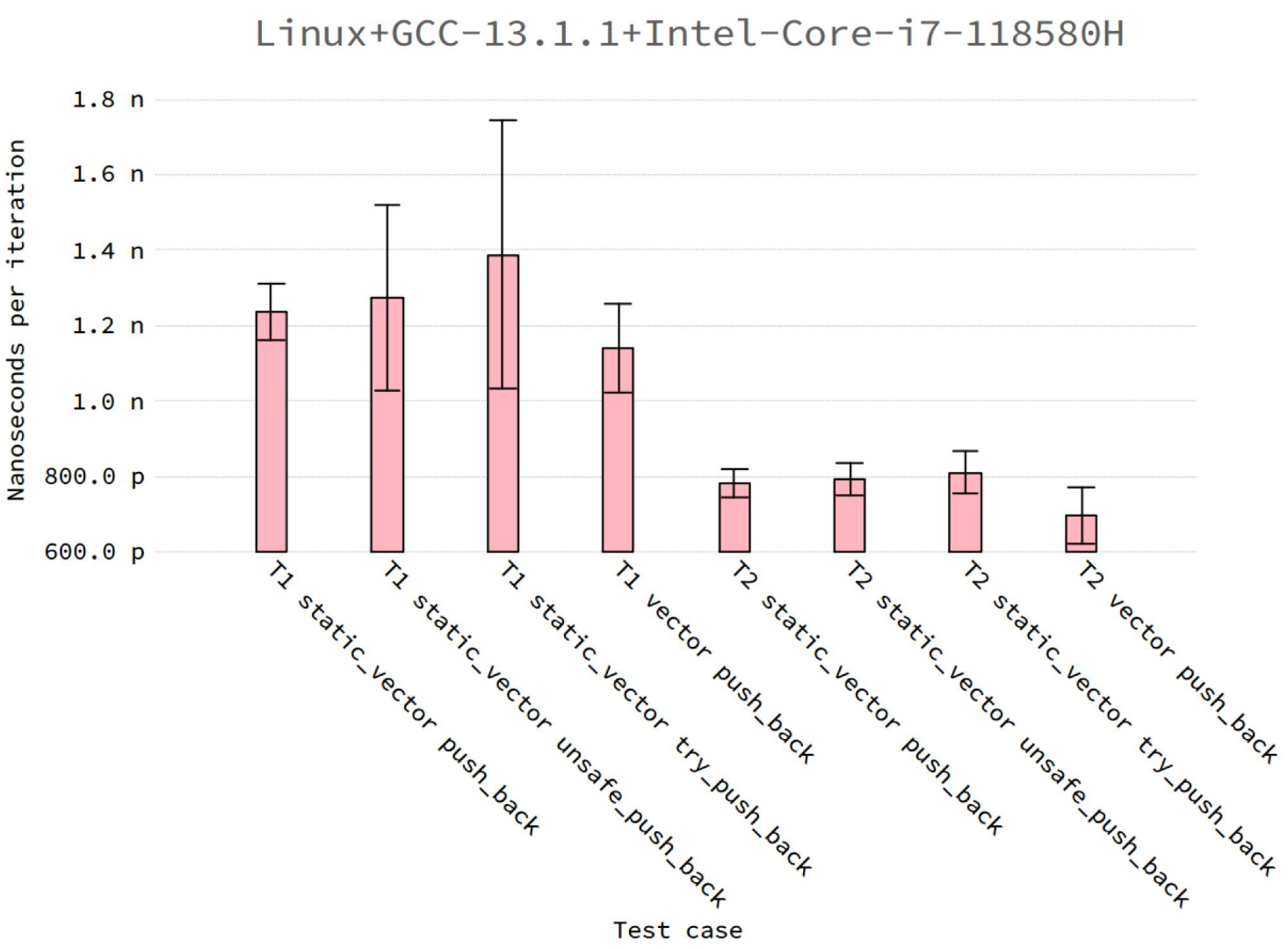
T1

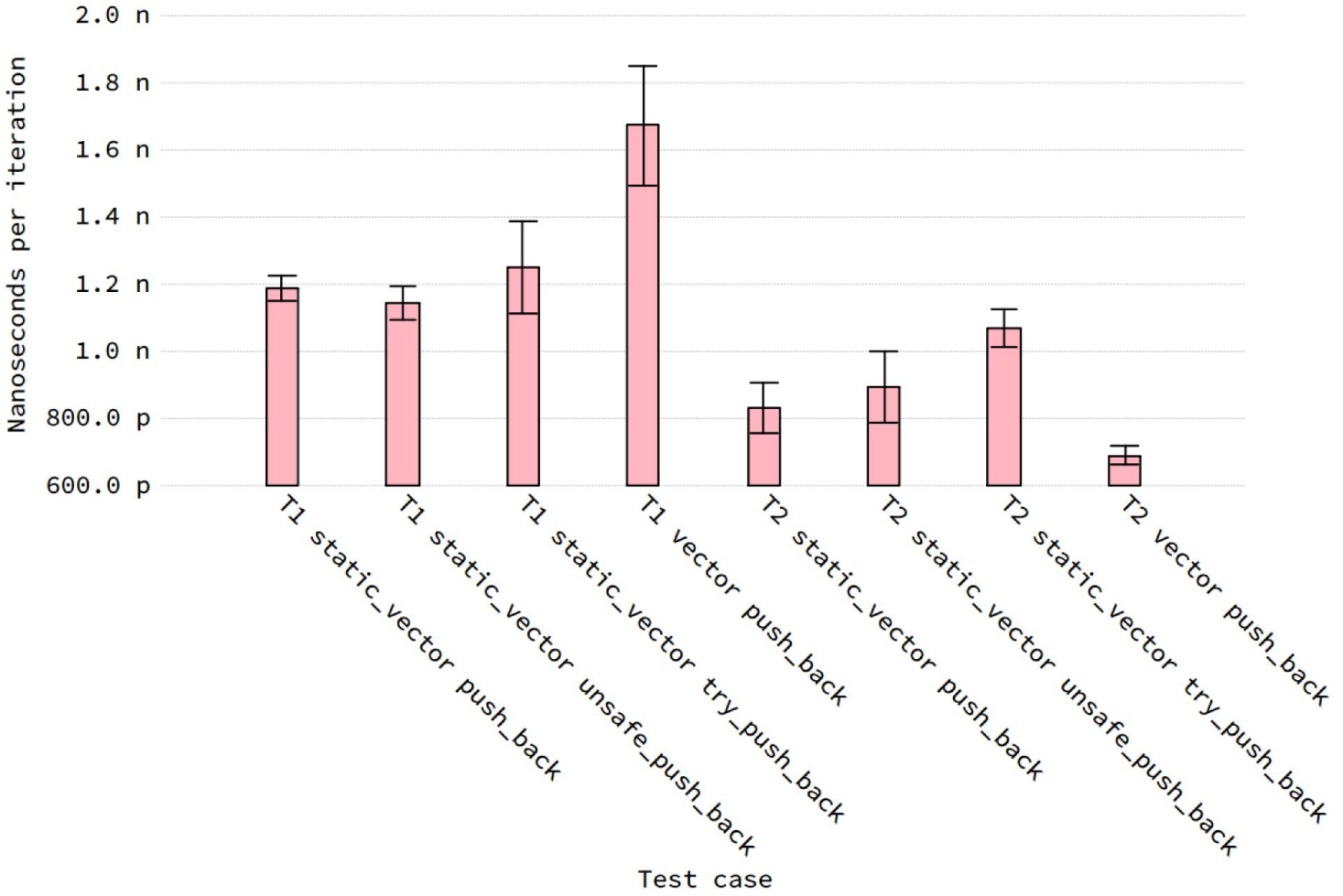
```
for (size_t i = 0; i < N; ++i)
  v.push_back(b[i]);
f(&v);
```

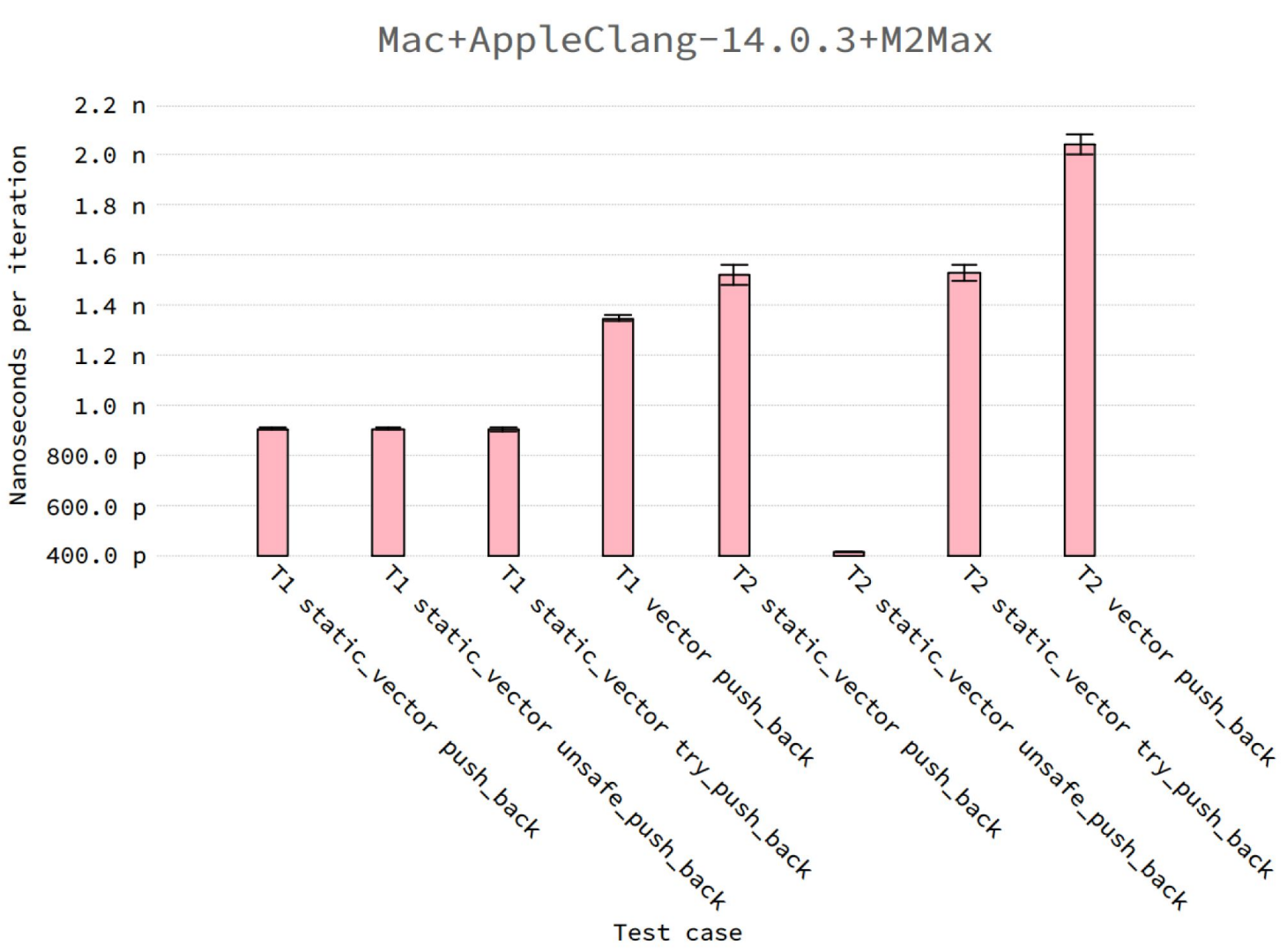
T2

```
for (size_t i = 0; i <N; ++i) {
  v.push_back(b[i]);
  f(&v);
}
```

- Attempt to capture common use cases
- In T2 the compiler must account for the possibility of f changing v's size. There is less opportunity for optimization (e.g. unrolling) here
- GPU/microcontroller performance wasn't checked


Linux+GCC-13.1.1+Intel-Core-i7-118580H
Nanoseconds per iteration
1.8 n
1.6 n
1.4 n
1.2 n
1.0 n
800.0 p
600.0 p
T1 static_vector push_back
T1 static_vector unsafe_push_back
T1 static_vector try_push_back
T1 vector push_back
T2 static_vector push_back
T2 static_vector unsafe_push_back
T2 static_vector try_push_back
T2 vector push_back
Test case


Linux+Clang-17.0.0+Intel-Core-i7-11850H
Nanoseconds per iteration
2.0 n
1.8 n
1.6 n
1.4 n
1.2 n
1.0 n
800.0 p
600.0 p
T1 static_vector push_back
T1 static_vector unsafe_push_back
T1 static_vector try_push_back
T1 vector push_back
T2 static_vector push_back
T2 static_vector unsafe_push_back
T2 static_vector try_push_back
T2 vector push_back
Test case


Mac+AppleClang-14.0.3+M2Max
Nanoseconds per iteration
2.2 n
2.0 n
1.8 n
1.6 n
1.4 n
1.2 n
1.0 n
800.0 p
600.0 p
400.0 p
T1 static_vector push_back
T1 static_vector unsafe_push_back
T1 static_vector try_push_back
T1 vector push_back
T2 static_vector push_back
T2 static_vector unsafe_push_back
T2 static_vector try_push_back
T2 vector push_back
Test case

Bikesheds

- Name of class (inplace_vector, static_vector, fixed_capacity_vector)
- Name of unsafe push_back (push_back_unchecked, unsafe_push_back)