

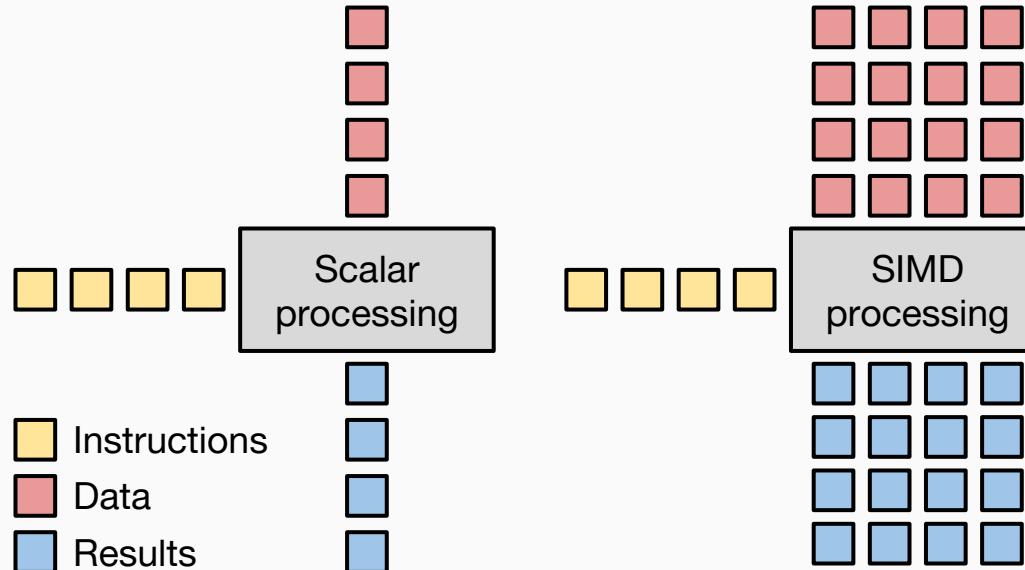
Speeding up Software with VecCore

G. Amadio, P. Canal, D. Piparo, S. Wenzel
for the GeantV and ROOT Teams

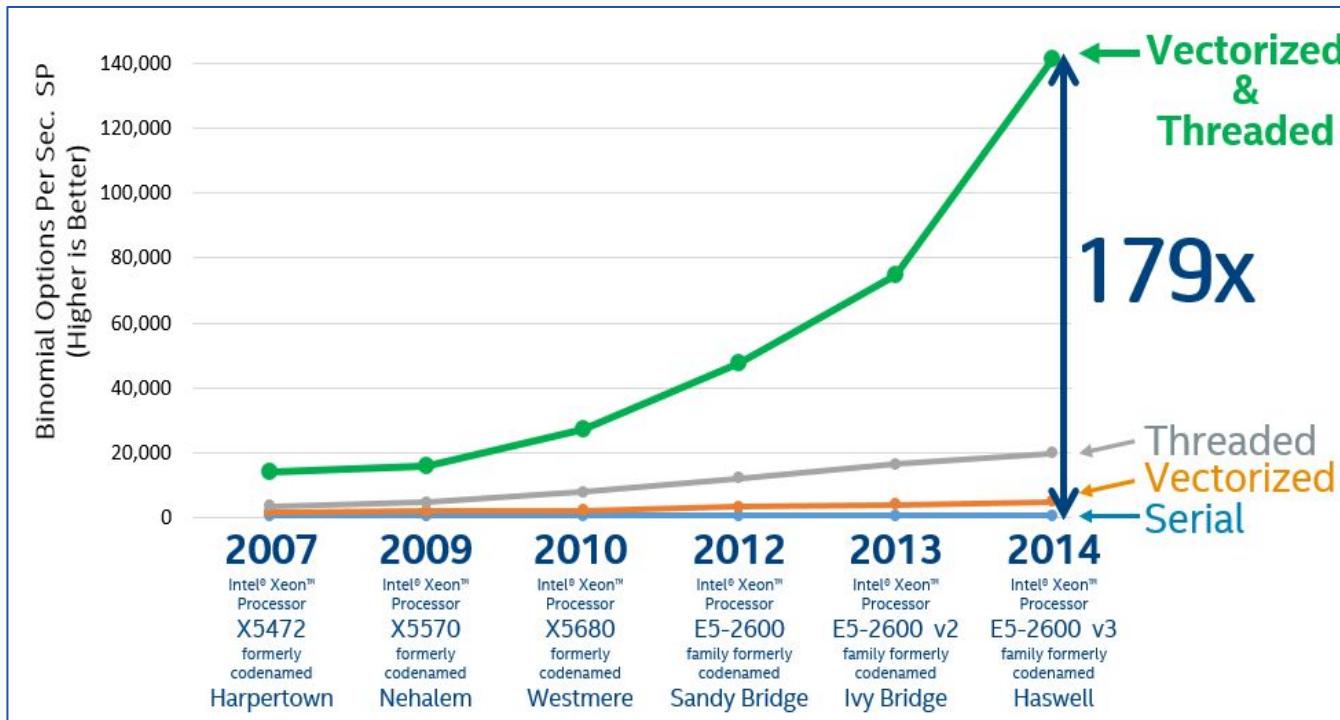
ACAT 2017, Seattle, USA

What is SIMD Vectorization?

- ▶ **SIMD** = Single Instruction, Multiple Data
- ▶ Instruction-level parallelism



Why SIMD Vectorization?



SIMD Programming Models

- ▶ Auto-vectorization
- ▶ OpenMP 4.1
- ▶ Compiler Pragmas
- ▶ SIMD Libraries
- ▶ Compiler Intrinsics
- ▶ Inline Assembly

```
float a[N], b[N], c[N];
for (int i = 0; i < N; i++)
    a[i] = b[i] * c[i];

#pragma ivdep
#pragma omp simd
for (int i = 0; i < N; i++)
    a[i] = b[i] * c[i];

#include <Vc/Vc>
Vc::SimdArray<float, N> a, b, c;
a = b * c;

#include <x86intrin.h>
__m256 a, b, c;
a = _mm256_mul_ps(b, c);

asm volatile("vmulps %ymm1, %ymm0");
```

Why Create a New Library?

- ▶ Auto-vectorization not reliable across compilers
- ▶ Compiler intrinsics are not an ideal interface
 - Limited to C name mangling, portability is an issue
- ▶ Libraries do not always work well across architectures
 - No KNL support in Vc yet
 - UME::SIMD is great on KNL, but usually worse than Vc on AVX2
- ▶ Need portable solution for when no library is available
 - For example, when compiling for ARM

VecCore API

```
namespace vecCore {

template <typename T> struct TypeTraits;
template <typename T> using Mask = typename TypeTraits<T>::MaskType;
template <typename T> using Index = typename TypeTraits<T>::IndexType;
template <typename T> using Scalar = typename TypeTraits<T>::ScalarType;

// Vector Size
template <typename T> constexpr size_t VectorSize();

// Get/Set
template <typename T> Scalar<T> Get(const T &v, size_t i);
template <typename T> void Set(T &v, size_t i, Scalar<T> const val);

// Load/Store
template <typename T> void Load(T &v, Scalar<T> const *ptr);
template <typename T> void Store(T const &v, Scalar<T> *ptr);

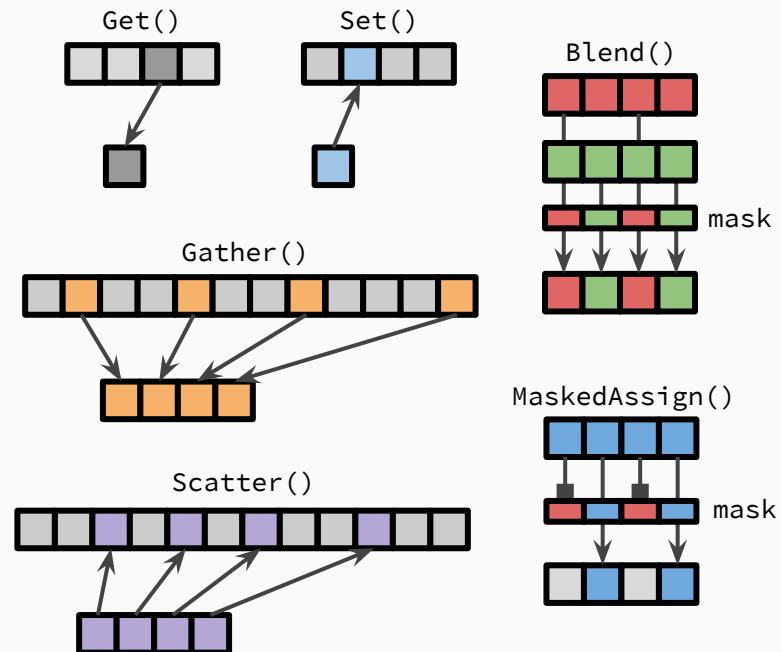
// Gather/Scatter
template <typename T, typename S = Scalar<T>>
T Gather(S const *ptr, Index<T> const &idx);

template <typename T, typename S = Scalar<T>>
void Scatter(T const &v, S *ptr, Index<T> const &idx);

// Masking/Blending
template <typename M> bool MaskFull(M const &mask);
template <typename M> bool MaskEmpty(M const &mask);

template <typename T> void MaskedAssign(T &dst, const Mask<T> &mask, const T &src);
template <typename T> T Blend(const Mask<T> &mask, const T &src1, const T &src2);

} // namespace vecCore
```

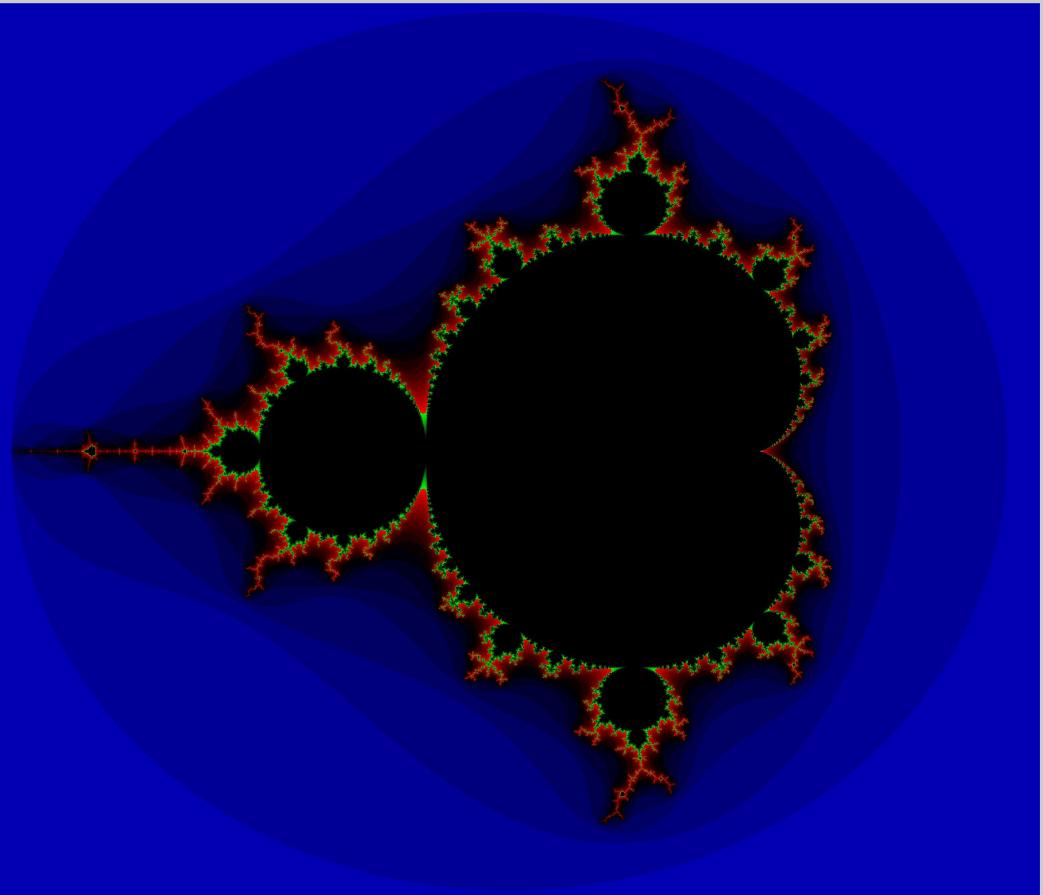


Example: Mandelbrot Set

Iterate

$$f(z) = z^2 + c$$

N times and
check if z
diverges



Example: Mandelbrot Set

Iterate

$$f(z) = z^2 + c$$

N times and
check if z
diverges

Scalar Implementation

```
template<typename T>
void mandelbrot(T xmin, T xmax, size_t nx,
                 T ymin, T ymax, size_t ny,
                 size_t max_iter, unsigned char *image)
{
    T dx = (xmax - xmin) / T(nx);
    T dy = (ymax - ymin) / T(ny);

    for (size_t i = 0; i < nx; ++i) {
        for (size_t j = 0; j < ny; ++j) {
            size_t k = 0;
            T x = xmin + T(i) * dx, cr = x, zr = x;
            T y = ymin + T(j) * dy, ci = y, zi = y;

            do {
                x = zr*zr - zi*zi + cr;
                y = 2.0 * zr*zi + ci;
                zr = x;
                zi = y;
            } while (++k < max_iter && (zr*zr+zi*zi < 4.0));

            image[ny*i+j] = k;
        }
    }
}
```

Example: Mandelbrot Set

Iterate

$$f(z) = z^2 + c$$

N times and
check if z
diverges

VecCore Implementation

```
template<typename T>
void mandelbrot_v(Scalar<T> xmin, Scalar<T> xmax, size_t nx,
                  Scalar<T> ymin, Scalar<T> ymax, size_t ny,
                  Scalar<Index<T>> max_iter, unsigned char *image)
{
    T iota;
    for (size_t i = 0; i < VectorSize<T>(); ++i) Set<T>(iota, i, i);

    T dx = T(xmax - xmin) / T(nx);
    T dy = T(ymax - ymin) / T(ny), dyv = iota * dy;

    for (size_t i = 0; i < nx; ++i) {
        for (size_t j = 0; j < ny; j += VectorSize<T>()) {
            Scalar<Index<T>> k{0};
            T x = xmin + T(i) * dx, cr = x, zr = x;
            T y = ymin + T(j) * dy + dyv, ci = y, zi = y;

            Index<T> kv{0};
            Mask<T> m{true};

            do {
                x = zr*zr - zi*zi + cr;
                y = T(2.0) * zr*zi + ci;
                MaskedAssign<T>(zr, m, x);
                MaskedAssign<T>(zi, m, y);
                MaskedAssign<Index<T>>(kv, m, ++k);
                m = zr*zr + zi*zi < T(4.0);
            } while (k < max_iter && !MaskEmpty(m));

            for (size_t k = 0; k < VectorSize<T>(); ++k)
                image[ny*i+j+k] = (unsigned char) Get(kv, k);
        }
    }
}
```

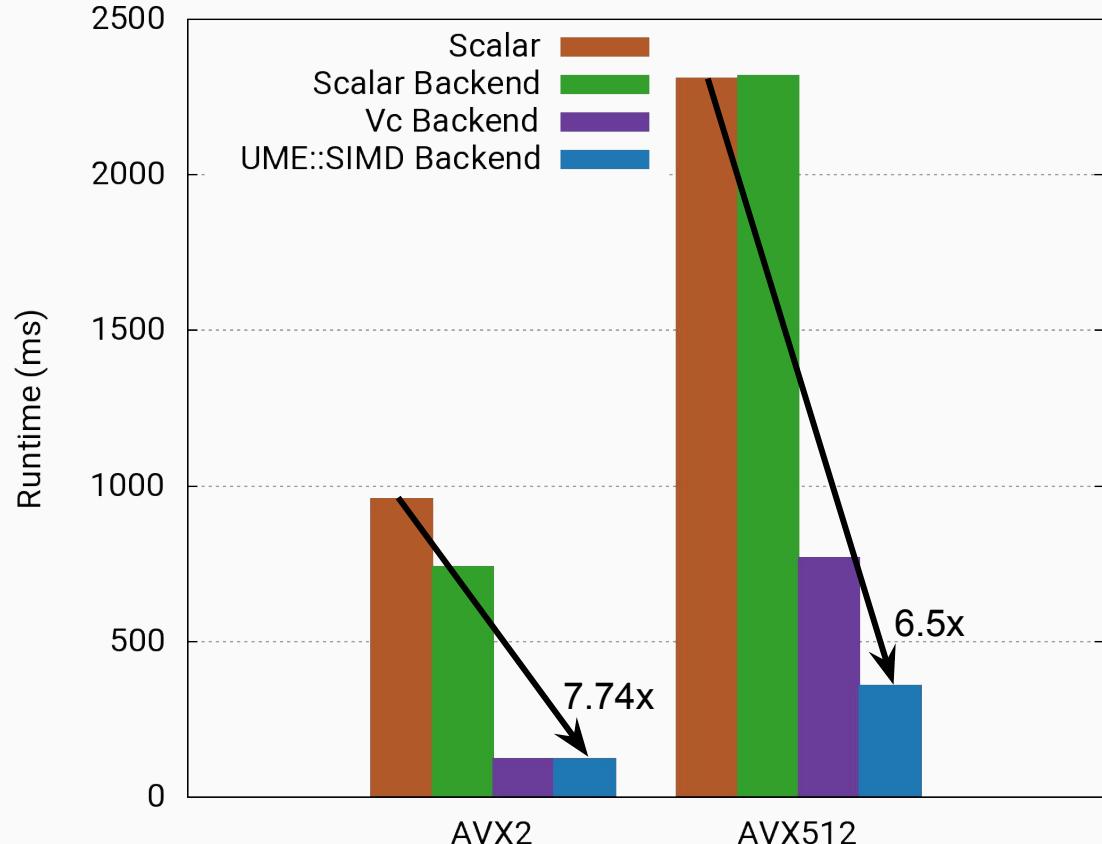
Example: Mandelbrot Set

Iterate

$$f(z) = z^2 + c$$

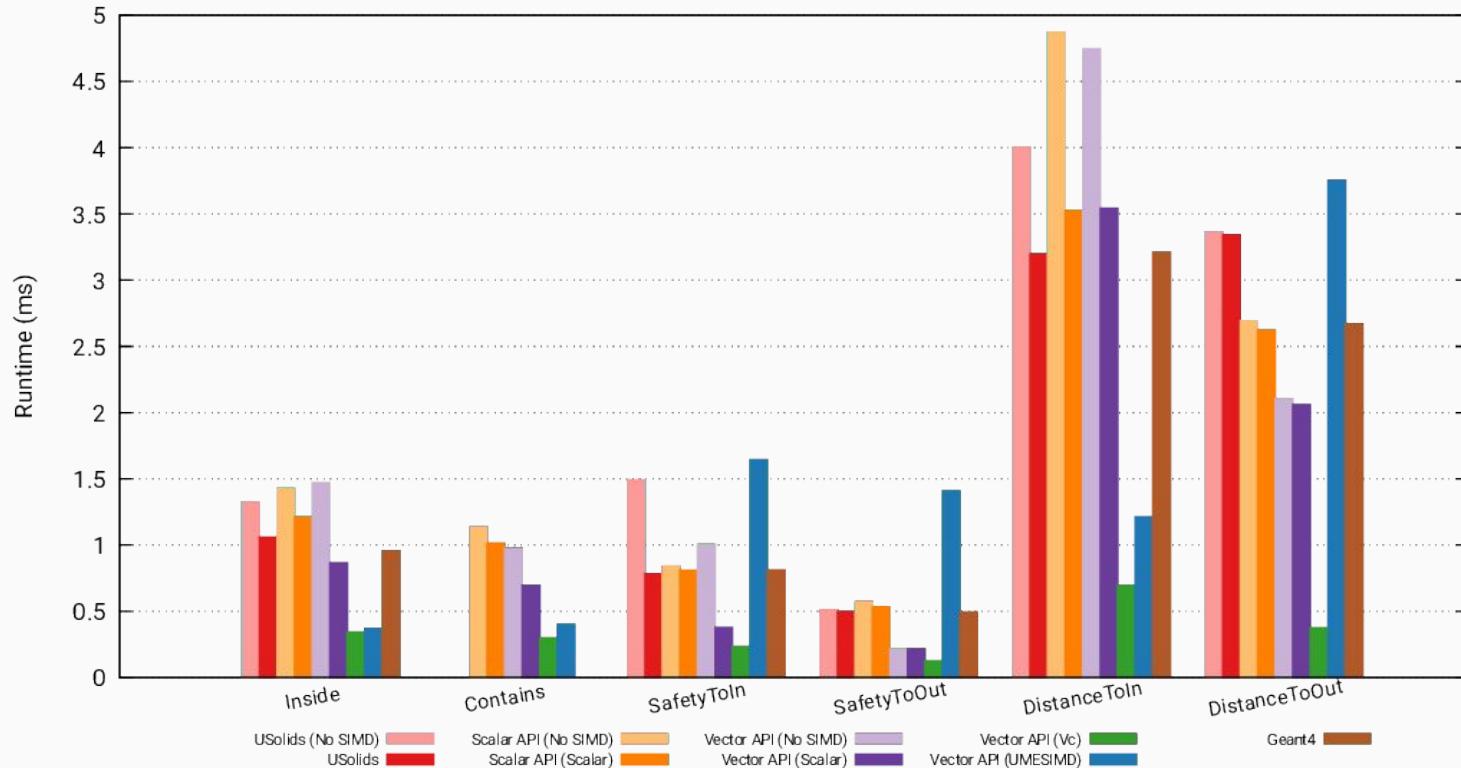
N times and
check if z
diverges

Performance with AVX2 and AVX512 using different backends

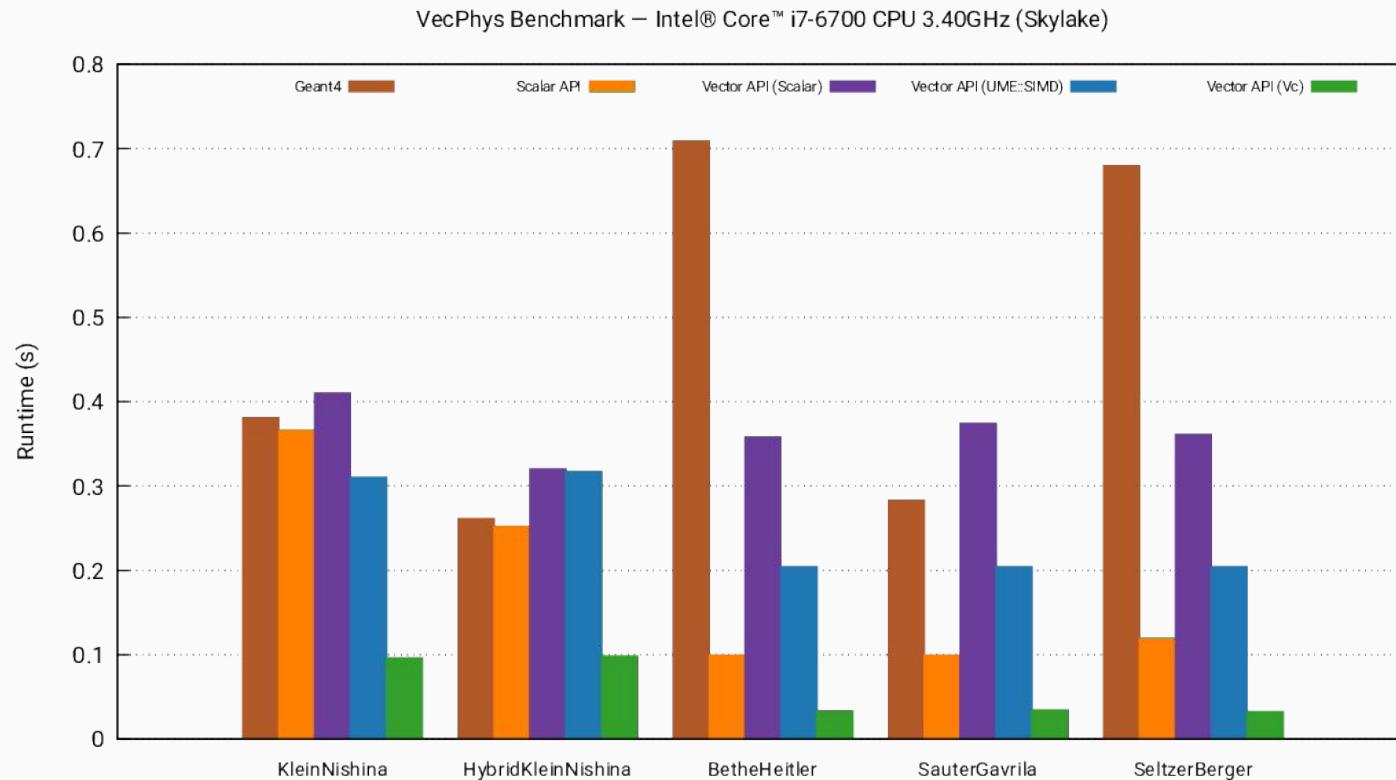


Geometry Algorithms: Box

Box Benchmark – Intel® Core™ i7-6700 CPU 3.40GHz (Skylake)



Electromagnetic Physics Models



Summary

- ▶ SIMD parallelism not easy to exploit
- ▶ Different architectures make portability a challenge
- ▶ VecCore lets users express vector code in portable way
- ▶ Multiple backends to get best performance
- ▶ New backends can get performance for free

Thank You!