Vectorisation

- Same operation on multiple data items
  -

Intel AVX/AVX2

Vectorisation example

- Can help compiler
  - Tell it loops ate inpnnt

Vectorisation example

Data alignment

- When vectorising data aligned data is essential for performance

- Unaligned data
  -

Align data

-

Multi-dimensional alignment

- Need to be careful with multi-dimensional arrays and alignment
  - If you _mm_malloc each dimension then it should be fine
  - If you do a single dimension _mm_malloc there may be issues:

```
float* a = _mm_malloc(16*15(sizeof(float), 64);
for(i=0;i<16;i++){
}
```

Gathers and Scatters

-

OpenMP SIMD directives

OpenMP SIMD clauses

- Clauses control data environment and paoitioning
- **`safelen(length)`** limits the numbe33525(o)-1.3 of ite33525(o)-1.3ations in
- **`linear(a1,a2,….)`** lists va33525(o)-1.3iables with a linea33525(o)-1.3 335 ite33525(o)-1.3ation space (ooop va33525(o)-1.3iable)o
- **`aligned(a1:base,…)`** specifies byte alignments of a list of va33525(o)-1.3iables
- **`private`**, **`lastprivate`**, **`reduction`** specify data scoping of functionality (o2853(a)-1.37670(s)-2.73224( )-0.689731(p)-1.37670(e)-1.37670(3335
-

SIMD example

```
int *loop_size;
void problem_function(float *data1, float
*data2, float *data3, int *index){
  int i,j;
  #pragm3,  s3, imd
  for i=0;i<*loop_size;++){
    j = index[i];
```

SIMD function

- Can define functions that can be called from within a vectorised loop
  - Can specify things about the function arguments
- Fortran:

SIMD function clauses

- **`simdlen(length)`** defines the vector length to be used, must be power of 2
- **`linear(a1,a2,….)`** lists variables with a linear relationship to the iteration space (loop variable)
- **`aligned(a1:base,…)`**

Cilk

- Long form

```
A[0:N] = B[0:N] + C[0:N];
D[0:N] = A[0:N] * B[0:N];
```

  - Concise

- Short form

```
for(i=0;i<N;i=i+V){
  A[i:V] = B[i:V] + C[i:V];
```

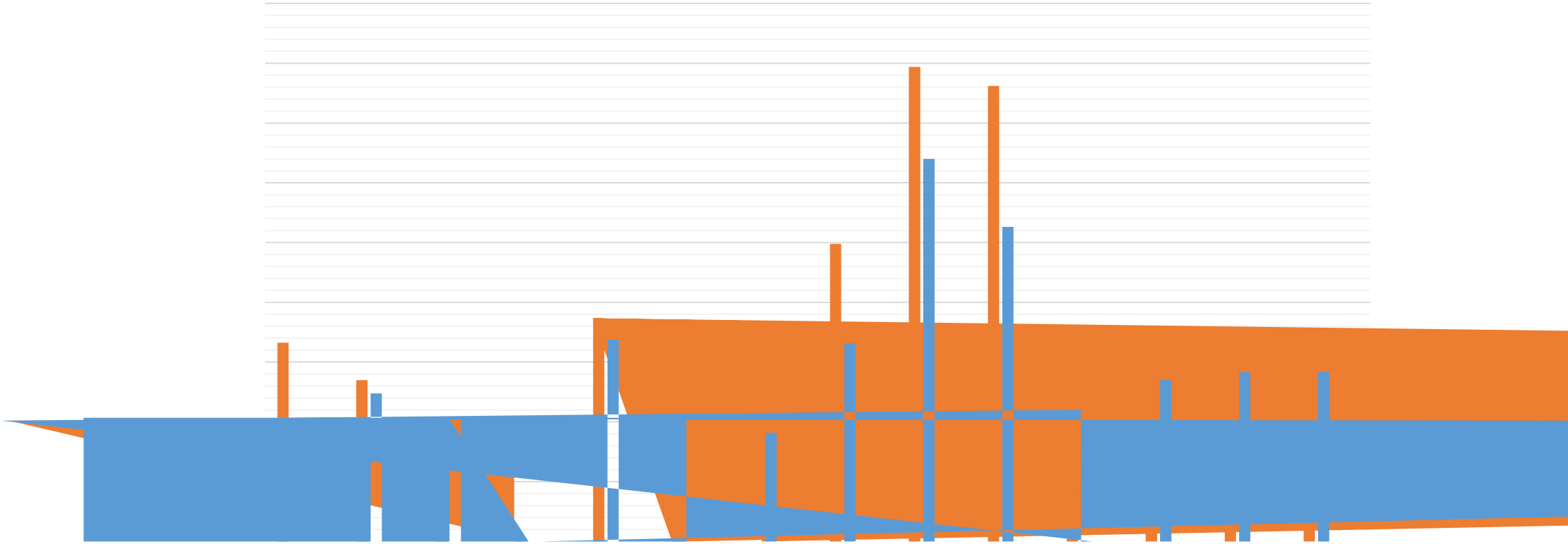
Fortran array syntax and elemental

- Standard Fortran array syntax should vectorise well

```
real, dimension(1024) :: a,b,c
!31(i)1.1931(o)1.1931(n)1.1533333 0 0 cm na,,,n,,,na,nn(:: 4):
```

m512d45 =4545 m512d45 =4545A vec =45A45 mm512 45

Comparing vectorisation performance



Summary

- Vectorisation key to performance on modern processors
  -