More Vectorization

Overview

- Aligned memory
  - Static
  - Dynamic
  - Peel/Remainder loops
- Dependencies
- Unit stride vs non-unit stride
- idioms
  - Reduction
  - Scatter/gather
- Compiler language extensions
- Compiler options for different hardware

Alignment Exercise 1

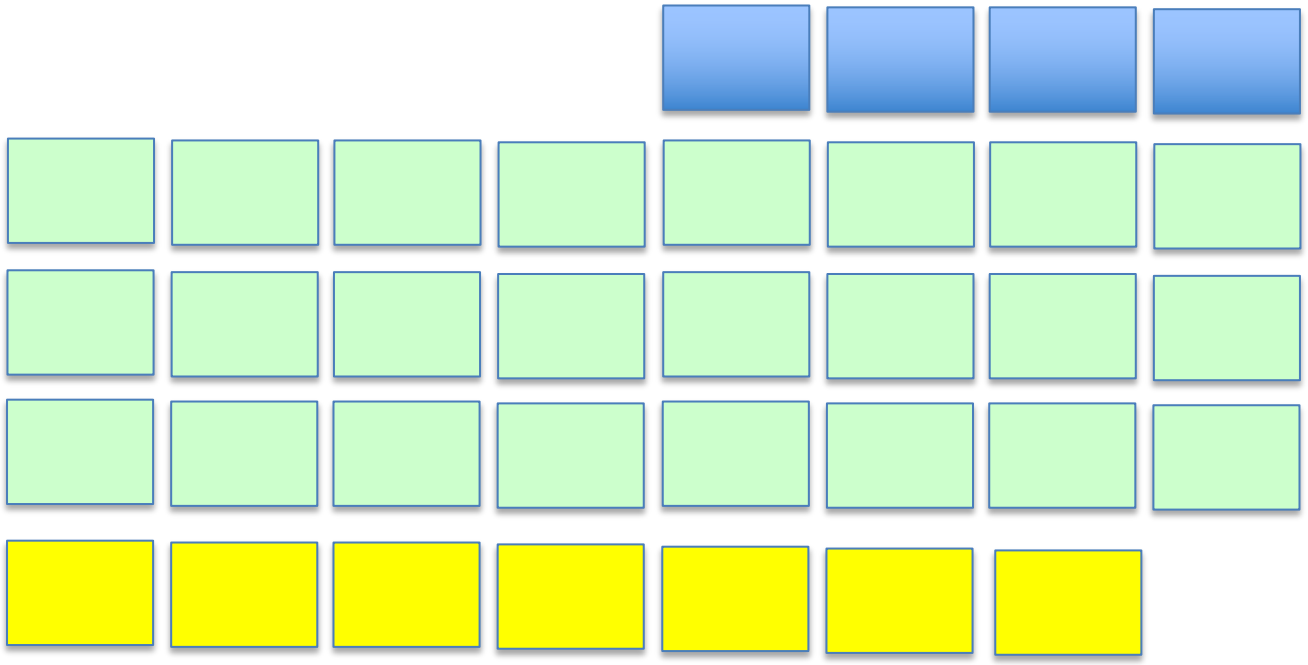
```
#define N 35

float *a, *b, *c;
void bar() {
   for (int i = 0; i < N; i++) {
      a[i] = b[i] + c[i];
   }
}

int main() {
   a = new float[N];
   b = new float[N];
   c = new float[N];
   bar();
}
```

```
$ icpc –qopt-report-phase=vec \
      –qopt-report=5 \
      –qopt-report-routine:bar \
      -fnoalias \
      Test.cpp
```

Unaligned Array



- 4 elements before getting to an aligned element
- 7 remaining elements

Compiler Report 1

- Check your output for
  - A peel loop
  - A vectorized loop
  - And a Remainder loop
- The estimated speedup: 2.730

Alignment Exercise 2

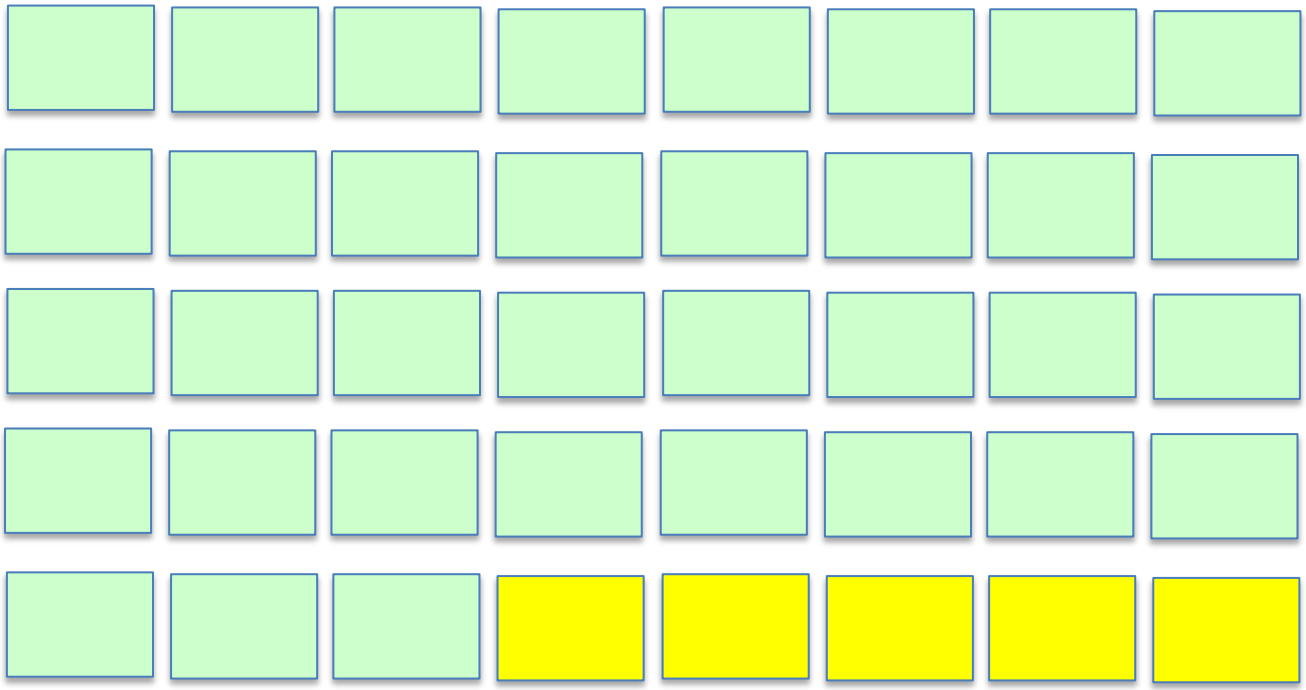
```
#define N 35
alignas(32) float *a;
alignas(32) float *b;
alignas(32) float *c;

void bar() {
   #pragma vector aligned
   for (int i = 0; i < N; i++) {
      a[i] = b[i] + c[i];
   }
}

int main() {
   a = new float[N];
   b = new float[N];
   c = new float[N];
   bar();
}
```

```
$ icpc –qopt-report-phase=vec \
      –qopt-report=5 \
      –qopt-report-routine:bar \
      -fnoalias \
      -std=c++11 \
      Test.cpp
```

Aligned Memory



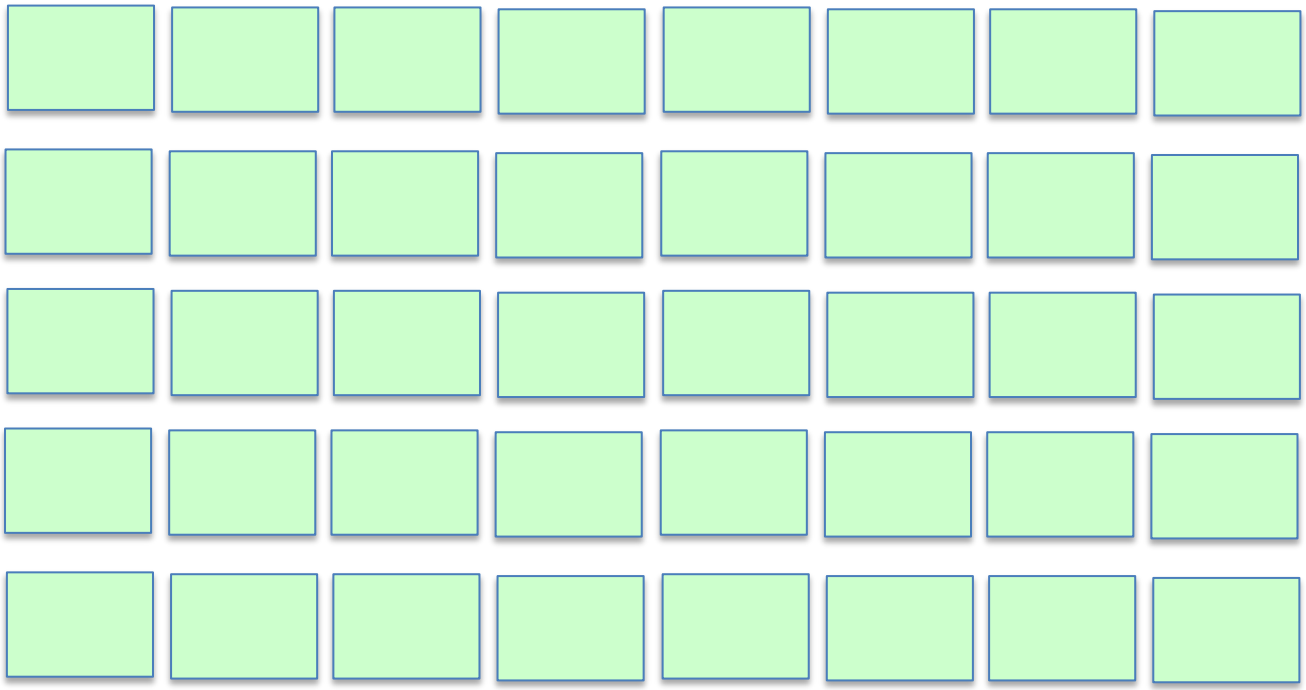
Compiler Report 2

- Check your output
- No peel loop
  - Vectorized loop and Remainder loop
- Estimated speedup: 6.700

Alignment Exercise 3

- Change size of arrays to an even multiple of vector length

Aligned Memory



Five elements of unnecessary work

Compiler Report 3

- Check output for
  - A single vectorized loop
- Estimated speedup: 14.400

Data Dependency 1

- Read-after-write (Flow dependency)

```
for (int i = 1; i < N-1; i++) {
    a[i] = a[i – 1] + 1;
}
```

- Visualize by unrolling

```
for (int i = 1; i < N-1; i = i + 2) {
    a[i] = a[i – 1] + 1;
    a[i + 1] = a[i – 1 + 1] + 1;      // a[i + 1] = a[i] + 1
}
```

Do NOT assume one vector lane will complete its operation before another lane.

Data Dependency 2

- Write-after-read (Anti-dependency)

```
for (int i = 1; i < N-1; i++) {
    a[i - 1] = a[i] + 1;
}
```

- Visualize by unrolling

```
for (int i = 1; i < N-1; i = i + 2) {
    a[i - 1] = a[i] + 1;
    a[i -1 + 1] = a[i + 1] + 1;      // a[i] = a[i + 1] + 1
}
```

- Again, do not assume one lane is faster

Data Dependency 3

- Write-after-write (Output dependency)

```
for (int i = 1; i < N-1; i++) {
    a[i - 1] = x[i];
    …
    a[i] = 2 * i;
}
```

- Cannot vectorize because incorrect results in vector mode.

Intel Compiler Language Extensions

- restrict
  - Requires compiler option -restrict
- _mm_malloc(), _mm_free()
- _assume_aligned()

- #pragma loop count *n*
- #pragma vector aligned | unaligned | always
- #pragma ivdep
- #pragma novector

- #ifdef __INTEL_COMPILER
- …
- #endif