Performance

Victor Eijkhout

Fall 2023

Justification

Programming for performance is an art.
Here are some examples.

Peak performance

- Requires all floating point units to be active
- Requires all data in L1 cache, or even register
- *Rightarrow* very hard to achieve.

Arithmetic intensity

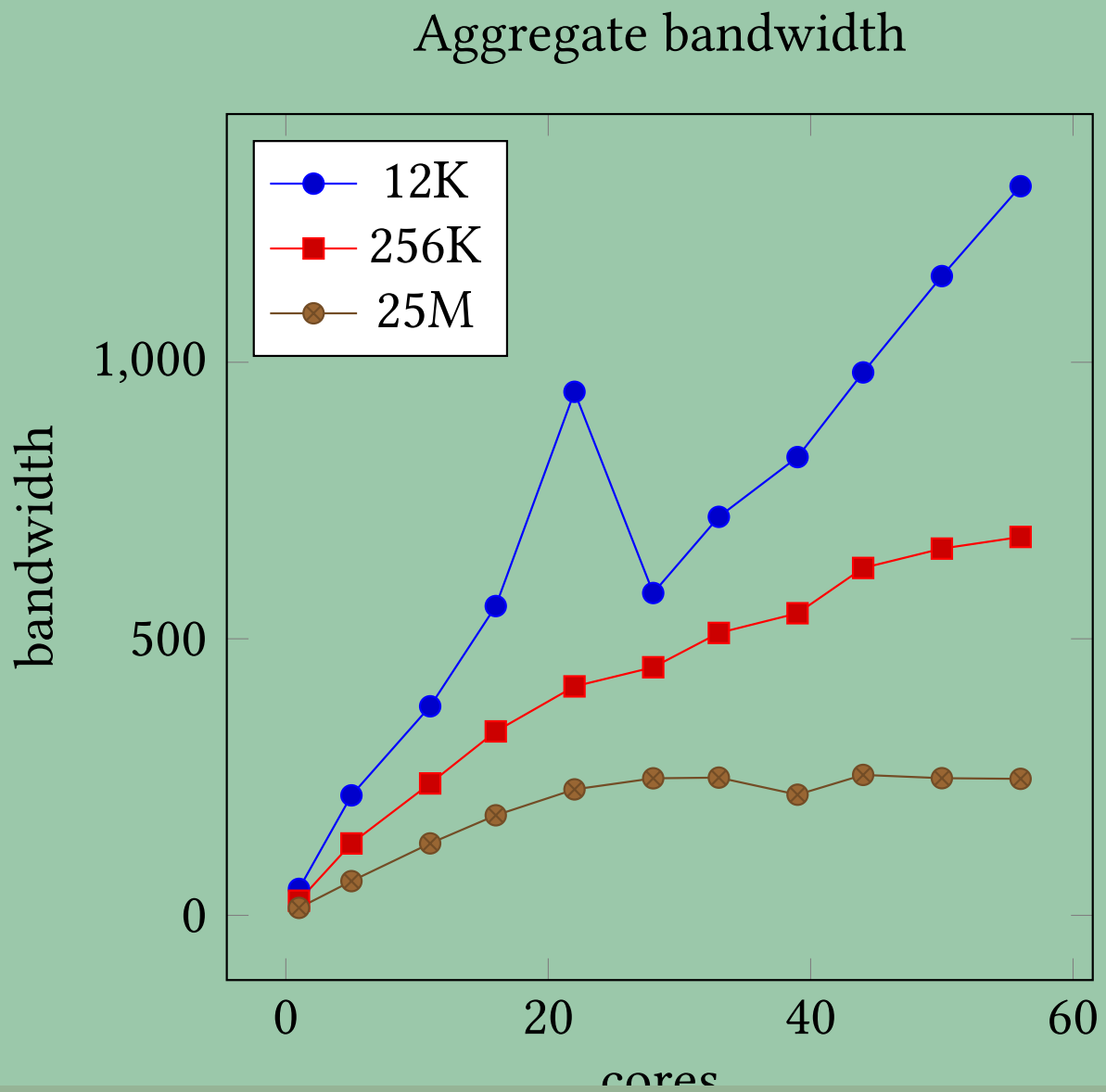
- How many operations per word?
- Equivalently: reuse factor = ratio between operations and data
- Reuse: algorithm vs implementation

Bandwidth-limited operations

- ‘Streaming’ operations

```
// bandwidth.cxx
    vector<double> results(nthreads,0.);
    for ( int t=0; t<nthreads; t++) {
      auto start_point = t*stream_length;
      threads.push_back
        ( thread( [=,&results] () {
            results[t] = memory.sumstream(
    how_many_repeats,stream_length,start_point);
          } ) );
    }
    for ( auto &t : threads )
      t.join();
```

Bandwidth measurement



Cache size effects

- Basic idea: go many times over a small data set.
- The following code is too simple:

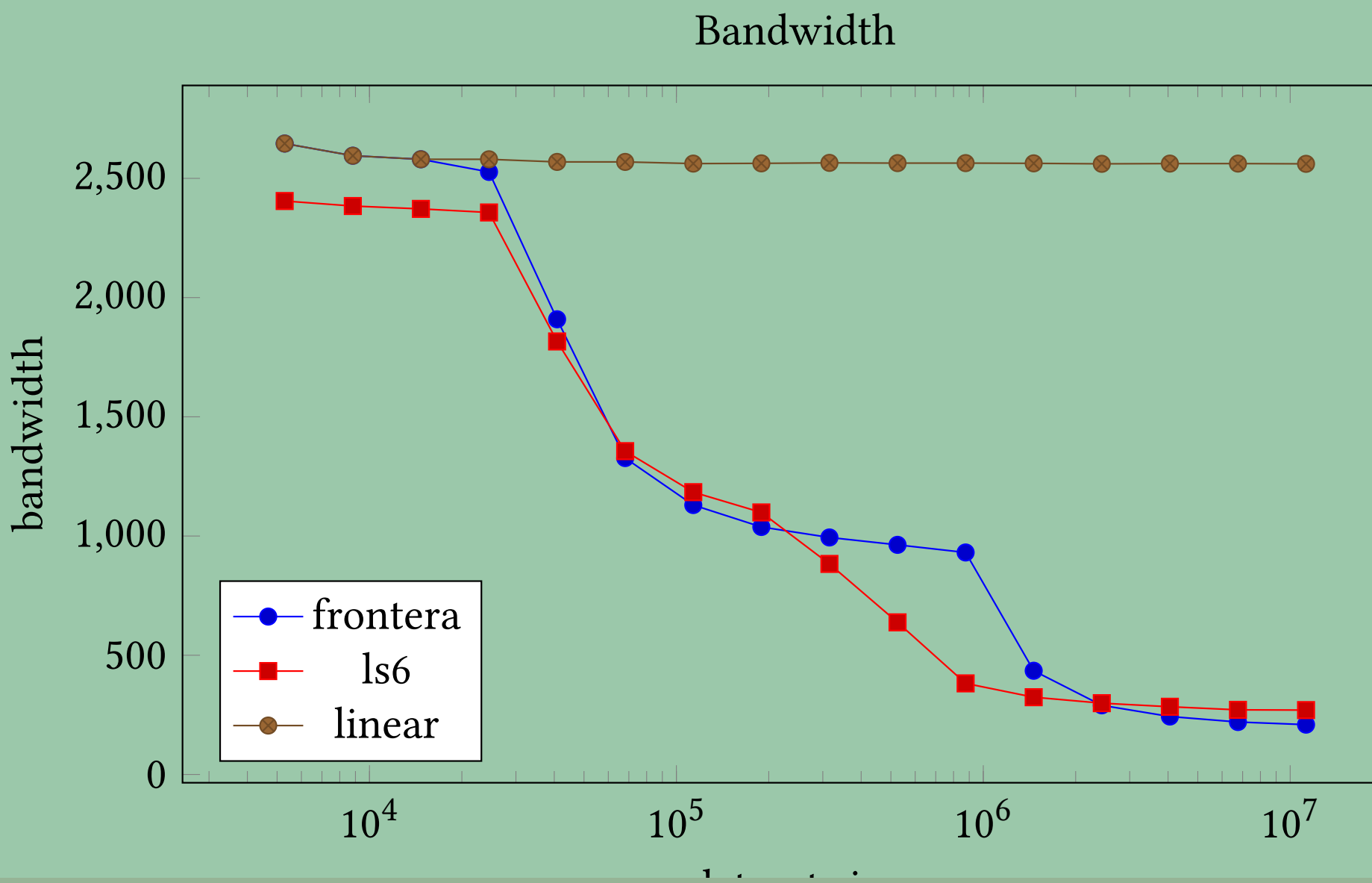
```
for (int irepeat=0; irepeat<how_many_repeats; irepeat++)
    {
  for (int iword=0; iword<cachesize_in_words; iword++)
    memory[iword] += 1.1;
}
```

Random traversal

- Emulate randomness by pointer chasing

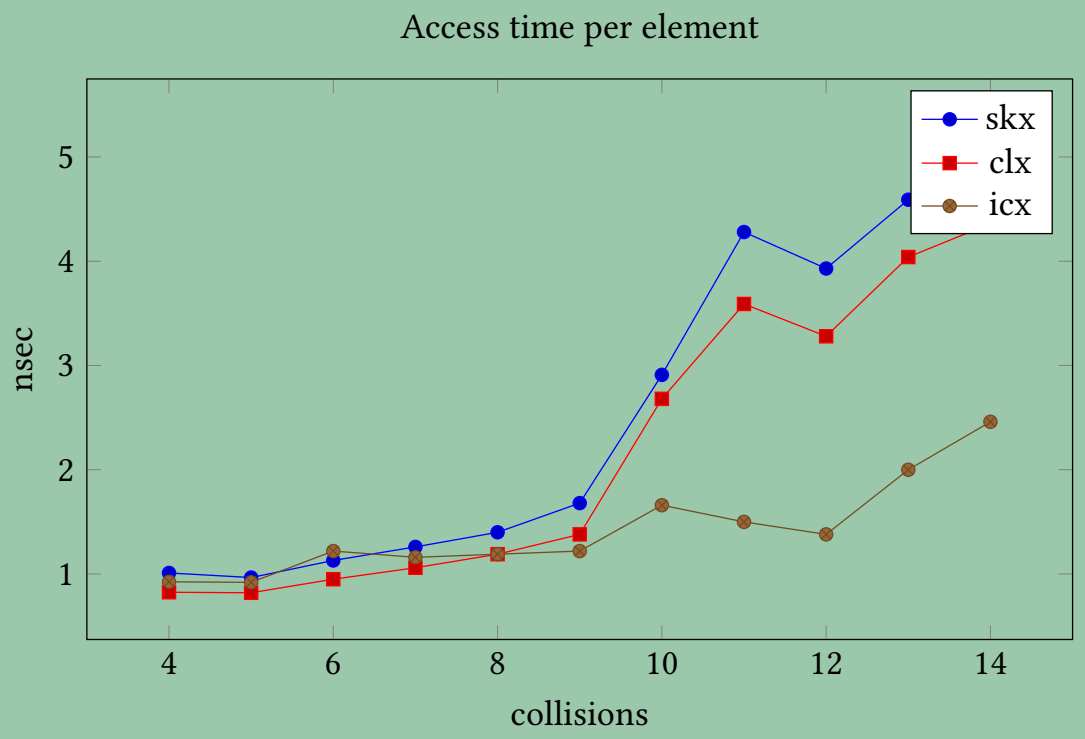
```
// setup
for (int iword=0; iword<cachesize_in_words; iword++)
    memory[iword] = (iword+1) % cachesize_in_words

// use:
ptr = 0
for (int iword=0; iword<cachesize_in_words; iword++)
    ptr = memory[ptr];
```


Bandwidth
bandwidth
2,500
2,000
1,500
1,000
500
0
10^4
10^5
10^6
10^7
frontera
ls6
linear

Associativity

- Words at certain distance map to the same associativity class
- Example: Ice Like has 48KiB cache, 12-way associative
- *Rightarrow* stride 4KiB gives conflict; 12 conflicts can be resolved
- Cascade Lake: 8-way associative



Loop tiling

- Multiple passes over array
- Rewrite as by-block
- *Rightarrow* One extra loop level

```
for (n=0; n<10; n++)
  for (i=0; i<100000; i++)
    ... = ...x[i] ...
```

```
bs = ... /* the blocksize */
for (b=0; b<100000/bs; b++)
  for (n=0; n<10; n++)
    for (i=b*bs; i<(b+1)*bs;
     i++)
      ... = ...x[i] ...
```


Example transpose

```
// regular.c
for (int i=0; i<N; i++)
  for (int j=0; j<N; j++)
    A[i][j] = B[j][i];
```

- Does this have any reuse of input or output?

Rewrite

```
// blocked.c
for (int ii=0; ii<N; ii+=blocksize)
  for (int jj=0; jj<N; jj+=blocksize)
    for (int i=ii*blocksize; i<MIN(N,(ii+1)*blocksize); i++)
      for (int j=jj*blocksize; j<MIN(N,(jj+1)*blocksize); j++)
        A[i][j] = B[j][i];
```