Optimizing Memory Traffic by Improving the Temporal and Spatial Locality of Data Access

Temporal Locality - Recently references items are likely to be referenced in the near future.
E.g. instructions in a loop, or data variables referenced in a loop
Spatial Locality - Items with nearby addresses tend to be referenced close together in time.
E.g array elements a[i] and a[i+1] tend to be referenced in sequence, or instructions like "pop %ebp" and "ret"

Exploiting the fact that we access the array sequentially and they are stored 'side-by-side' in main memory, so we access multiple values and have high hit rates.

```
Sum = 0;
for(i = 0; i < n; i++)
        Sum += 1[i];
Return sum;
```

Stride-1 pattern(sequential in row-major order(row of one, single dimensional matrix))

2-Dimensional Blocking
Organize your application into chunks that fit into L1 cache and are read/written together.
Divide matrix into blocks, cache them, multiply block by block, then discard.

Loop unrolling:

```
Int x[100]
for(i=0;1<100;i++)
        X[i] = 0
#loops = 100
#instr per loop: 5
```

Unrolled:

```
Int x[100]
for(i=0;1<100;i+=2)
        X[i] = 0
        X[i+1] = 0
#loops = 50
#instr per loop: 8
```

Further:

```
Int x[100]
for(i=0;1<100;i+=4)
        X[i] = 0
        X[i+1] = 0
        X[i+2] = 0
```

X[i+3] = 0
#loops = 25
#instr per loop: 14

Number of total instructions: 500, 400, 350

-find the most appropriate loop step for best optimization, there is an upper bound.Unrolling too many times can lead to excessive register use.

Loop Blocking:
Similar to loop unrolling, trying to get better cache performance.

```
for(int i =0; i<n i++)
    for(int j=0; j<n; j++)
```

Two matrices, a and b
Take entire row of a and multiply by col of b
Inner-most loop, k is the only one that is changing.
Use spatial locality to advantage

Smooth function strategy:

Smooth function takes one element and takes neighboring pixels and set to the mean color value.
Loop through and call smoothing function, add changes to matrix - lots of function overhead setting input values each time. *Maybe incorporate function code instead of calling func every time.