**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.