CENG331
Performance Lab Recitation

Fall 2024

Outline

Introduction

- Normalization
- Kronecker Product

Optimization

- Code motion
- Avoiding costly operations
- Reducing sequential dependency
- Loop unrolling
- Writing cache friendly code

CODE MOTION

```
int i,j;
for (i = 0; i < N; i++){
      for (j = 0; j < N; j++)
            a[N*i+j] = b[j];
}
```

```
int i,j,ni;
for (i = 0; i < N; i++){
      ni = N*i;
      for (j = 0; j < N; j++)
            a[ni+j] = b[j];
}
```

AVOIDING COSTLY OPERATIONS

```
int i,j;
for (i = 0; i < N; i++){
      ni = N*i;
      for (j = 0; j < N; j++)
            a[ni+j] = b[j];
}
```

```
int i,j,ni;
ni = 0;
for (i = 0; i < N; i++){
      for (j = 0; j < N; j++)
            a[ni+j] = b[j];
       ni += N;
}
```

LOOP UNROLLING

```
int i,sum;
sum = 0;
for (i = 0; i < N; i++){
        sum += a[i];
}
```

```
int i,sum;
sum = 0;
for (i = 0; i < N; i+=2){
        sum += a[i]+a[i+1];
}
```

REDUCING SEQUENTIAL DEPENDENCY

```
int i,sum;
sum = 0;
for (i = 0; i < N; i+=2){
      sum += a[i]+a[i+1];
}
```

```
int i,sum,s1,s2;
s1 = 0;
s2 = 0;
for (i = 0; i < N; i+=2){
      s1 += a[i];
      s2 += a[i+1];
}
sum = s1+s2;
```

WRITING CACHE FRIENDLY CODE I

For spatial locality:

```
int i,j,sum;
sum = 0;
for (i = 0; i < N; i++){
    for (j = 0; j < N; j++)
        sum += a[j][i];
}
```

```
int i,j,sum;
sum = 0;
for (i = 0; i < N; i++){
    for (j = 0; j < N; j++)
        sum += a[i][j];
}
```

WRITING CACHE FRIENDLY CODE II

For temporal locality:

```
int i,j,k;
for (i = 0; i < N; i++){
    for (j = 0; j < N; j++)
        for (k = 0; k < N; k++)
            c[i*N+j] += a[i*N+k] * b[k*N+j];
}
```

```
int i,j,k,i1,j1,k1;
for (i = 0; i < N; i+=B){
    for (j = 0; j < N; j+=B)
        for (k = 0; k < N; k+=B)
            for (i1 = i; i1 < i+B; i++)
                for (j1 = j; j1 < j+B; j++)
                    for (k1 = k; k1 < k+B; k++)
                        c[i1*N+j1] += a[i1*N+k1] * b[k1*N+j1];
}
```

Writing cache friendly code II - Cont.

Blocking

$$\begin{bmatrix} C1 & C2 & C3 \\ C4 & C5 & C6 \\ C7 & C8 & C9 \end{bmatrix} = \begin{bmatrix} A1 & A2 & A3 \\ A4 & A5 & A6 \\ A7 & A8 & A9 \end{bmatrix} \begin{bmatrix} B1 & B2 & B3 \\ B4 & B5 & B6 \\ B7 & B8 & B9 \end{bmatrix}$$

$$C1 = A1 * B1 + A2 * B4 + A3 * B7$$
$$C2 = A1 * B2 + A2 * B5 + A3 * B8$$
$$C3 = A1 * B3 + A2 * B6 + A3 * B9$$
$$\vdots$$