[Chap.6-4] The Memory Hierarchy

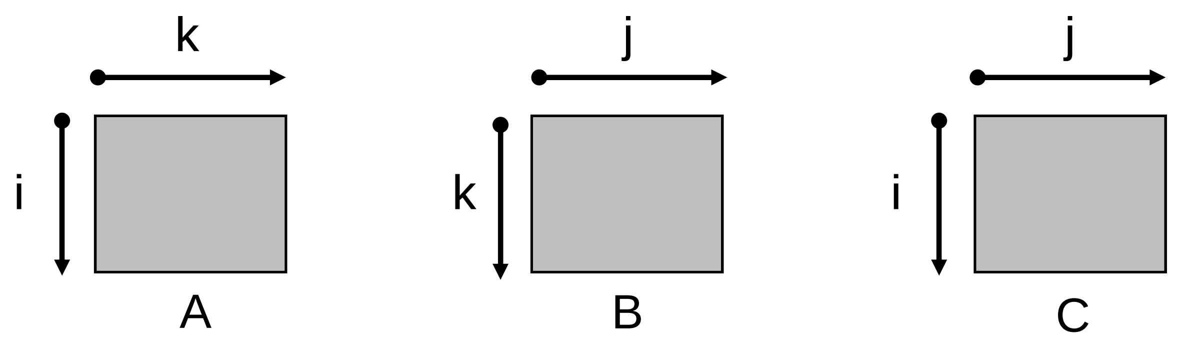
Young Ik Eom (yieom@skku.edu, 031-290-7120)
Distributing Computing Laboratory
Sungkyunkwan University
http://dclab.skku.ac.kr

Contents

- Storage technologies
- Locality
- Memory hierarchy
- Cache memories
- Writing cache-friendly code
- Impact of caches on program performance

Example: Rearranging Loops

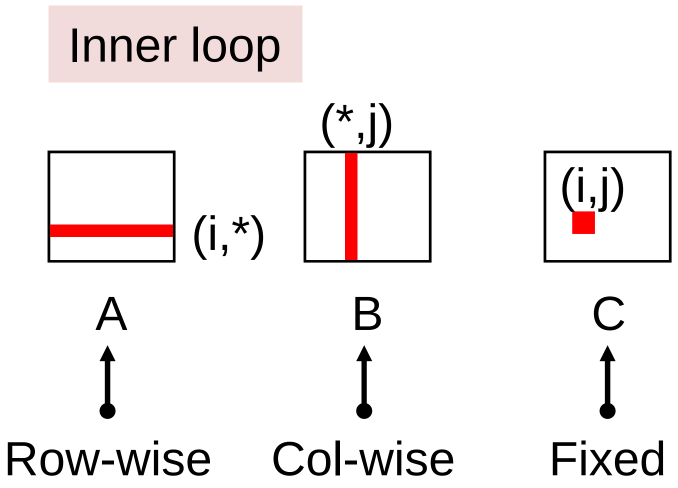
- **Rearranging loops to increase spatial locality (Matrix multiplication)**
  - Assumptions
    - **n**×**n** array of double, sizeof(double) = 8
    - Single cache with 32B block size
    - The array size **n** is so large
      (Single matrix row does not fit in the cache)
    - Local variables in registers
  - Access pattern



Example: Rearranging Loops

- Ordinary code for matrix multiplication

```
/* matrix multiplication */
for (i=0; i<n; i++)  {
  for (j=0; j<n; j++) {
    c[i][j] = 0.0;
    for (k=0; k<n; k++)
      c[i][j] += a[i][k] * b[k][j];
  }
}
```

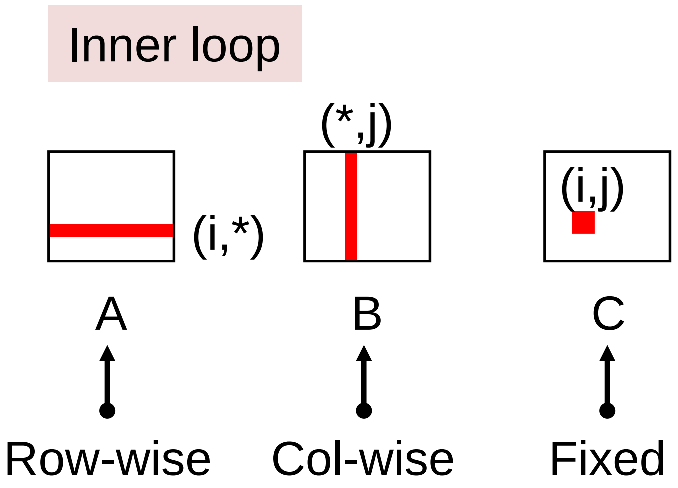



```
c[i][j] = c[i][j] + a[i][k] * b[k][j];
```

Example: Rearranging Loops

- Rearranging loops to increase spatial locality (Matrix multiplication: version-ijk)

```
/* ijk */
for (i=0; i<n; i++)  {
  for (j=0; j<n; j++) {
    sum = 0.0;
    for (k=0; k<n; k++)
      sum += a[i][k] * b[k][j];
    c[i][j] = sum;
  }
}
```



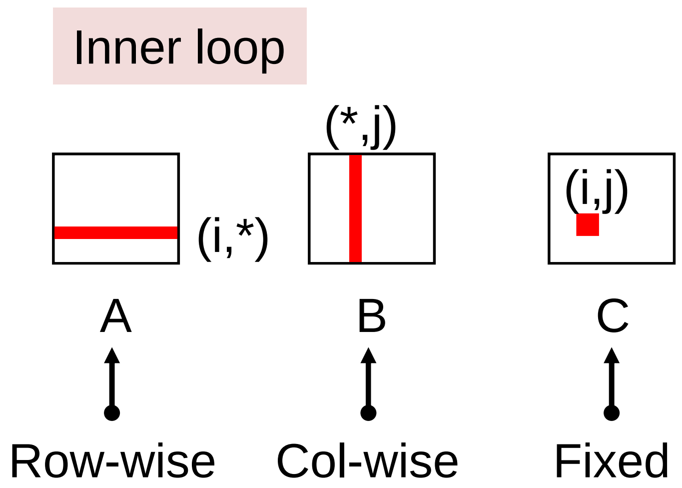

```
c[i][j] = c[i][j] + a[i][k] * b[k][j];
```

- Miss rates per inner loop iteration
  - a (0.25), b (1.00), c (0.00)

Example: Rearranging Loops

- Rearranging loops to increase spatial locality (Matrix multiplication: version-jik)

```
/* jik */
for (j=0; j<n; j++) {
  for (i=0; i<n; i++) {
    sum = 0.0;
    for (k=0; k<n; k++)
      sum += a[i][k] * b[k][j];
    c[i][j] = sum;
  }
}
```



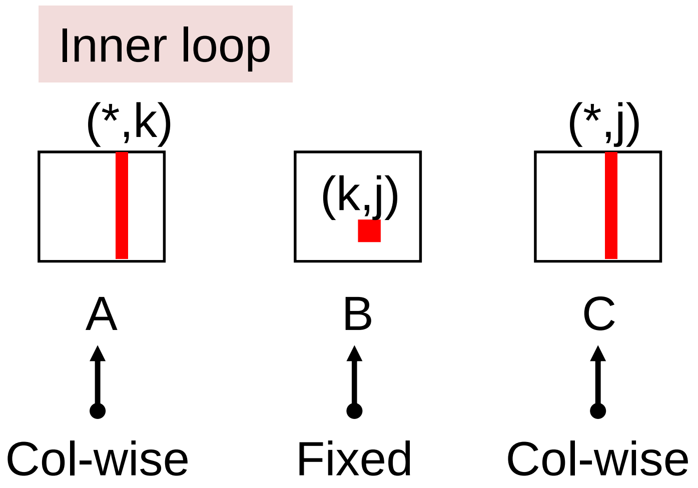

```
c[i][j] = c[i][j] + a[i][k] * b[k][j];
```

- Miss rates per inner loop iteration
  - a (0.25), b (1.00), c (0.00)

Example: Rearranging Loops

- Rearranging loops to increase spatial locality (Matrix multiplication: version-jki)

```
/* jki */
for (j=0; j<n; j++) {
  for (k=0; k<n; k++) {
    r = b[k][j];
    for (i=0; i<n; i++)
      c[i][j] += a[i][k] * r;
  }
}
```



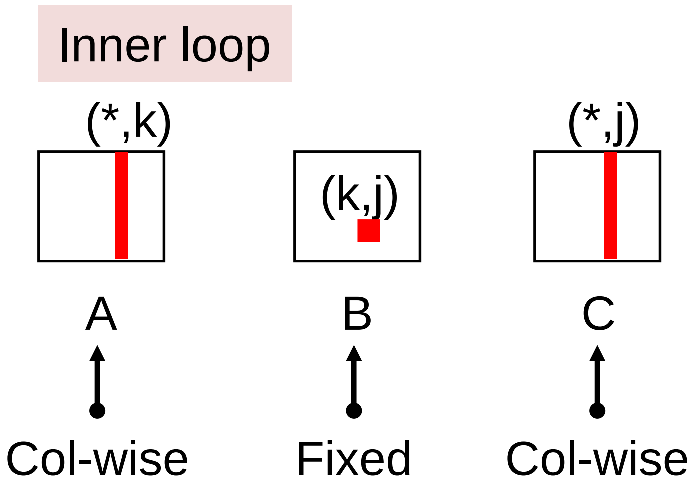

```
c[i][j] = c[i][j] + a[i][k] * b[k][j];
```

- Miss rates per inner loop iteration
  - a (1.00), b (0.00), c (1.00)

Example: Rearranging Loops

- **Rearranging loops to increase spatial locality (Matrix multiplication: version-kji)**

```
/* kji */
for (k=0; k<n; k++) {
  for (j=0; j<n; j++) {
    r = b[k][j];
    for (i=0; i<n; i++)
      c[i][j] += a[i][k] * r;
  }
}
```




```
c[i][j] = c[i][j] + a[i][k] * b[k][j];
```

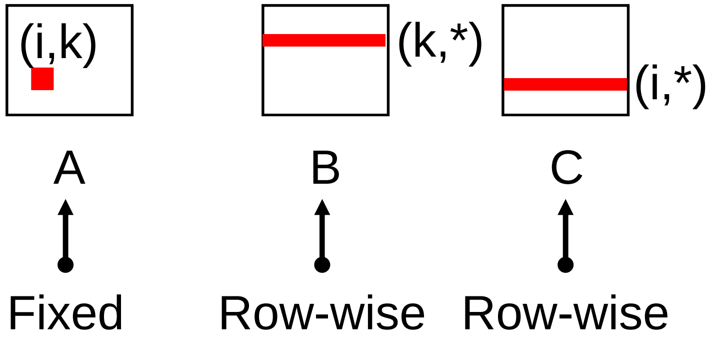
- Miss rates per inner loop iteration
  - a (1.00), b (0.00), c (1.00)

Example: Rearranging Loops

- Rearranging loops to increase spatial locality (Matrix multiplication: version-kij)

```
/* kij */
for (k=0; k<n; k++) {
  for (i=0; i<n; i++) {
    r = a[i][k];
    for (j=0; j<n; j++)
      c[i][j] += r * b[k][j];
  }
}
```

Inner loop




```
c[i][j] = c[i][j] + a[i][k] * b[k][j];
```

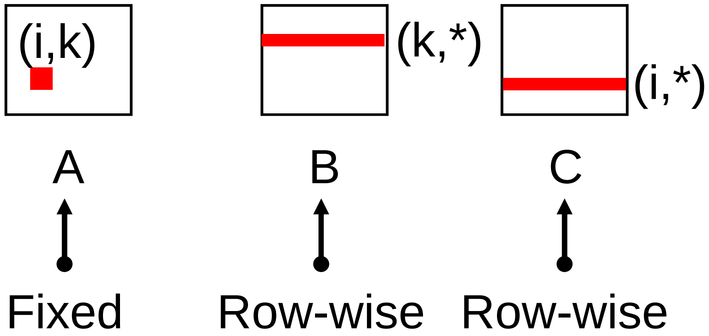
- Miss rates per inner loop iteration
  - a (0.00), b (0.25), c (0.25)

Example: Rearranging Loops

- Rearranging loops to increase spatial locality (Matrix multiplication: version-ikj)

```
/* ikj */
for (i=0; i<n; i++) {
  for (k=0; k<n; k++) {
    r = a[i][k];
    for (j=0; j<n; j++)
      c[i][j] += r * b[k][j];
  }
}
```

Inner loop




```
c[i][j] = c[i][j] + a[i][k] * b[k][j];
```

- Miss rates per inner loop iteration
  - a (0.00), b (0.25), c (0.25)

Example: Rearranging Loops

- Rearranging loops to increase spatial locality (Matrix multiplication: summary)

```
for (i=0; i<n; i++) {
  for (j=0; j<n; j++) {
   sum = 0.0;
   for (k=0; k<n; k++)
     sum += a[i][k] * b[k][j];
   c[i][j] = sum;
 }
}
```

- ijk (& jik):
  - 2 loads, 0 stores
  - miss-ratio/iter = 1.25

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

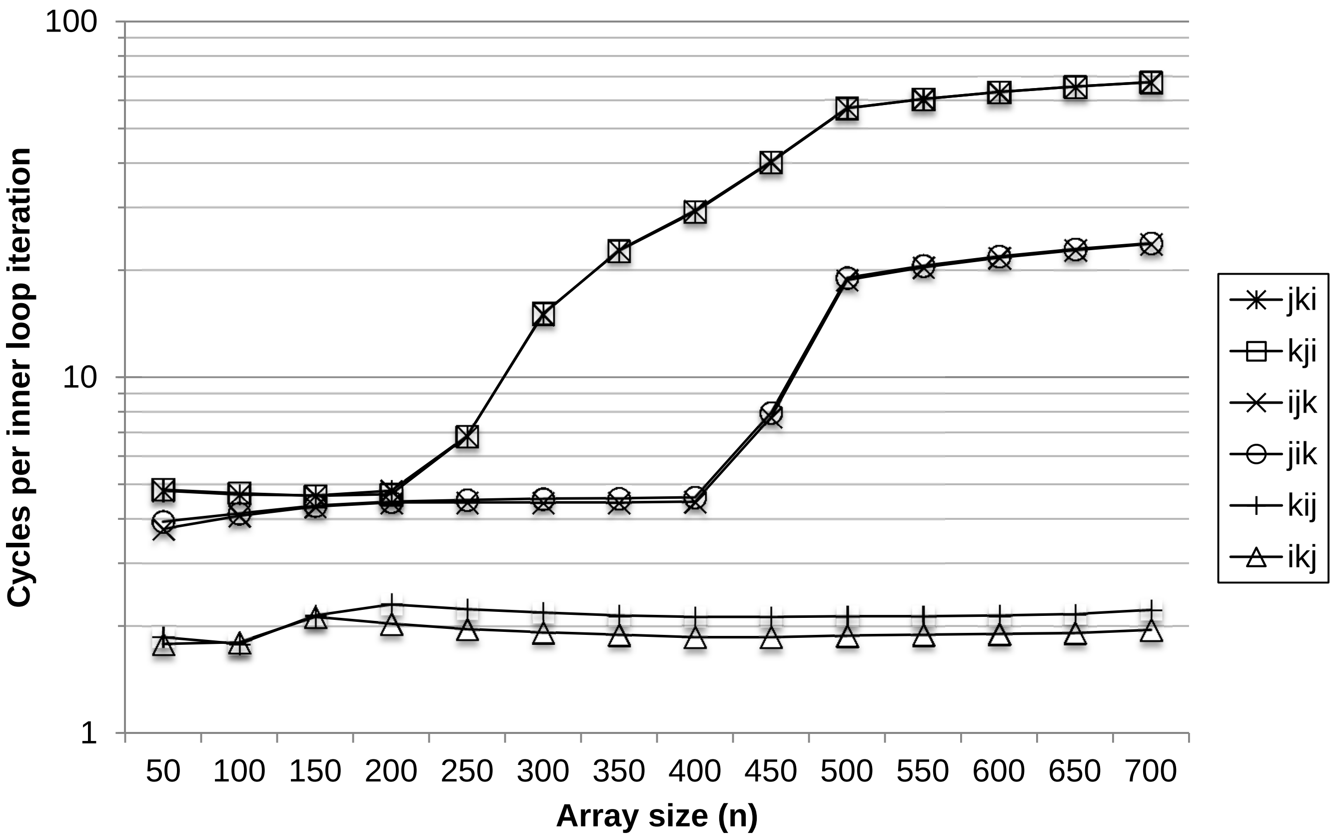
- jki (& kji):
  - 2 loads, 1 store
  - miss-ratio/iter = 2.0

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

- kij (& ikj):
  - 2 loads, 1 store
  - miss-ratio/iter = 0.5

Example: Rearranging Loops

- Rearranging loops to increase spatial locality (Matrix multiplication: summary)



Example: Rearranging Loops

- **Rearranging loops to increase spatial locality (Matrix multiplication: some points)**
  - For large values of **n**,
    the fastest version runs almost 40 times faster
    than the slowest version,
    <u>even though they perform the same # of FP operations</u>
  - Miss ratio is a better predictor of performance
    than the total # of memory references (in this case)
  - For large values of **n**,
    the performance of the fastest pair of versions is constant
  - Etc.

Exploiting Locality in Pgms

- Programs with good locality
  - Access most of their data from fast cache memories
- Programs with poor locality
  - Access most of their data from slow DRAM memories

- Programmers must exploit this to write more efficient programs

Tips

- Focus your attention on the inner loops
- Try to maximize the spatial locality in your programs
  - Reading data objects sequentially, with stride-1
- Try to maximize the temporal locality in your programs
  - Using data objects repeatedly once they are read from memory

Summary