Cache Lab

MILOZMS

问题分析

- s=5 E=1 b=5
- 每个set能存储32字节——8个整型数据
- 64*64的整型数组
- 数组每行64*4字节
- 沿列方向访问时每4行会发生一次冲突：A[i][j]和A[i+4k][j]

Set index	Data
0	A[0][0]-A[0][7], A[4][0]-A[4][7], A[8][0]-A[8][7],.....
1	A[0][8]-A[0][15], A[4][8]-A[4][15], A[8][8]-A[8][15],.....
2	A[0][16]-A[0][23],.....
3	A[0][24]-A[0][31],.....
4	A[0][32]-A[0][39],.....
5	A[0][40]-A[0][47],.....
6	A[0][48]-A[0][55],.....
7	A[0][56]-A[0][63],.....
....	....

```c
char trans_desc[] = "Simple row-wise scan transpose";
void trans(int M, int N, int A[N][M], int B[M][N])
{
    int i, j, tmp;

    REQUIRES(M > 0);
    REQUIRES(N > 0);

    for (i = 0; i < N; i++) {
        for (j = 0; j < M; j++) {
            tmp = A[i][j];
            B[j][i] = tmp;
        }
    }

    ENSURES(is_transpose(M, N, A, B));
}
```

trace.f0

```
 1  S 0068710c,1
 2  L 00687120,8
 3  L 00687104,4
 4  L 00687100,4
 5  L 00607100,4
 6  S 00647100,4
 7  L 00607104,4
 8  S 00647200,4
 9  L 00607108,4
10  S 00647300,4
11  L 0060710c,4
12  S 00647400,4
13  L 00607110,4
14  S 00647500,4
15  L 00607114,4
16  S 00647600,4
17  L 00607118,4
18  S 00647700,4
19  L 0060711c,4
20  S 00647800,4
21  L 00607120,4
```

trace.f0

S 0068710c,1 L 00687120,8 L 00687104,4 L 00687100,4

L 0060710**0**,4 S 0064710**1**,4

L 0060710**4**,4 S 0064720**2**,4

L 0060710**8**,4 S 0064730**3**,4

L 0060710**c**,4 S 0064740**4**,4

L 0060711**0**,4 S 0064750**5**,4

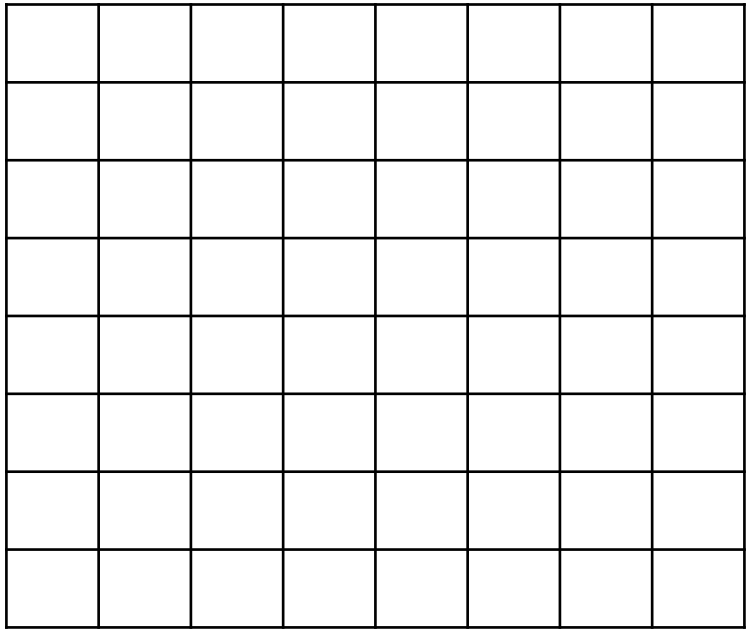
L 0060711**4**,4 S 0064760**6**,4

……

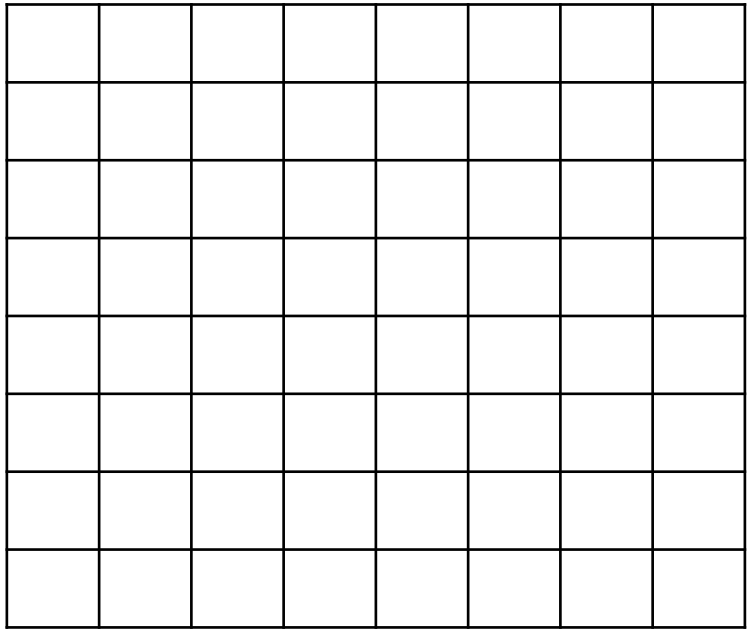
问题分析

- s=5 E=1 b=5
- 每个set能存储32字节——8个整型数据
- 64*64的整型数组
- 数组每行64*4字节
- 沿列方向访问时每4行会发生一次冲突：A[i][j]和A[i+4k][j]
- A[i][j]和B[i+4k][j]也会冲突
- 若横向读取A，纵向写入B，则Cache中只能同时存在B中的4行
- 最多只有8到10个临时变量能用

Blocking: 8*8



A[i][j]



B[j][i]

问题分析

- 假设i，j为8的倍数(0,8,16,...,56)
- 当i = j时，A[i][j]与B[j][i]冲突
- 当i != j时
- &B[j][i]-&A[i][j]=64*j + i+ 64*64 – (64*i + j) = 63*(j – i) + 4096
- A[i][j]与B[j][i]不冲突

Blocking: 8*8

0	1	2	3	4	5	6	7

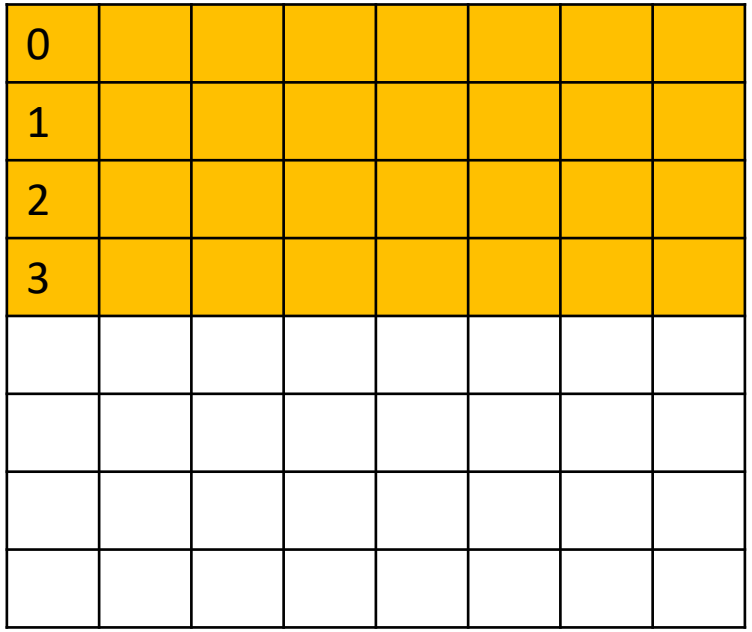
A[i][j]

0							
1							
2							
3							
4							
5							
6							
7							

B[j][i]

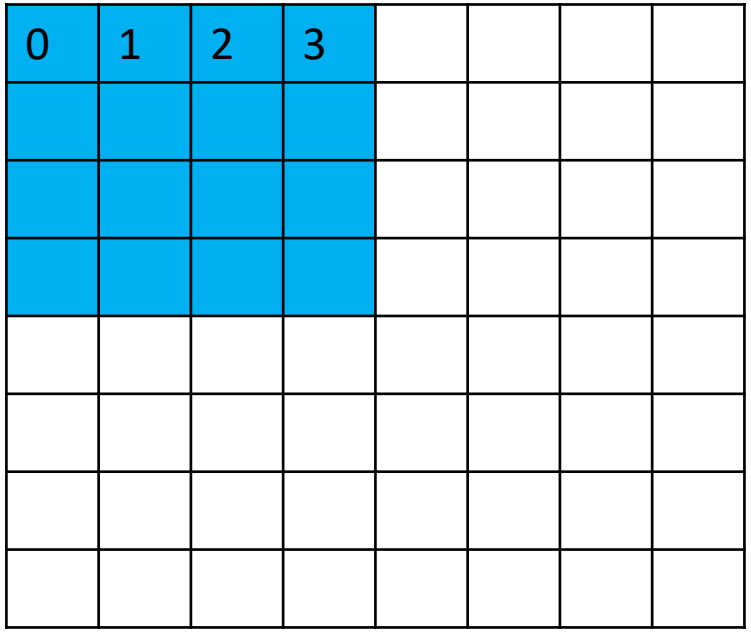
Blocking: 8*8

0	1	2	3	4	5	6	7

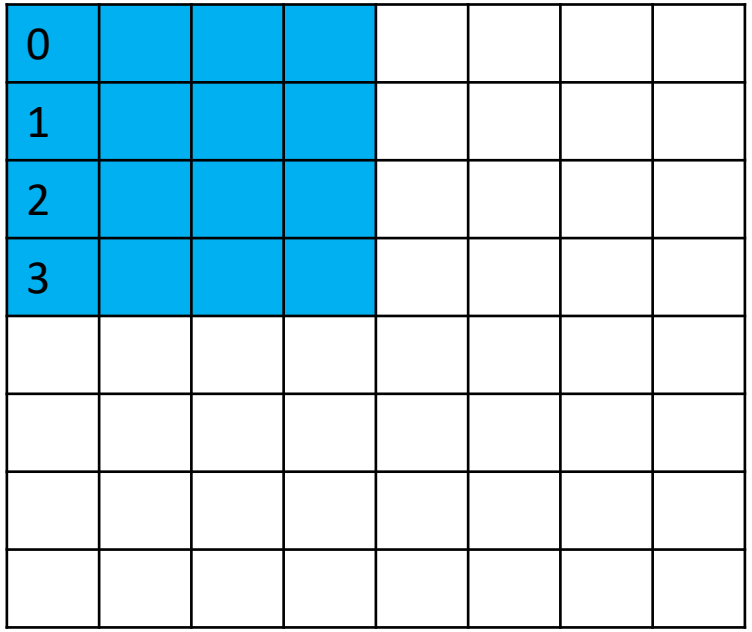


关键：充分利用每一次load

Step 1

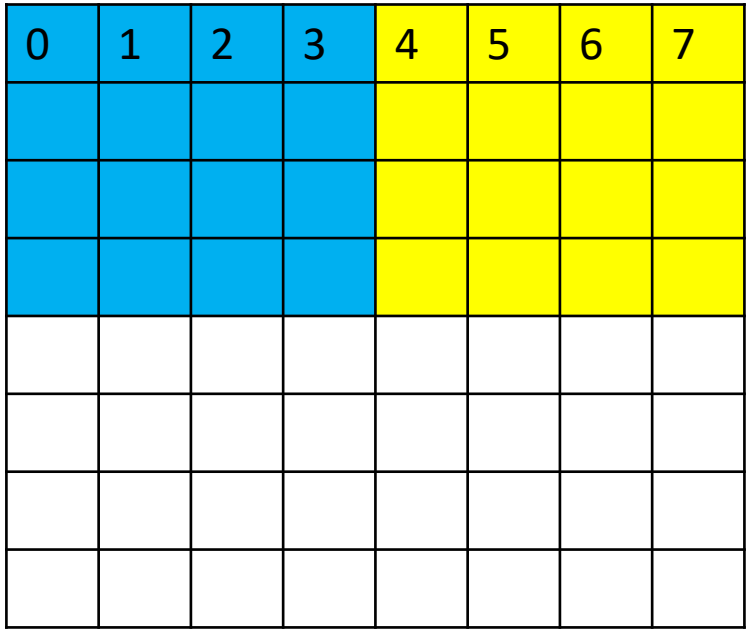


A[i][j]

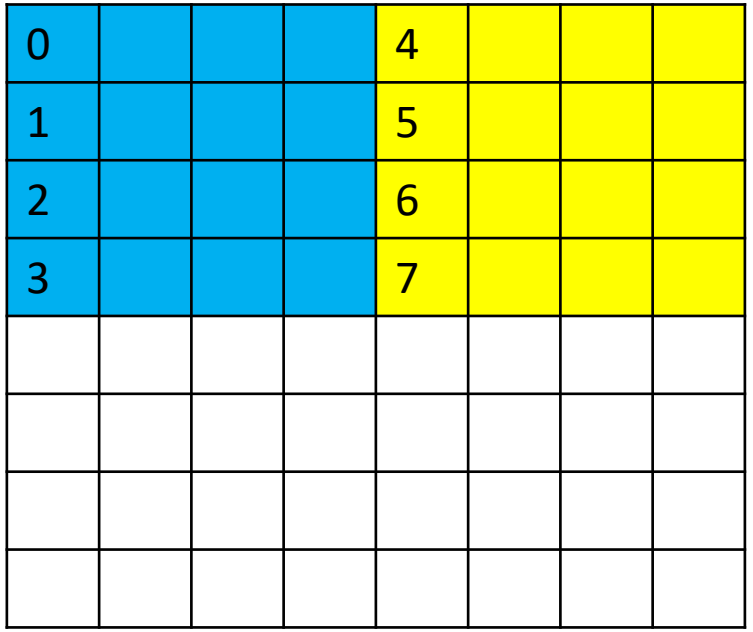


B[j][i]

Step 2



A[i][j]



B[j][i]

Step 3

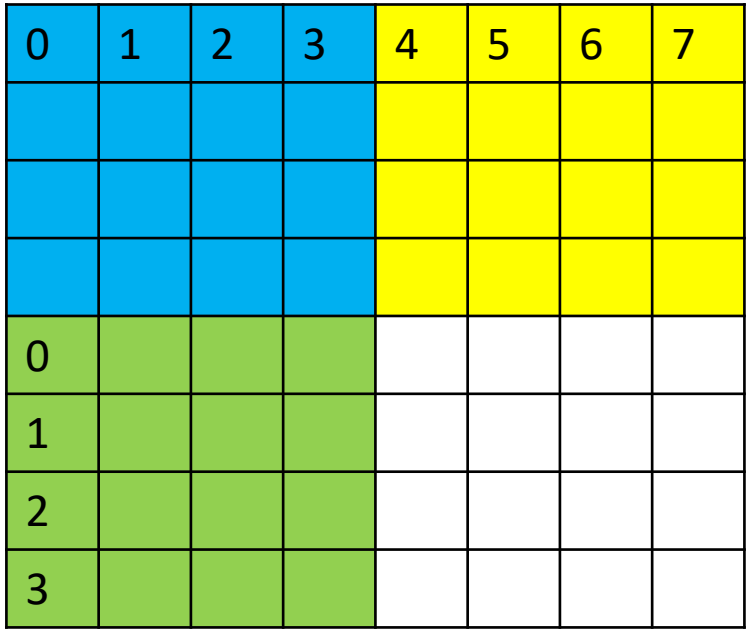
0	1	2	3	4	5	6	7
0							
1							
2							
3							

A[i][j]

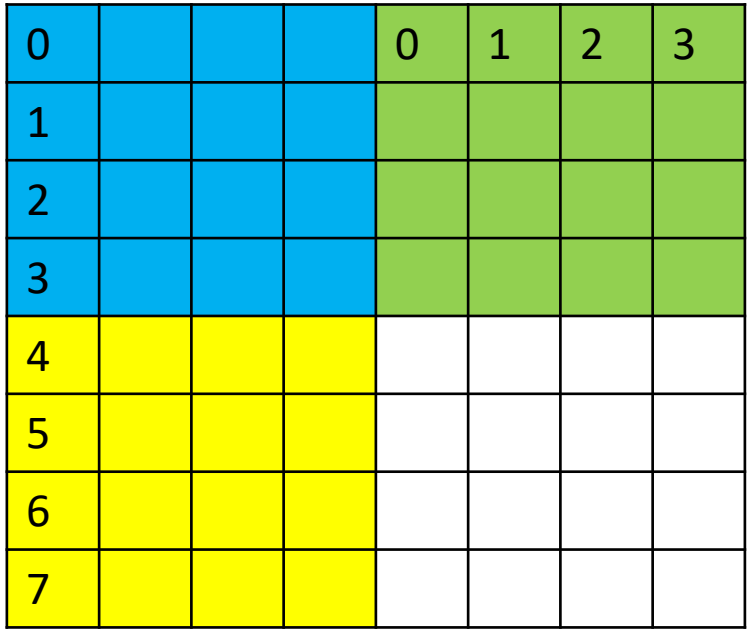
0				0	1	2	3
1							
2							
3							
4							
5							
6							
7							

B[j][i]

Step 3

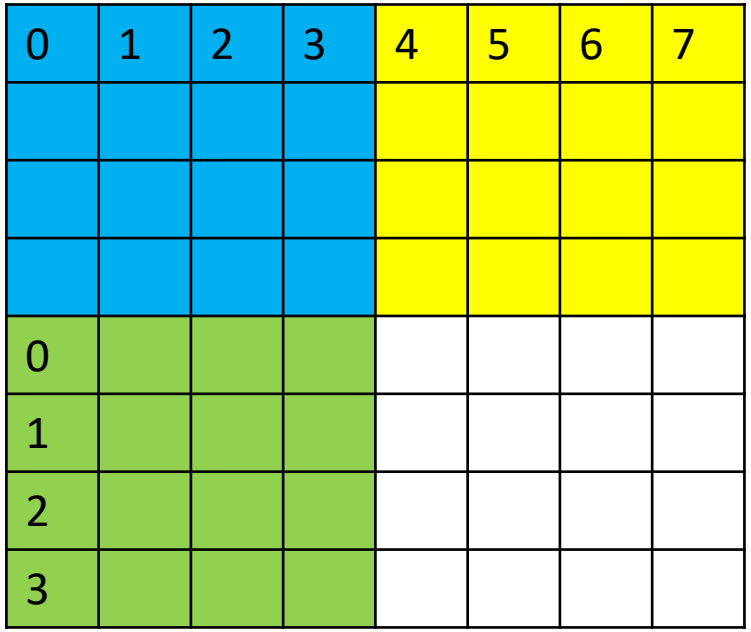


A[i][j]

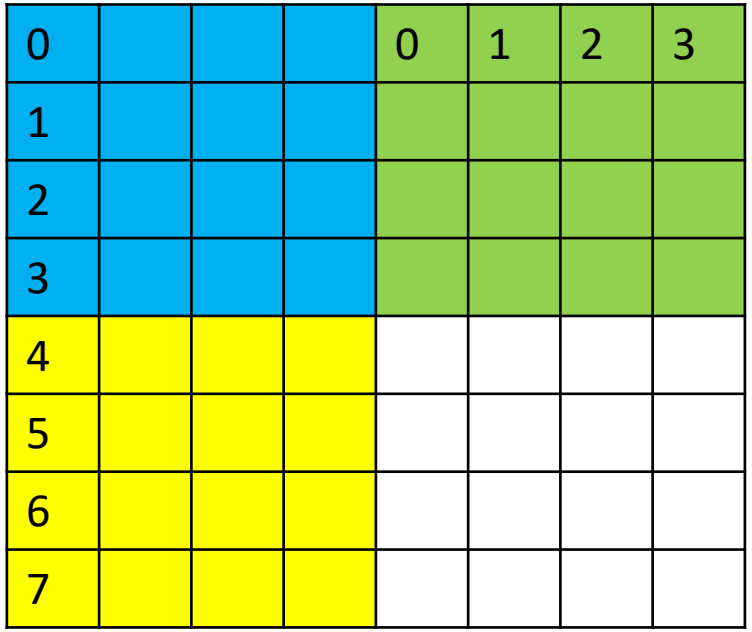


B[j][i]

Step 3

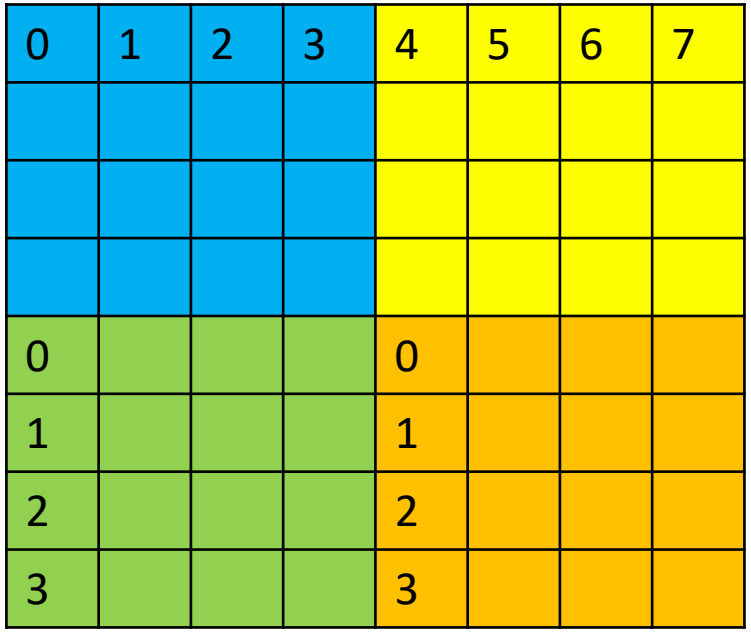


A[i][j]

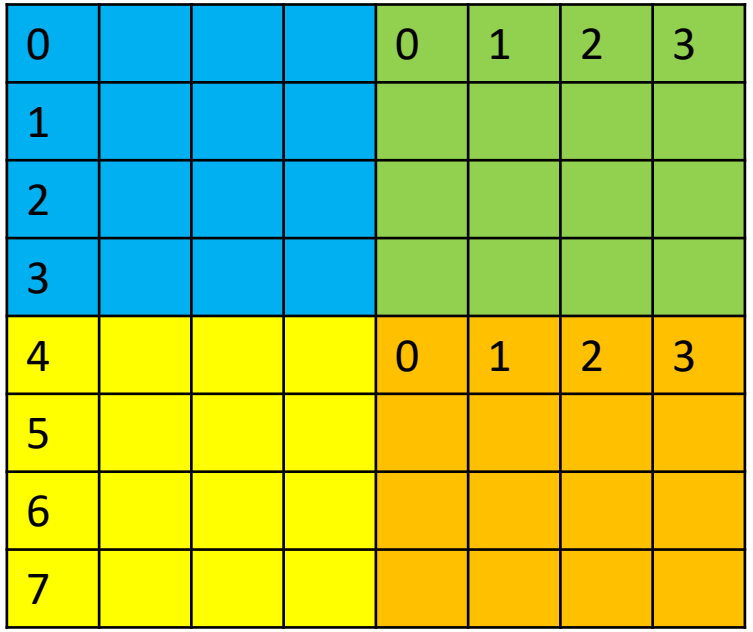


B[j][i]

Step 4



A[i][j]



B[j][i]

Step 5

- 提前对i = j的特殊情况做特别处理
- 使用以B[56][56]为起点的block做跳板
- 或使用B[j][(i+8)%64], B[j][(i+16)%64] 为起点的空间等不冲突的空间做跳板
- 跳板区域要恢复

Thanks!