

H26年度  
並列コンピュータの高速化技法入門  
参考資料

2014年 6月17日  
大阪大学サイバーメディアセンター  
日本電気株式会社

---

本資料は、東北大学サイバーサイエンスセンターとNECの共同により作成され、大阪大学サイバーメディアセンターの環境で実行確認を行い、修正を加えたものです。  
無断転載等は、ご遠慮下さい。

# キャッシュブロッキング

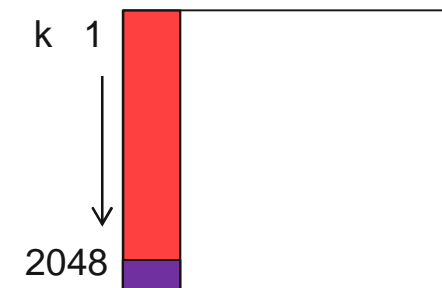
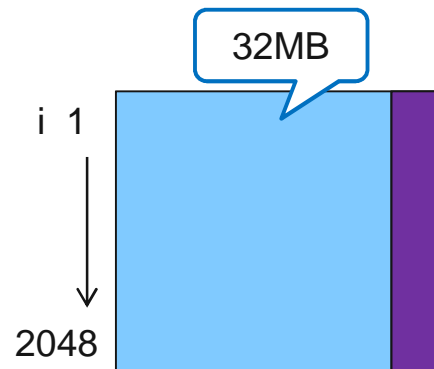
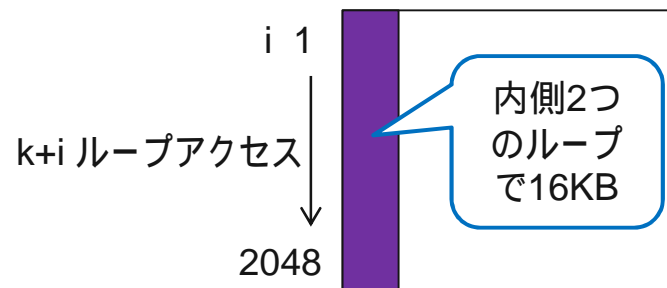
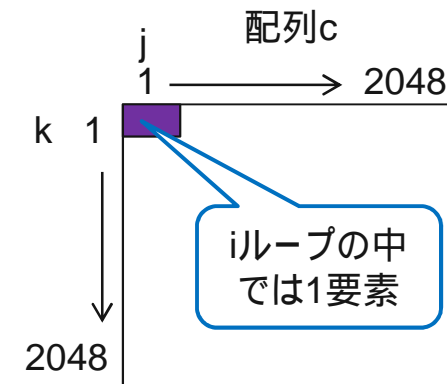
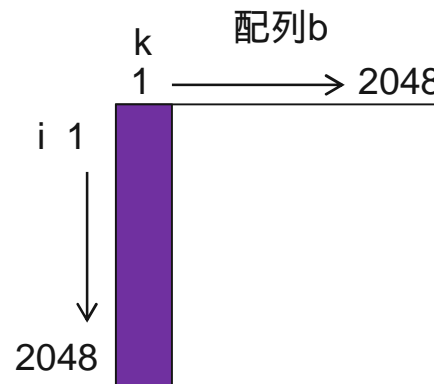
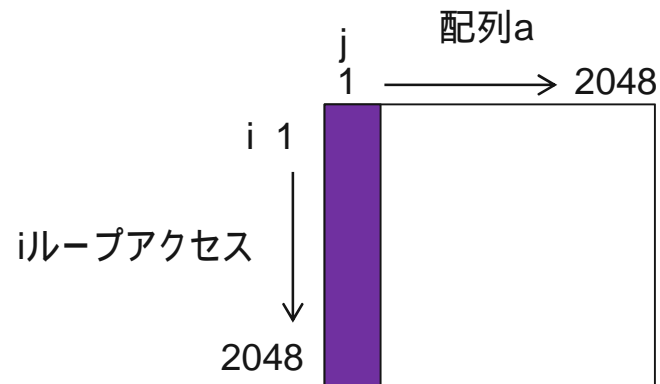
## メモリアクセスの確認

### ●iループとkループ時の配列アクセス

倍精度の配列  
 $2048 \times 8 / 1024 = 16\text{K}$   
 $2048 \times 2048 \times 8 / 1024 / 1024 = 32\text{MB}$

行列積コード

```
do j=1,n  
  do k=1,n  
    do i=1,n  
      a(i,j)=a(i,j)+b(i,k)*c(k,j)
```



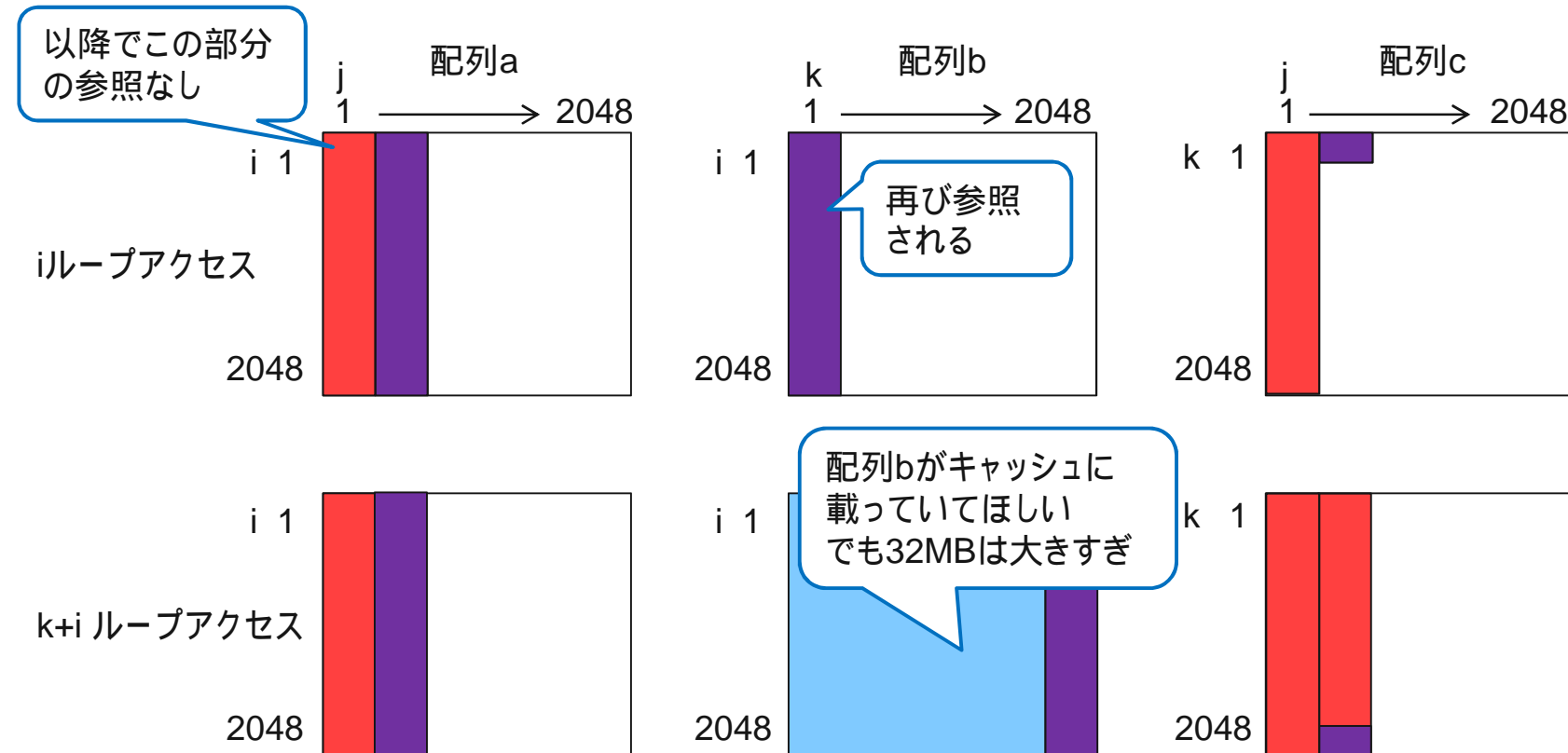
# キャッシュブロッキング

## メモリアクセスの確認 (続き)

- 外側jが1つ進んだ時の配列アクセス

行列積コード

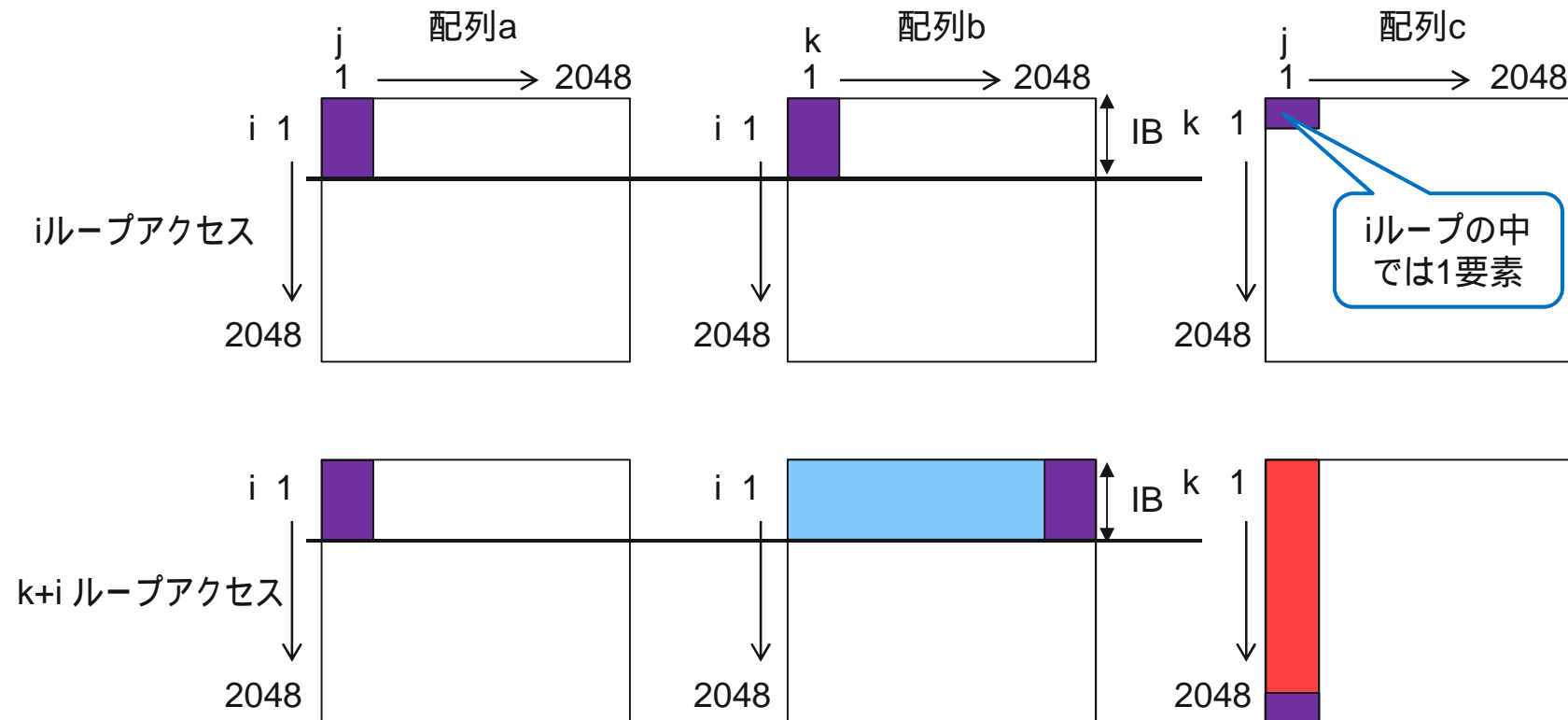
```
do j=1,n  
  do k=1,n  
    do i=1,n  
      a(i,j)=a(i,j)+b(i,k)*c(k,j)
```



# キャッシュブロッキング

- $i$ のブロッキングの場合
- $i$ をブロッキングした時の  $i$ ループと  $k$ ループ配列アクセス
- $IB$ がブロッキングサイズ

```
行列積コード  
do ii=1,n,IB  
do j=1,n  
do k=1,n  
do i=ii,min(n,ii+IB-1)  
a(i,j)=a(i,j)+b(i,k)*c(k,j)
```



# キャッシュブロッキング

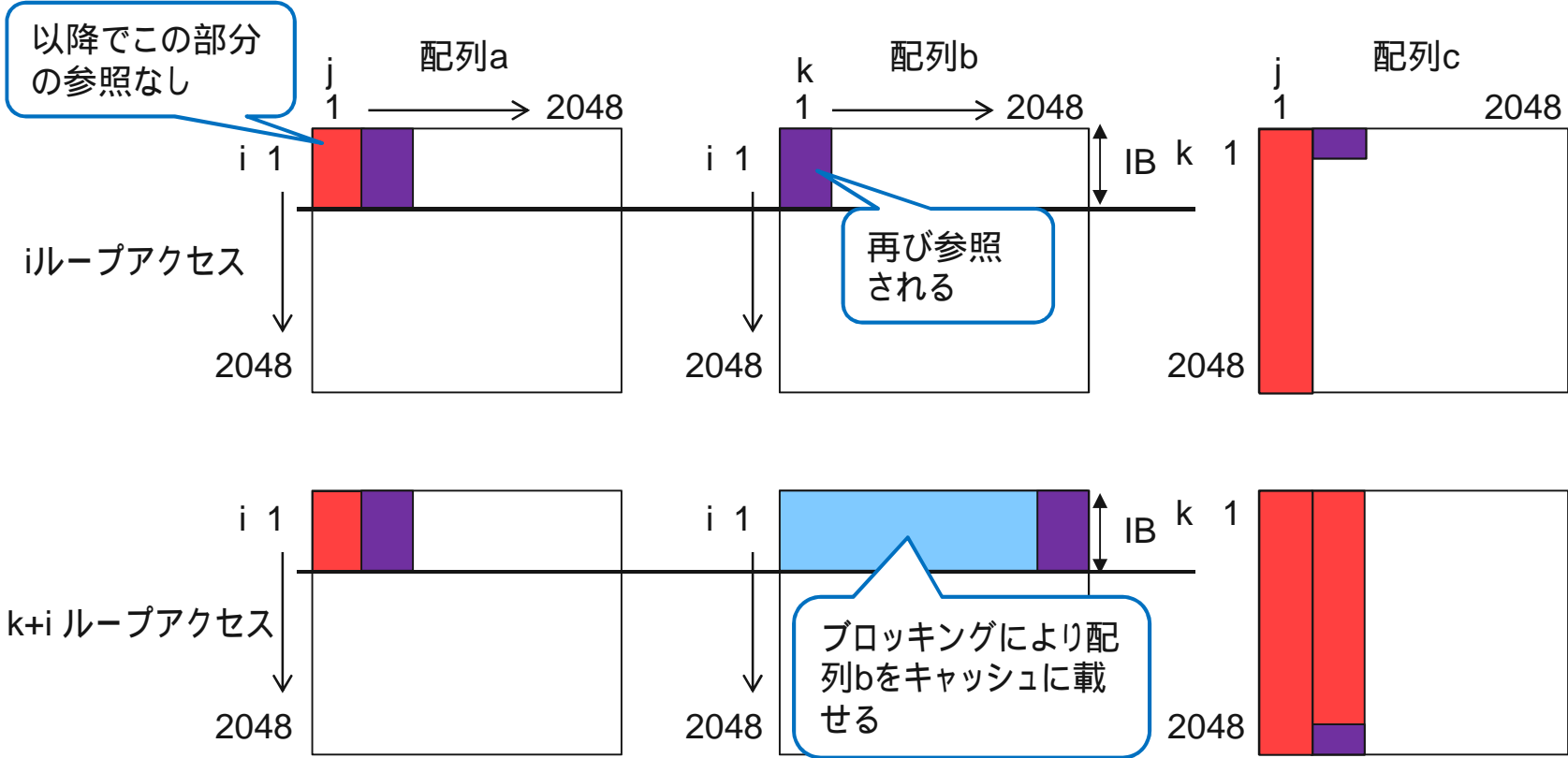
## iのブロッキングの場合(続き)

- iをブロッキングし、外側jが1つ進んだ時の配列アクセス
- 配列bに着目し、 $2048 * IB$ のサイズがキャッシュに収まればjループの繰り返しの間はキャッシュヒット

```

行列積コード
do ii=1,n,IB
do j=1,n
do k=1,n
do i=ii,min(n,ii+IB-1)
a(i,j)=a(i,j)+b(i,k)*c(k,j)

```



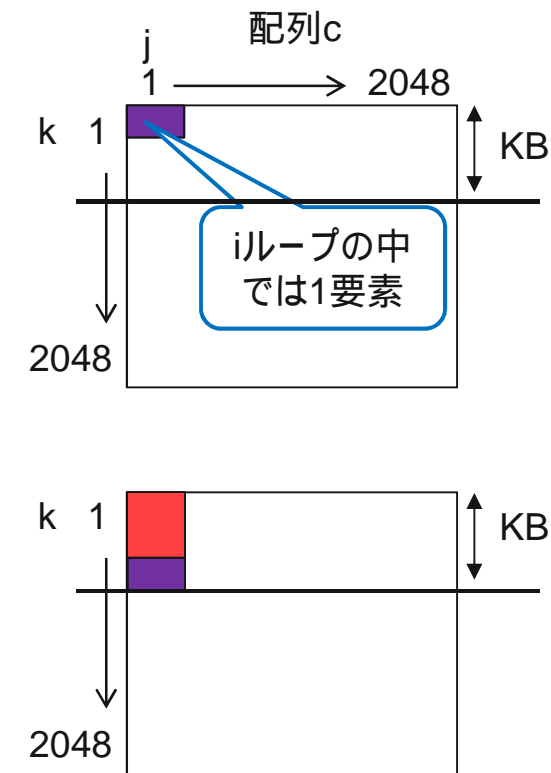
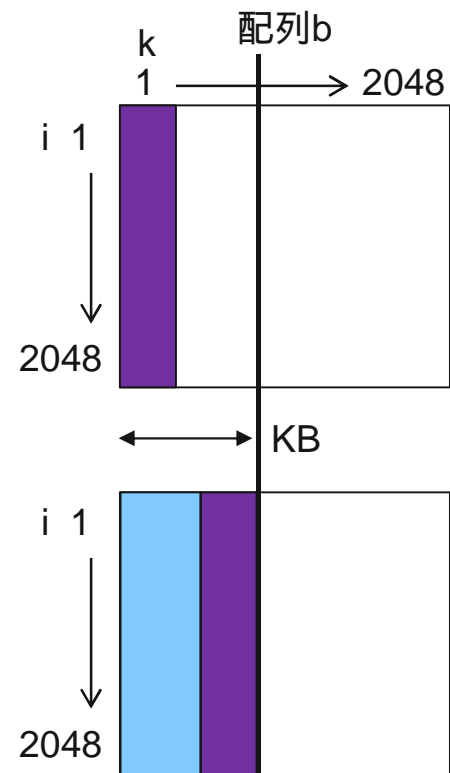
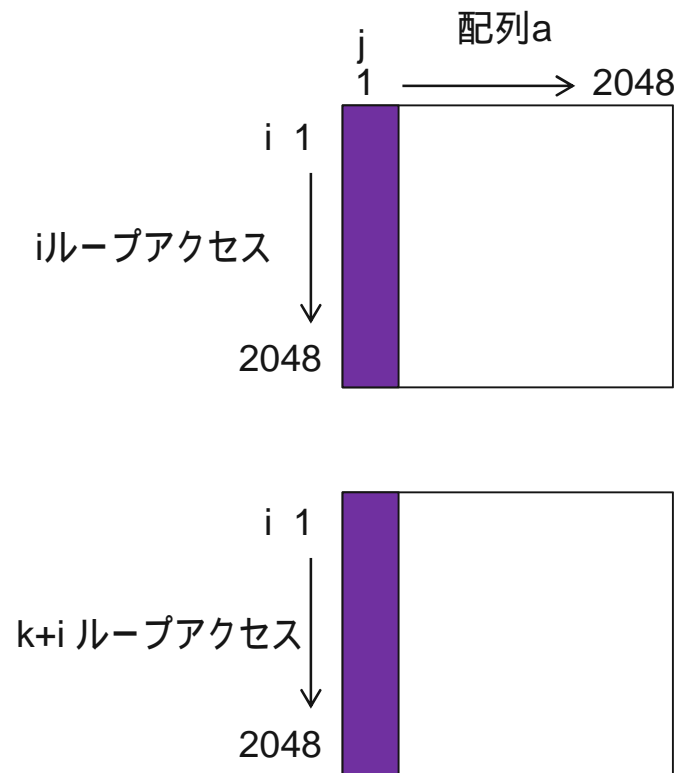
# キャッシュブロッキング

- kのブロッキングの場合
  - kをブロッキングした時のiループとkループ配列アクセス
  - KBがブロッキングサイズ

```

行列積コード
do kk=1,n,KB
do j=1,n
do k=kk,min(n,kk+KB-1)
do i=1,n
a(i,j)=a(i,j)+b(i,k)*c(k,j)

```



# キャッシュブロッキング

## kのブロッキングの場合(続き)

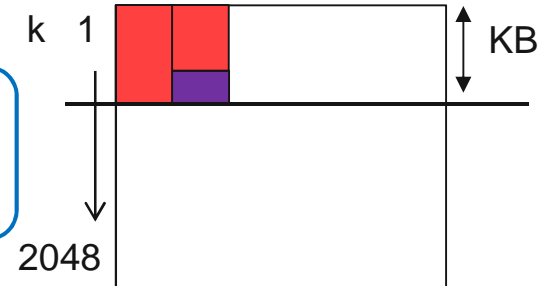
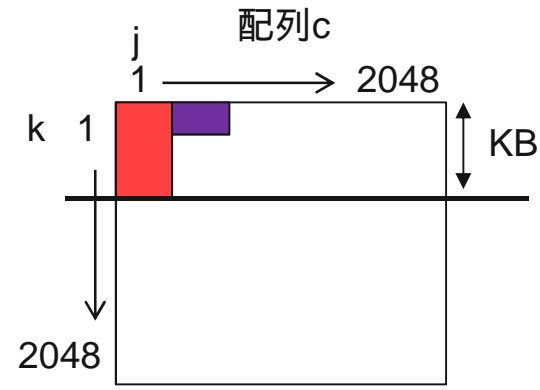
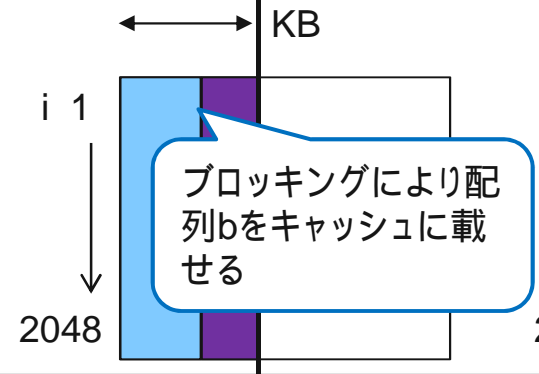
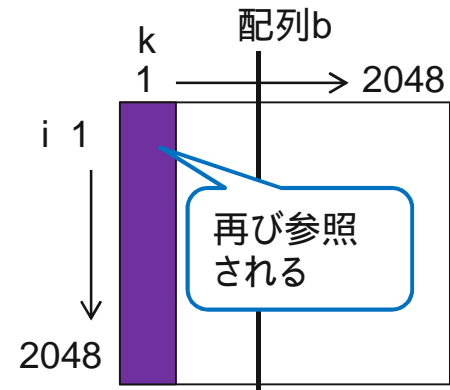
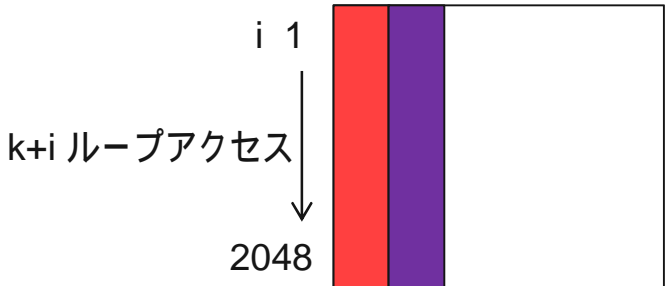
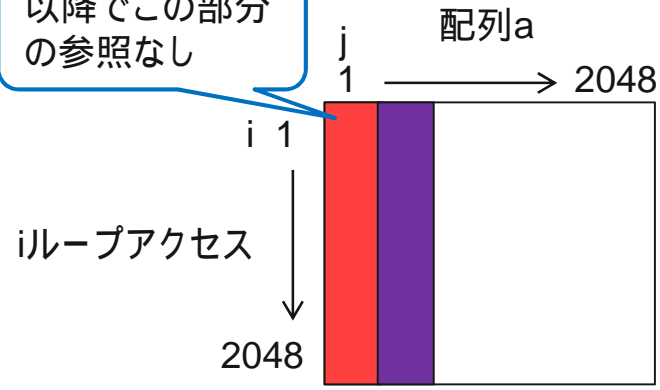
- kをブロッキングし  
外側jが1つ進んだ時の配列アクセス
- 配列bに着目し、 $2048 * KB$ のサイズが  
キャッシュに収まればjループの繰り返しの  
間はキャッシュヒット

```

行列積コード
do kk=1,n,KB
do j=1,n
do k=kk,min(n,kk+KB-1)
do i=1,n
a(i,j)=a(i,j)+b(i,k)*c(k,j)

```

以降でこの部分  
の参照なし





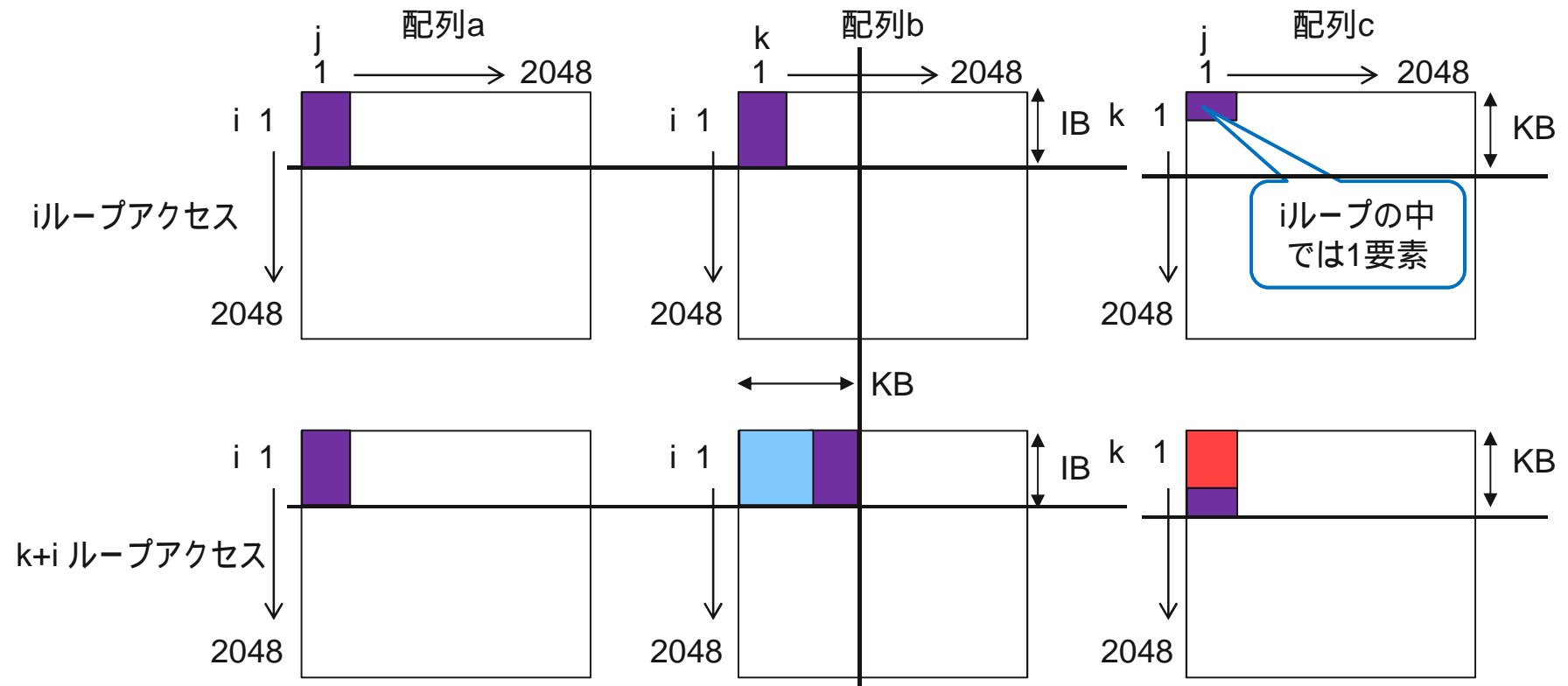
# キャッシュブロッキング

- $i$ と $k$ のブロッキングの場合
  - $i$ と $k$ をブロッキングした時の  $i$ ループと $k$ ループ配列アクセス
  - $IB$ 、 $KB$ がブロッキングサイズ

```

行列積コード
do kk=1,n,KB
do ii=1,n,IB
do j=1,n
do k=kk,min(n,kk+KB-1)
do i=ii,min(n,ii+IB-1)
a(i,j)=a(i,j)+b(i,k)*c(k,j)

```



# キャッシュブロッキング

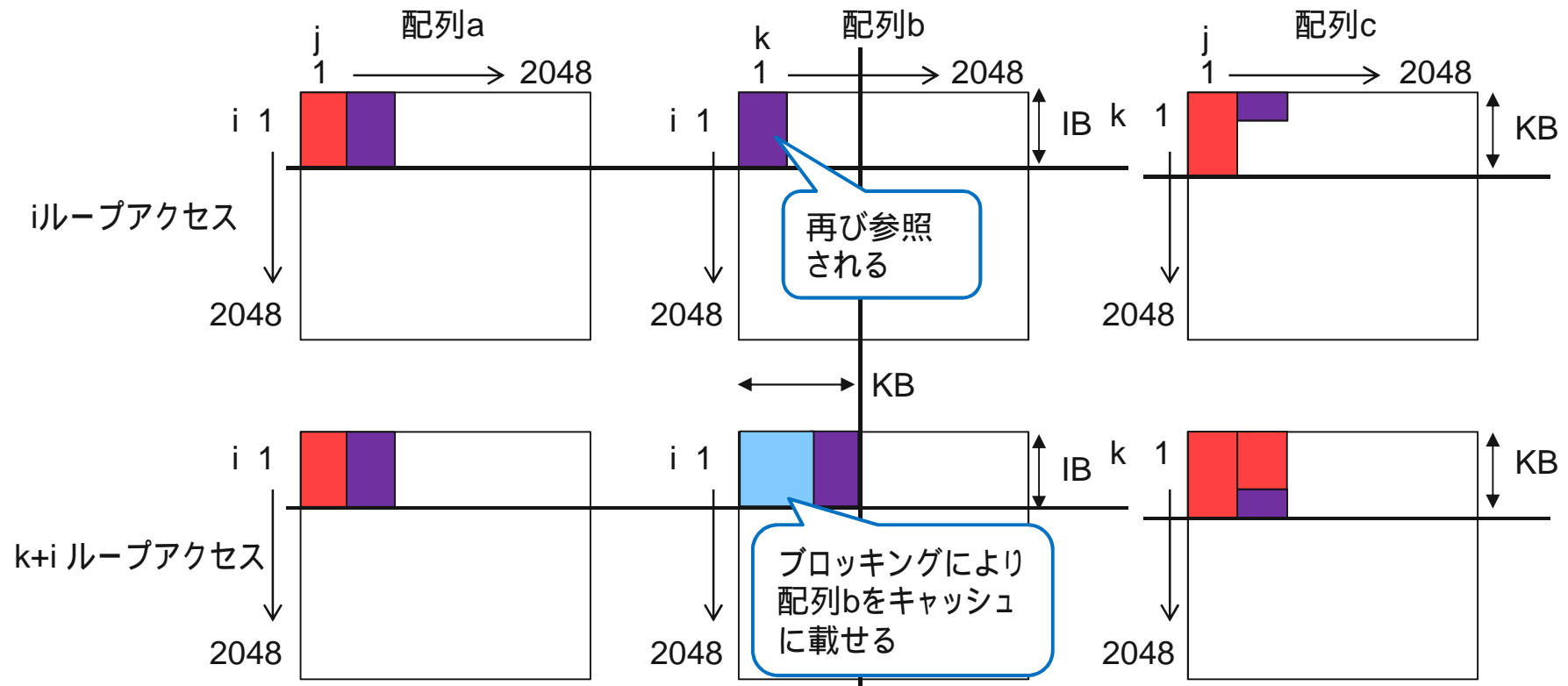
## iとkのブロッキングの場合(続き)

- iとkをブロッキングし  
外側jが1つ進んだ時の配列アクセス
- 配列bに着目し、 $IB * KB$ のサイズが  
キャッシュに収まればjループの繰り返しの  
間はキャッシュヒット

```

行列積コード
do kk=1,n,KB
do ii=1,n,IB
do j=1,n
do k=kk,min(n,kk+KB-1)
do i=ii,min(n,ii+IB-1)
a(i,j)=a(i,j)+b(i,k)*c(k,j)

```



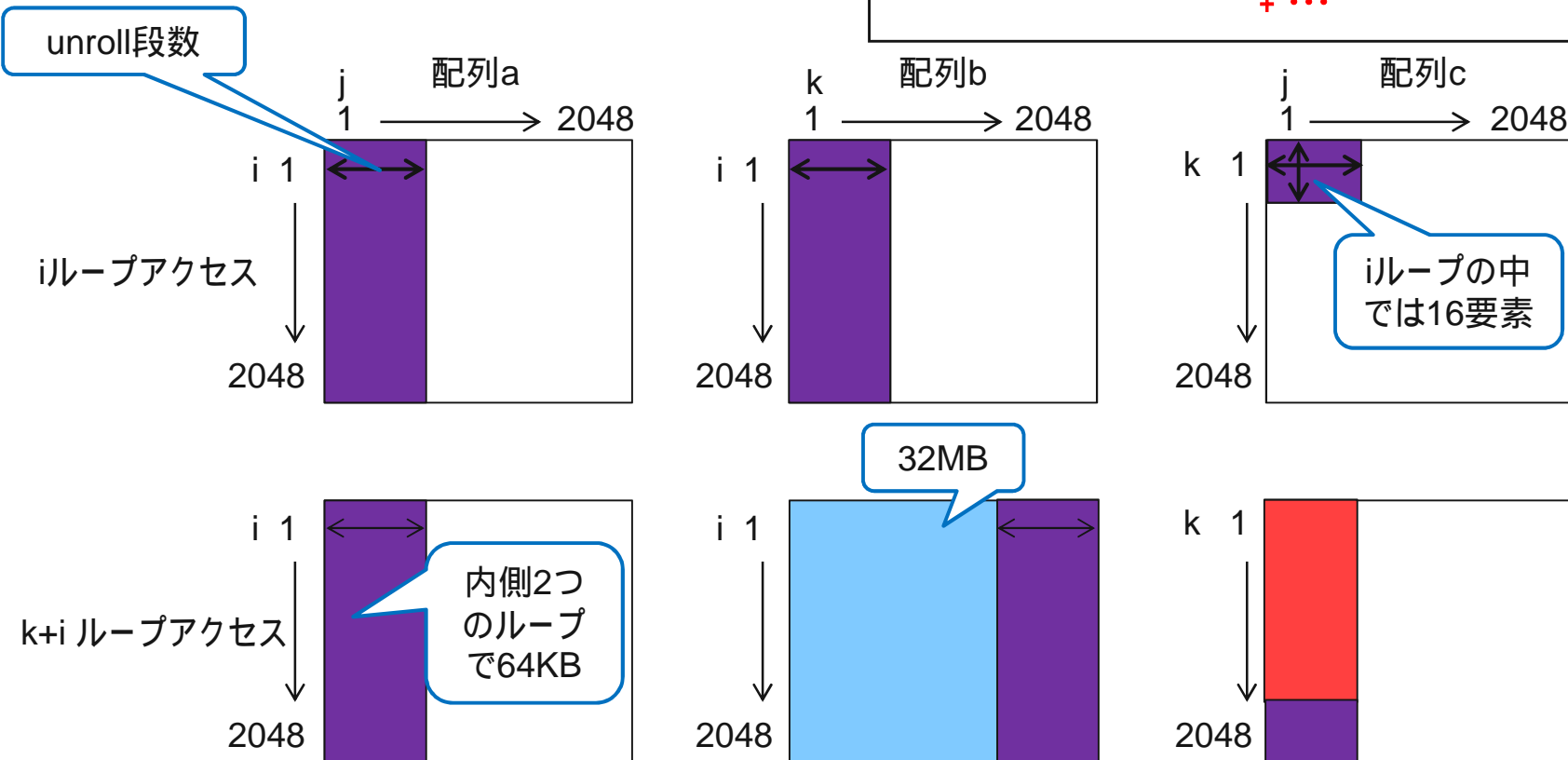
# キャッシュブロッキング

## unrollありでのメモリアクセスの確認

- iループとkループ時の配列アクセス  
外側2ループがそれぞれ4段unroll
- 以降、jループを進めた時のアクセスと  
ブロッキングの考え方は、前記と同じ

行列積コード

```
do j=1,n,4
  do k=1,n,4
    do i=1,n
      a(i,j)=a(i,j)+b(i,k)*c(k,j)
              +b(i,k+1)*c(k+1,j)
              + ...
      a(i,j+1)=a(i,j+1)+b(i,k)*c(k,j+1)
              + ...
    end do
  end do
end do
```

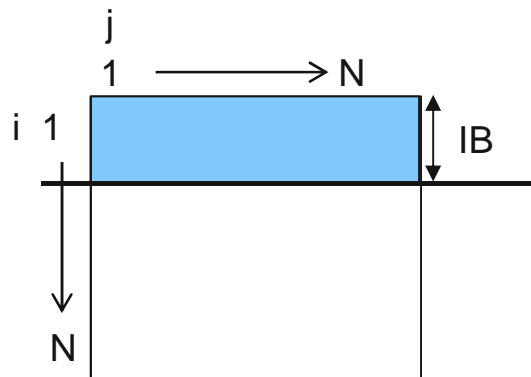


# キャッシュブロッキング

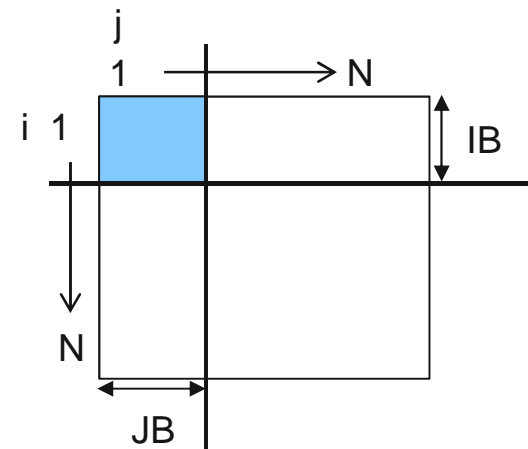
## 注意点

- ブロッキングサイズはL2キャッシュサイズをターゲットとする
- ブロッキングサイズと最内ループ長に注意する

➡ 最内ループ長が短くなりすぎない、かつ、キャッシュに載るサイズを考える必要なら複数のブロッキングを組み合わせる



$N$ が大きい場合、キャッシュに載せるために  $IB$ を小さくすると、最内が  $i$ となるループのループ長が短くなる



$IB$ と $JB$ のブロッキングを組み合わせ、 $IB$ を小さくしすぎないようにする