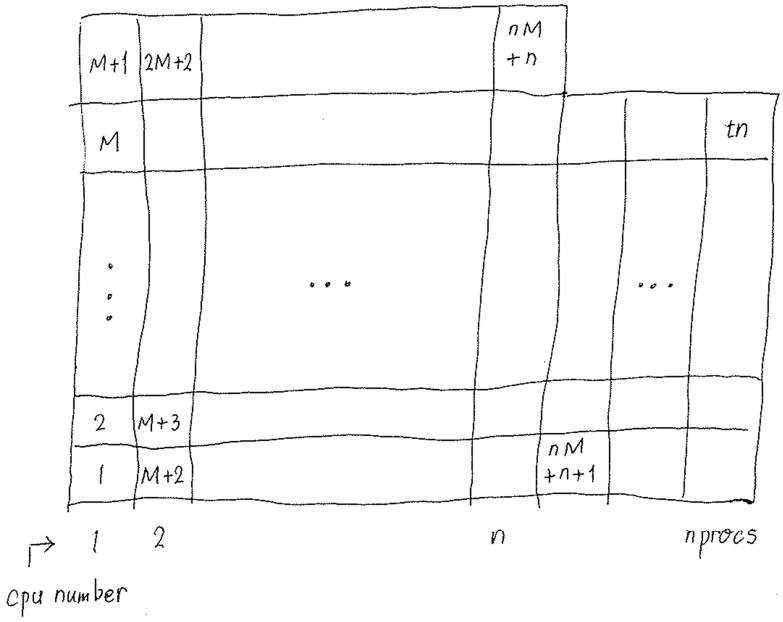
プログラム内で MPI がすでに起動されていると仮定します。まず、カウンター変数を用いて、実行される計算を一次元上に並べるマッピングをつくります。

```
counter = 0

do i1 = 1, Nmax
do i2 = 1, Nmax
do i3 = 1, Nmax
  ia = SomeFunction(i1, i2, i3)
  if( ia .LE. Nmax ) then
     counter = counter + 1
     i1keep(counter) = i1
     i2keep(counter) = i2
     i3keep(counter) = i3
  end if
end do
end do
end do

tn = counter ! total number of calculations
```

次に一次元上に並べた計算を下図のように擬長方形上に配置することにより、cpu へ分配します。



ここで、

```
M = tn/nprocs      ! smaller number of calculations carried by a cpu
n = mod(tn,nprocs)   ! number of cpu carrying M+1 calculations
```

とおいています。nprocs は用いる cpu の総数です。四角の中の数字が一次元上の計算番号、横軸は cpu 番号、縦軸は各 cpu が担当する計算番号を示します。これを用いて、以下のように並列計算します。

```
if ( myrank+1 <= n+1 ) then
 mycal0 = myrank*(M+1)      ! number of calculations before myrank
else
 mycal0 = n*(M+1) + (myrank-n)*M
endif

if ( myrank+1 <= n ) then
 mycaln = M+1             ! number of calculations carried by myrank
else
 mycaln = M
endif

do ione=mycal0+1,mycal0+mycaln
 i1 = i1keep(ione)
 i2 = i2keep(ione)
 i3 = i3keep(ione)

 ! Do the calculation here.

enddo
```

myrank (starting from 0) はあらかじめ各 cpu が得ていると仮定しています。この後必要に応じて、各 cpu の計算結果を他の cpu に適当な通信サブルーチンを用いて伝達します。

以上