Multigrid Computations in High Performance Fortran

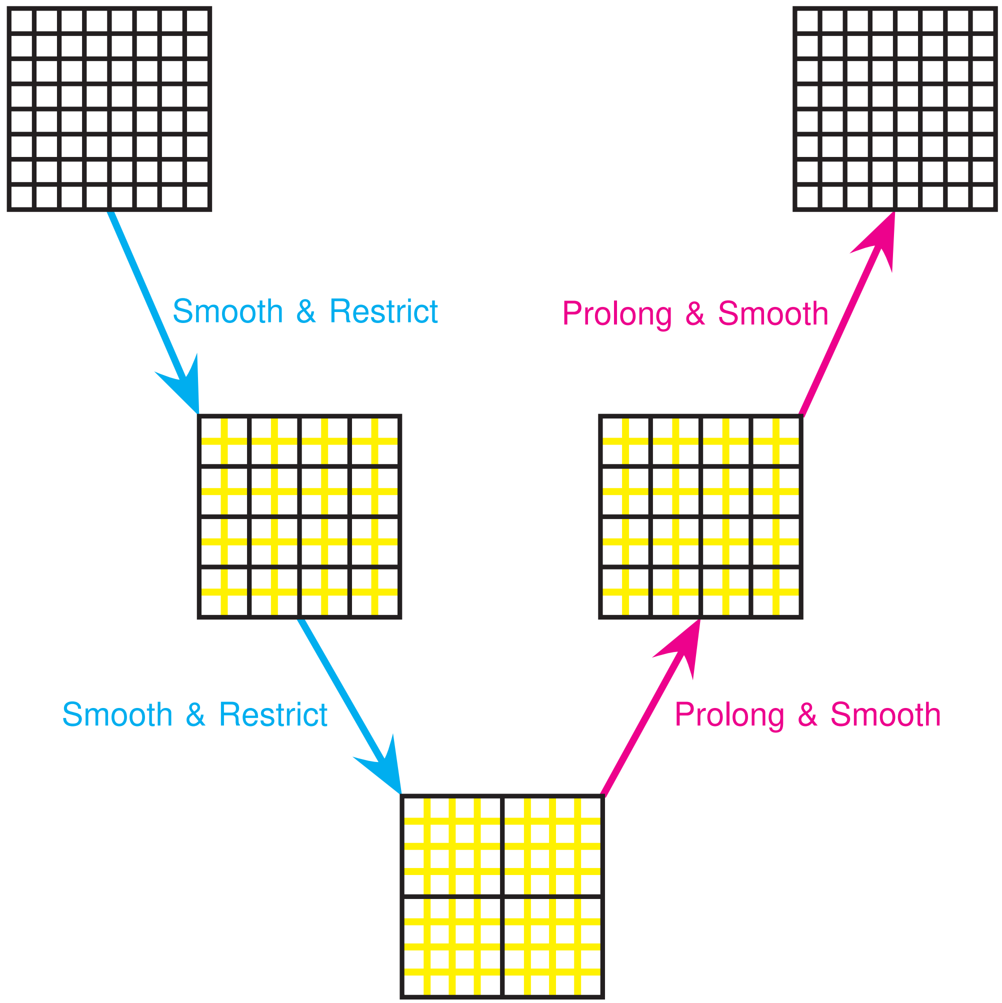
Charles Koelbel
Rice University

CRPC

A Simple Multigrid Application

- **Solving a 2-D partial differential equation (Poisson's equation)**
- **Important features:**
  - Rectangular discretization
  - Periodic boundary conditions
  - Grids distributed by block in both dimensions
  - Multigrid scheme: V-cycle
  - Smoothing: red-black SOR
  - Restriction: 9-point operator
  - Prolongation: linear interpolation
- **Thanks to John van Rosendale (ICASE) for the original code**

Multigrid in Pictures



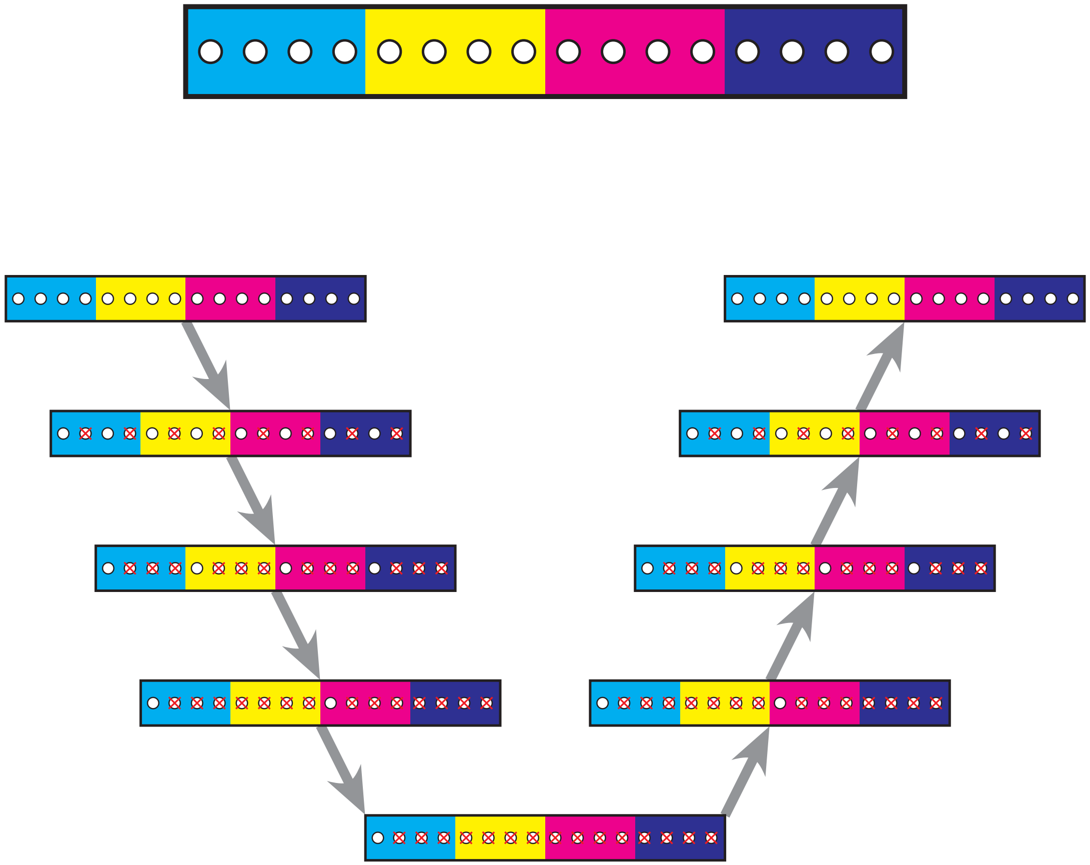
The Bottom Line

- **Multigrid cycle: Same as Fortran 77**
- **Smoothing: Easy**
  - Array intrinsics, FORALL, or INDEPENDENT
- **Prolongation: Not bad**
  - As above, with non-unit strides
- **Restriction: Not bad**
  - Same comments as prolongation
- **Memory management: The hard part**
  - Affects communication
  - Affects load balance
  - Affects compilation

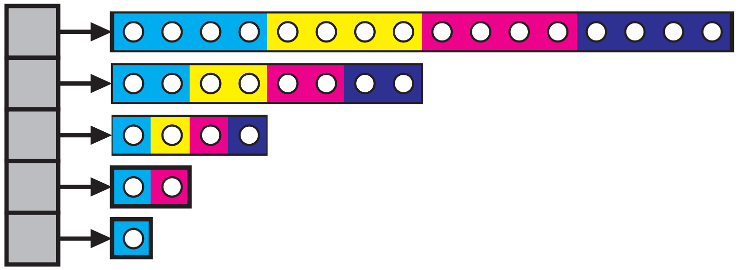
Memory Allocation

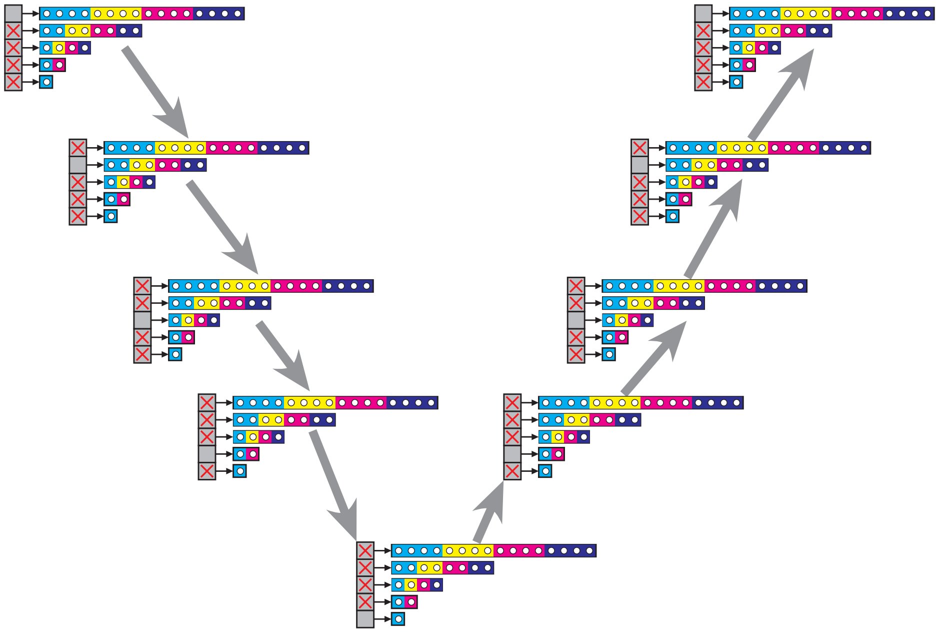
- **ALLOCATABLE or POINTER arrays are needed**
  - BLOCK size depends on array size
  - Allocated once, so minimal overhead
- **Method 1: All levels of grids stored in the same array**
  - "Neighbor" elements in coarse meshes are power-of-2 strides apart
  - Option 1A: Pass whole array to subroutines
  - Option 1B: Pass section that is in use
- **Method 2: List of arrays**
  - As in tutorial presentations
  - Same effect as "allocating" out of large COMMON

Method 1: Overlaid Storage



Method 2: List of Arrays

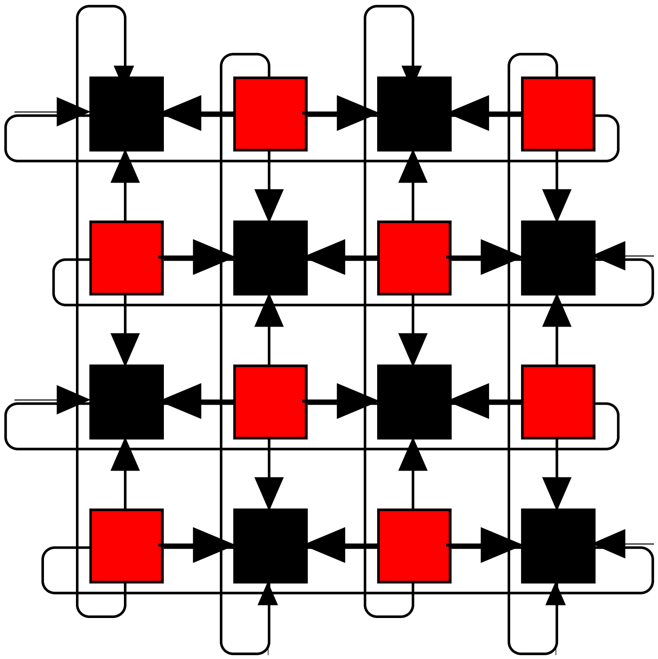




CRPC

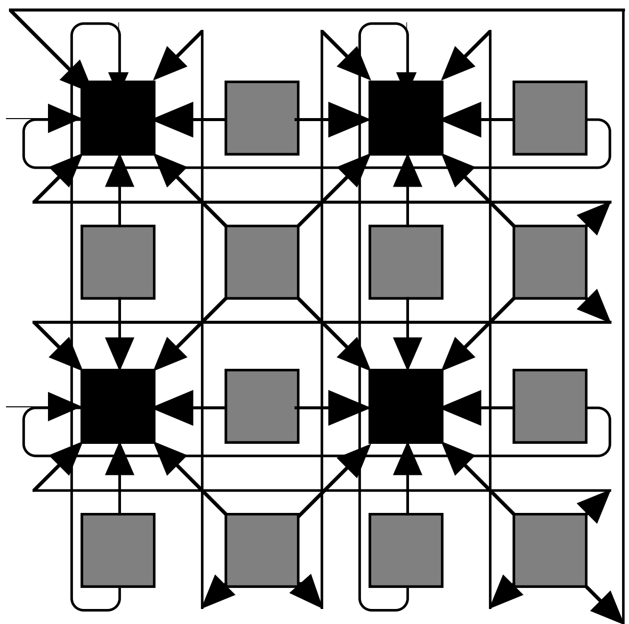
Smoothing

```
!HPF$ INDEPENDENT, &
!HPF$ NEW(jp,jm,ip,im,i0,i)
DO j = 0, ny, jmp
  jp = MOD(nyp + j + jmp, nyp)
  jm =  MOD(nyp + j - jmp, nyp)
  i0 = jmp*MOD(icolor + j/jmp, 2)
  DO i = i0, nx, jmp
    ip = MOD(nxp + i + jmp, nxp)
    im =  MOD(nxp + i - jmp, nxp)
    u(i,j) = 0.25*(u(ip,j) + u(im,j) + &
      u(i,jp) + u(i,jm)) + f(i,j)
  END DO
END DO
```



Restriction

```
r(0:nx:jmp2,0:ny:jmp2) = r(0:nx:jmp2,0:ny:jmp2)+&
  0.5*(r(0:nx:jmp2,jmp:ny:jmp2)+&
    CSHIFT(r(0:nx:jmp2,jmp:ny:jmp2),-1,2)+&
    r(jmp:nx:jmp2,0:ny:jmp2)+&
    CSHIFT(r(jmp:nx:jmp2,0:ny:jmp2),-1,1))+&
  0.25*(r(jmp:nx:jmp2,jmp:ny:jmp2)+&
    CSHIFT(r(jmp:nx:jmp2,jmp:ny:jmp2),(/0,-1/))+&
    CSHIFT(r(jmp:nx:jmp2,jmp:ny:jmp2),(/-1,0/))+&
    CSHIFT(r(jmp:nx:jmp2,jmp:ny:jmp2),(/-1,-1/)))
```



Prolongation

```
FORALL ( i=0:nx:jmp2, j=0:ny:jmp2 )
    u(i,j+jmp) = u(i,j+jmp) + 0.5*u(i,j)
    u(i,j+jmp) = u(i,j+jmp)+0.5*u(i,MOD(j+jmp2+ny,nyp))
    u(i+jmp,j) = u(i+jmp,j) + 0.5*u(i,j)
    u(i+jmp,j) = u(i+jmp,j)+0.5*u(MOD(i+jmp2+nx,nxp),j)
END FORALL
```

(Other points updated by SOR)