HPF Templates for Data Parallel Applications

Geoffrey C Fox, Ken Hawick, Sanjay Ranka

Northeast Parallel Architectures Center

Syracuse University

111 College Place

Syracuse NY 13244-4100

http://www.npac.syr.edu/

NPAC Technical Note SCCS-734

Contents:

- Brief HPF Overview
- HPF Application/Algorithm Categories
- Issues for Solver algorithms in HPF
- Focus on specific HPF code templates for solvers
- HPF Applications Web Package and Related HPF activities on the Web

HPF Overview:

- High Performance Fortran (HPF): adds to Fortran 90 by allowing various directives to specify data decompositions;
- Explicit parallel constructs like FORALL;
- Additional parallel intrinsics functions - optimised on particular architecture.
- Many Details given elsewhere (see WebPackage including tutorial or HPF Forum page)

HPF Application/Algorithm Categories:

- http://www.npac.syr.edu/hpfa/algorithms.html
- Some coarse-grained category list might look like:
  - Sparse matrix, grid and PDE problems
  - Monte Carlo Methods
  - Particle Dynamics
  - Hybrid PDE and Particle Methods
  - Full Matrix formulated algorithms
  - Image Processing, Convolution and Clustering
  - “Embarassingly Parallel”
- difficult to produce ‘orthogonal basis’ of categories!

Sparse matrix, grid and PDE problems Category:

- 1. Regular Grid PDEs and Iterative Solvers
- 2. "Crystalline" Monte Carlo
- 11. Unstructured Grid
- 13. Domain Decomposition Methods
- 16. Structured Multigrid
- 17. Unstructured Multigrid
- 18. Structured and Unstructured Adaptive Mesh
- 22. Direct Sparse Methods
- 24. Network Simulations
- (numbers index into original list on Web pages)

Monte Carlo Methods Category:

- 2. "Crystalline" Monte Carlo
- 3. Quantum and other Independent Monte Carlo Methods
- 21. Monte Carlo Clustering
- (numbers index into original list on Web pages)
- http://www.npac.syr.edu/hpfa/algorithms.html

Particle Dynamics Category:

- 10. O(N**2) N-Body Particle Dynamics
- 12. CHARMM-like Molecular Dynamics
- (numbers index into original list on Web pages)
- http://www.npac.syr.edu/hpfa/algorithms.html

Hybrid PDE and Particle Methods Category:

- 14. Particle in Cell Method
- 15. Discrete Simulation Monte Carlo
- 19. Fast Multipole
- (numbers index into original list on Web pages)
- http://www.npac.syr.edu/hpfa/algorithms.html

Full Matrix formulated algorithms Category:

- 6. Computational Electromagnetics
- 7. Full (Dense) Matrix Algorithms
- 27. Computational Chemistry
- (numbers index into original list on Web pages)
- http://www.npac.syr.edu/hpfa/algorithms.html

Image Processing, Convolution and Clustering Category:

- 8. Spectral and Fast Fourier Transforms
- 9. Low Level Image Processing
- 20. Region Growing and Image Segmentation
- 21. Monte Carlo Clustering
- 23. High Level Image Analysis
- (numbers index into original list on Web pages)
- http://www.npac.syr.edu/hpfa/algorithms.html

"Embarassingly Parallel" Category:

- 4. Unindexed Text Search in Database
- 5. Analysis of Physics data
- 6. Computational Electromagnetics
- 25. Simplex Method
- 26. Graphics Rendering
- (numbers index into original list on Web pages)
- http://www.npac.syr.edu/hpfa/algorithms.html

Issues for Solver algorithms in HPF - focus on:

- Stencil Methods - regular localized data problems
- ADI Algorithm - conflict in optimal data decomposition
- Panel Method and Full Matrix (Direct) Algorithms
- Sparse Matrices in HPF - Conjugate Gradient Algorithm
- Selected relevant NPAC Projects
- Conclusions on HPF applicability for science/engineering problems

Solver Issues for HPF (Part 1):

- HPF's data-parallel model excellent for mesh/stencil methods for initial value problems
- regular mesh boundary value problem still good although some data-parallelism wasted during time to propagate values in from boundary.
- many intrinsics functions do BLAS like operations on dense vectors and arrays.
- line solvers like ADI can work well iff implementation has good communications support for run time library.
- as for all parallel solver algorithms, tradeoff of memory-efficiency and compute-efficiency occurs.

Solver Issues for HPF (Part 2):

- direct (full matrix) methods can be expressed in HPF but are not as efficient as the task parallelism that pure message passing can achieve (eg ScaLAPACK could be a good run time library module)
- unstructured sparse systems can be implemented using INDEPENDENT but are inefficient without new features planned for HPF-2 (private data for processors)
- structured sparse systems can be efficient in storage space (store diagonals as vectors etc, providing iterative method used instead of direct method and can avoid fill-in problems.)

Some Case Studies:

- Stencil Methods;
- Alternate Direction Implicit Problem Formulation;
- Panel Method (dense linear algebra);
- Iterative methods (conjugate gradient).

Stencil Methods (1):

- widely used for PDE solvers (floating point); Cellular Automaton models (bits); Computational Physics Models - Ising (bits), Potts (integers), Heisenberg (floating point), as well as image processing (pixel/voxel bytes)
- key feature is relatively localized data on regular mesh.
- classic example is FTCS Laplace/Poisson solver like:

$$\psi_{i,j} = \psi_{i+1,j} + \psi_{i-1,j} - 4\psi_{i,j} + \psi_{i,j+1} + \psi_{i,j-1} \qquad (1)$$

- HPF very well suited to such problems through mechanisms for the communication of neighbouring data values on regular meshes - CSHIFT/FORALL.

Stencil Methods (2)- CSHIFT/FORALL:

- Laplace/Poisson example coded as:

```
 psi = cshift(psi,1,1) + cshift(psi,-1,1) +
&      cshift(psi,1,2) + cshift(psi,-1,2) + 4.0 * psi
```

- or (in case of non-periodic boundaries):

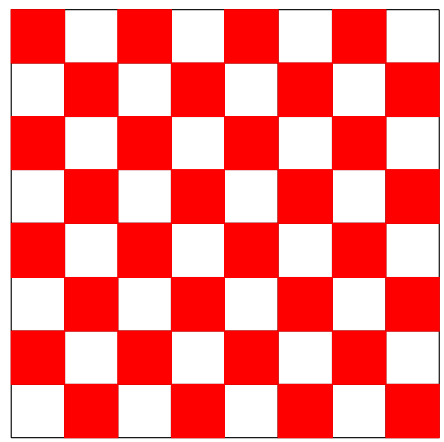
```
 forall(i=2:m-1,j=2:n-1)
   psi(i,j) = psi(i+1,j) + psi(i-1,j) - 4.0 * psi(i,j) +
&             psi(i,j+1) + psi(i,j-1)
 end forall
```

- HPF Block Distribution probably optimal.

```
!HPF DISTRIBUTE PSI(BLOCK,BLOCK)
```

Stencil Methods (3) - Red-Black variations:

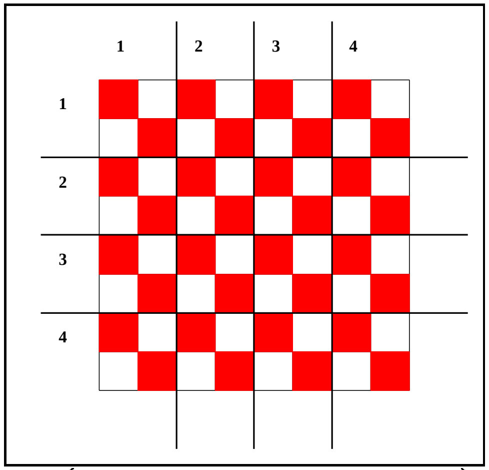
- Algorithmic variations - such as red-black ordering still efficient with correct distributed storage - BLOCK again - originally known as ‘CRINKLE mapping’ on DAP in early 1980’s.
- ‘Red’-‘Black’ mesh colouring. (actually ‘Black-White’ on this slide.



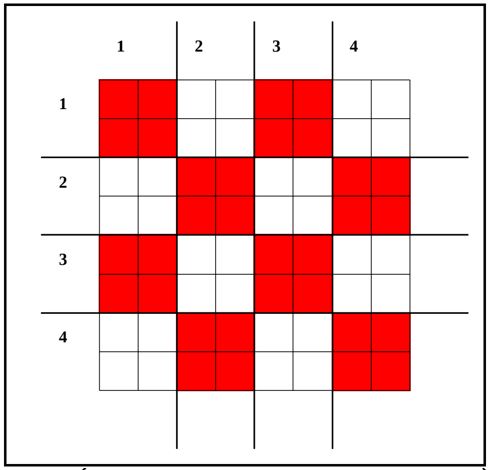
Stencil Methods (4) - DISTRIBUTE

Mappings:

- Red-Black mesh i. distributed (BLOCK,BLOCK) and ii. distributed (CYCLIC,CYCLIC) on 4x4 processor grid.



i. (BLOCK,BLOCK)



ii. (CYCLIC,CYCLIC)

Stencil Methods (5):

- Stencil PDE solvers not totally trivial for 'interesting' applications.
- tradeoff exists between memory use and communications
- consider FTCS solution method for Cahn-Hilliard field equation (used for binary alloy models):

$$\frac{\partial \phi}{\partial t} = m\nabla^2 \left(-b\phi + u\phi^3 - K\nabla^2\phi\right) \quad (2)$$

- field variable $\phi_{i,j}$ is on a regular mesh $\nabla^2(\nabla^2)$,
- update procedure requires next and next-next nearest neighbouring mesh point data.

Stencil Methods (6):

- formulate update as memory efficient but communications-inefficient using 20 CSHIFT operations:

```
 phi = phi + dtby2 * (  -16.0 * phi
&  + 7.0 * ( cshift(phi, 1,1) + cshift(phi, 1,2)
&          + cshift(phi,-1,1) + cshift(phi,-1,2) )
&        - ( cshift(phi, 2,1) + cshift(phi, 2,2)
&          + cshift(phi,-2,1) + cshift(phi,-2,2) )
&   -2.0 * ( cshift(cshift(phi, 1,1), 1,2)
&          + cshift(cshift(phi, 1,1),-1,2)
&          + cshift(cshift(phi,-1,1), 1,2)
&          + cshift(cshift(phi,-1,1),-1,2) )
&   -4.0 * phi ** 3
&        + ( cshift(phi, 1,1)**3 + cshift(phi, 1,2)**3
&          + cshift(phi,-1,1)**3 + cshift(phi,-1,2)**3 )   )
```

Stencil Methods (6):

- or formulate using twice memory requirements but using only 8 CSHIFT operations:

```
 f = 3.0 * phi - ( cshift(phi,1,1) + cshift(phi,-1,1) +
&                   cshift(phi,1,2) + cshift(phi,-1,2) ) + phi ** 3
 phi = phi + dtby2 * ( cshift(f,1,1) + cshift(f,-1,1) +
&                       cshift(f,1,2) + cshift(f,-1,2) -4.0 * f)
```

- f is temporary array and is same size as phi - inefficient use of memory if problem mesh sizes are very much larger than processor grid.
- perhaps future HPF will allow processors to have private local working data (temporary working array is then only same size as processor grid)

Alternate Direction Implicit Problem Formulation:

- Consider Equation:

$$\nabla^2\psi = f \quad on \quad \Omega = (0,1)\times(0,1), \tag{3}$$
$$\psi = \psi_g \quad on \quad \partial\Omega, \tag{4}$$

- Dirichlet boundary conditions on ψ.
- $N_x + 1$ intervals in x and $N_y + 1$. intervals in y

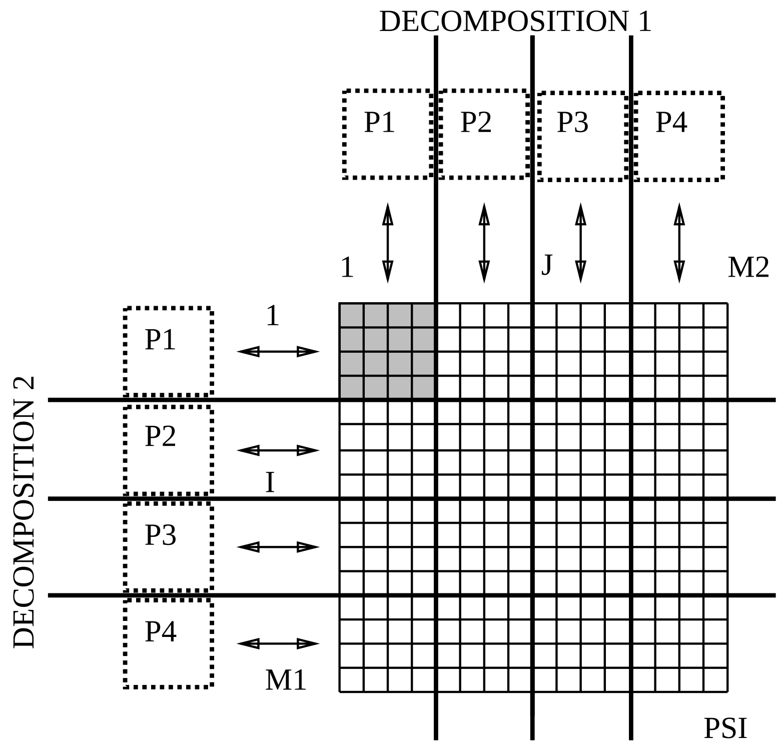
```
   PROGRAM ADI_SEQUENTIAL
...  declarations, interfaces and initializations ...
   DO ITER = 1,ITERMAX    ! begin iteration loop
     DO I = 1,NX          ! x sweep
       C(1,1:NY) = DY2INV
       C(2,1:NY) = -2.0_FPNUM*(DX2INV+DY2INV)
       C(3,1:NY) = DY2INV
       C(4,1:NY) = F(I,1:NY)-DX2INV*(PSI(I+1,1:NY)+ PSI(I-1,1:NY))
       CALL THOMAS(NY,C,BCBOT,BCTOP,PSI(I,0:NY+1))
     END DO
     DO J = 1,NY !y sweep
       C(1,1:NX) = DX2INV
       C(2,1:NX) = -2.0_FPNUM*(DX2INV+DY2INV)
       C(3,1:NX) = DX2INV
       C(4,1:NX) = F(1:NX,J)-DY2INV*(PSI(1:NX,J+1)+ PSI(1:NX,J-1))
       CALL THOMAS(NX,C,BCLFT,BCRHT,PSI(0:NX+1,J))
     END DO
...  check convergence ...
     END DO
   END
```

Thomas algorithm for ADI code:

```
      SUBROUTINE THOMAS (NK,C,ZO,ZN,Z)
...   declarations ...
      D(1,0) = 0.0
      D(2,0) = ZO
      DO K=1,NK
        D(1,K) = -C(1,K)/(C(2,K) + C(3,K)*D(1,K-1))
        D(2,K) = (C(4,K)-C(3,K)*D(2,K-1))/ &
                 (C(2,K) + C(3,K)*D(1,K-1))
      END DO
      Z(NK+1) = ZN
      DO K=NK,1,-1
        Z(K) = D(1,K)*Z(K+1) + D(2,K)
      END DO
      Z(0) = ZO
      RETURN
      END
```

Modified x- and y-sweeps for data parallel execution:

```
      DO I = 1,NX     !x sweep
      C(1,I,1:NY) = DY2INV
      C(2,I,1:NY) = -2.0_FPNUM*(DX2INV+DY2INV)
      C(3,I,1:NY) = DY2INV
      C(4,I,1:NY) = F(I,1:NY) - &
                    DX2INV*(PSI(I+1,1:NY)+PSI(I-1,1:NY))
      END DO
...   solve a set of tridiagonal system of equations
      DO J = 1,NY       !y sweep
      C(1,J,1:NX) = DX2INV
      C(2,J,1:NX) = -2.0_FPNUM*(DX2INV+DY2INV)
      C(3,J,1:NX) = DX2INV
      C(4,J,1:NX) = F(1:NX,J) - &
                    DY2INV*(PSI(1:NX,J+1)+PSI(1:NX,J-1))
      END DO
...   solve a set of tridiagonal system of equations
```



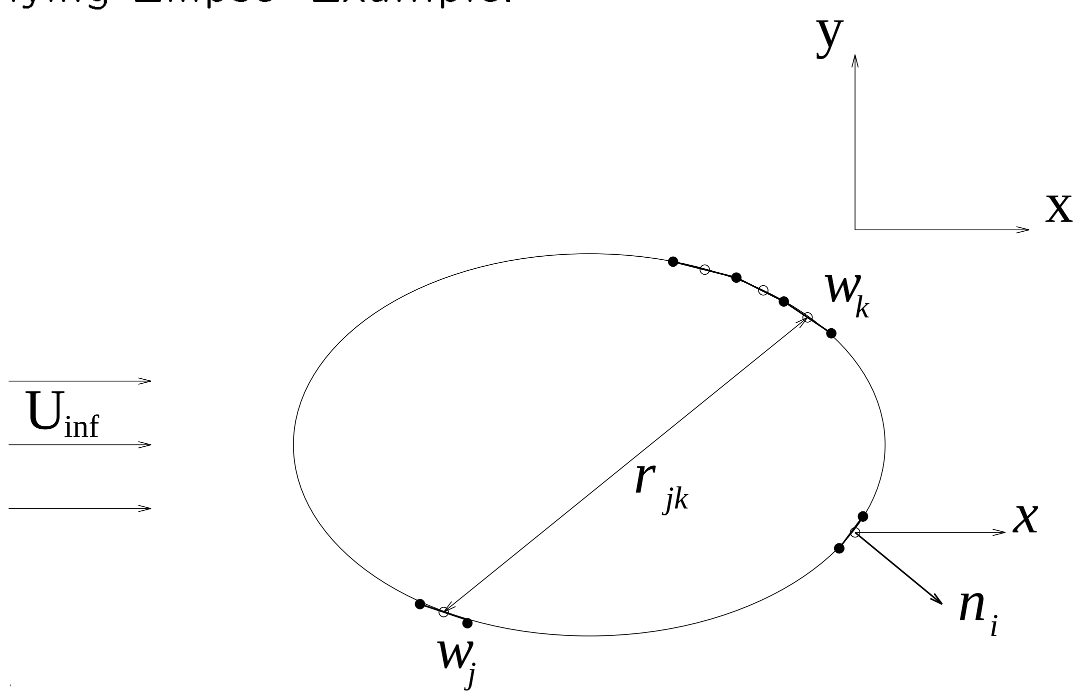
Optimal Data Decompositions for Matrix equations in ADI Solver (2D problem)

ADI Conclusions:

- ADI can be neatly expressed in HPF.
- many of BLAS type operations are in-line equivalents of Fortran 90 intrinsics. (eg SAXPY is one line of code, MATMUL intrinsic replaces all dense array multiplications.)
- REDISTRIBUTE is used instead of TRANSPOSE for 3d and higher dimensionality problems
- Performance depends on how well intrinsics and run time library is implemented.

Full Matrix Methods: Panel Method - Ellipse Example

- 'Flying Ellipse' Example:



- Ellipse in uniform incident flow U_{inf}, showing k'th panel of source strength $w_k \int ds_k$ at r_k.

Panel Method Formulation: Body in a uniform velocity U_{inf}, distribution of N source panels produces potential:

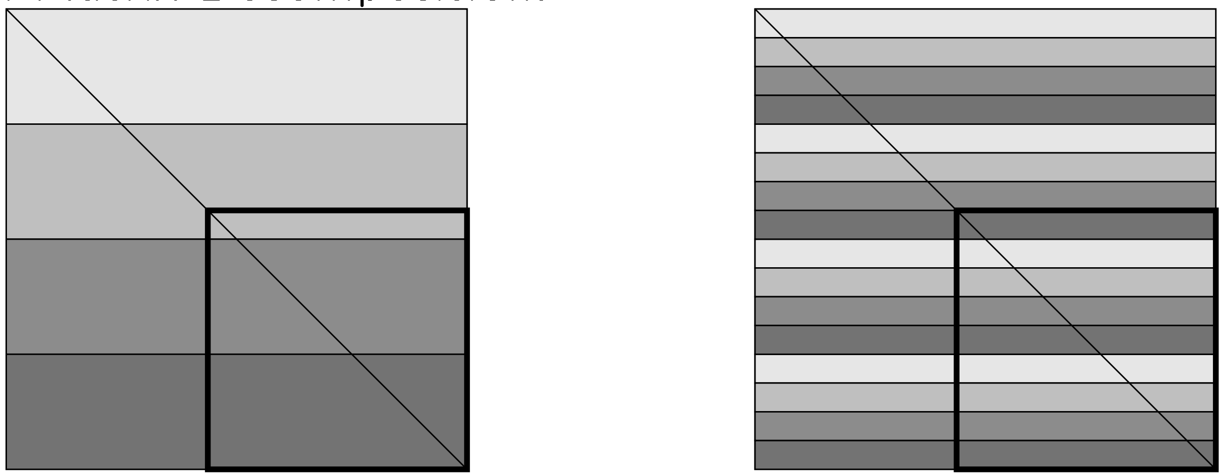
$$\Phi(\vec{r}_k) = U_0 x_k + \frac{1}{2\pi} \sum_{j=1}^{N} w_j \int \ln |\vec{r}|_{k,j} ds_j, \tag{5}$$

$\vec{r}_k = (x_k, y_k)$ is position of each panel's control point, $|\vec{r}|_{k,j}$ is distance between two panels, and $w_k \int ds_k$ is the source strength of the k'th panel. Source densities determined from boundary condition of zero normal flow through body surface: $v_n = \frac{\partial \Phi}{\partial n_k} = 0$ generates system of linear equations $A \cdot \vec{w} = \vec{b}$ with each component of A:

$$A_{k,j} = \frac{\delta_{k,j}}{2} + \frac{1}{2\pi} \int \frac{\partial}{\partial n_k} (\ln r_{k,j}) ds_j, \tag{6}$$

and RHS vector is $b_k = U_0 \sin \alpha_k$, where α_k is angle between panel and x-axis. Once vector of source densities determined, velocity field obtained from potential.

Full Matrix Decomposition:

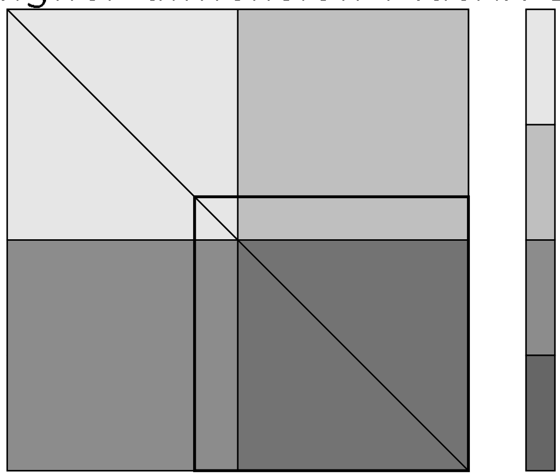


BLOCK CYCLIC

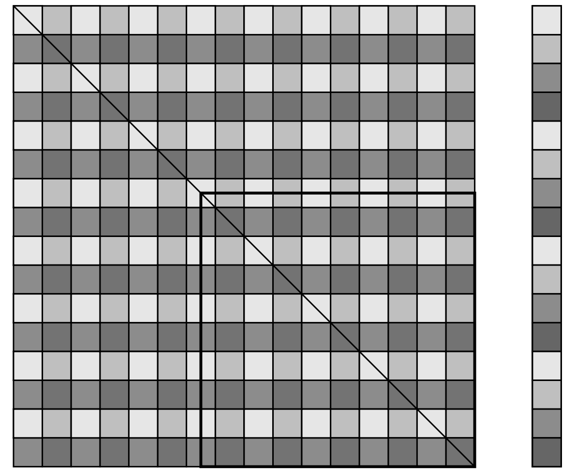
Row distribution across (4) processors' memories. (Column is similar).

Row distribution requires a distributed global test for the pivot, whereas for column this is poorly balanced - but requires no communication. Row requires broadcast of partial matrix row, column requires broadcast of multiplication factor.

Higher dimension Matrix Distribution



BLOCK



CYCLIC

Cyclic row distribution provides best decomposition for both matrix and RHS vector operations.

Column distributions are poor since a single processor is required to work on entire RHS at all stages of back -substitution. Mixing distributions is also possible but REDISTRIBUTE may be costly.

Full Matrix Conclusions:

- HPF allows problem to be expressed quite neatly
- as important to remove serial code as to add parallel code sections
- Once code is in HPF, it allows different decompositions to be tuned with only minor code changes - eg (BLOCK,*) easily changed to (*,CYCLIC)
- which of the decompositions performs best depends on problems sizes and characteristics of particular computer system (how well TRANSPOSE and REDISTRIBUTE are implemented)
- present implementations cannot outperform message passing for this sort of application (for example ScaLAPACK library), although in future ScaLAPACK may be part of run time library invoked by the compiler on programmers behalf.

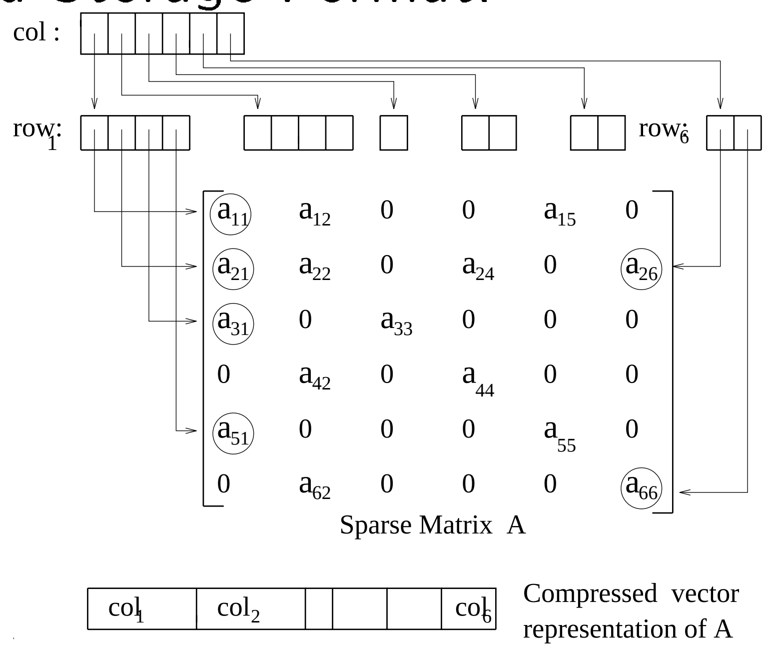
Sparse Methods - Conjugate Gradient example. The non-preconditioned CG algorithm is summarised as:

$\vec{p} = \vec{r} = \vec{b};\ \vec{x} = 0;\ \vec{q} = A\vec{p}$
$\rho = \vec{r} \cdot \vec{r};\ \alpha = \rho/(\vec{p} \cdot \vec{q}$
$\vec{x} = \vec{x} + \alpha\vec{p};\ \vec{r} = \vec{r} - \alpha\vec{q}$
DO $k = 2$, Niter
    $\rho_0 = \rho;\ \rho = \vec{r} \cdot \vec{r};\ \beta = \rho/\rho_0$
    $\vec{p} = \vec{r} + \beta\vec{p};\ \vec{q} = A \cdot \vec{p}$
    $\alpha = \rho/\vec{p} \cdot \vec{q}$
    $\vec{x} = \vec{x} + \alpha\vec{p};\ \vec{r} = \vec{r} - \alpha\vec{q}$
    IF(stop_criterion)exit
ENDDO

for the initial “guessed” solution vector $\vec{x}^0 = 0$.

Implementation of this algorithm requires storage for four vectors:, $\vec{x}$, $\vec{r}$, $\vec{p}$ and $\vec{q}$ as well as the matrix A and working scalars α and β.

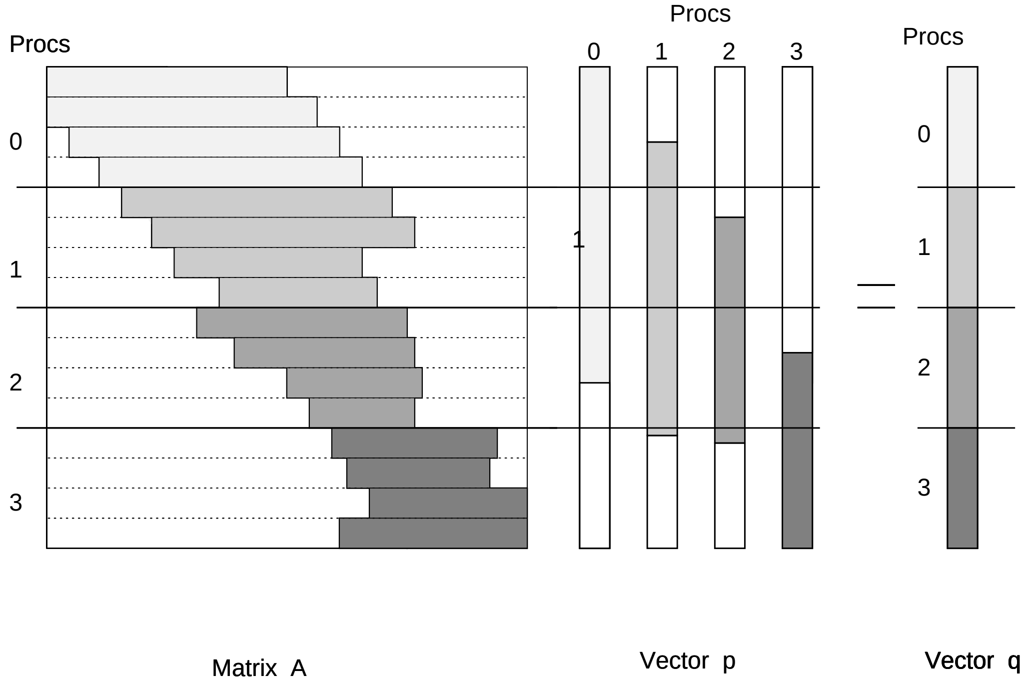
Compressed Storage Format:



Compressed Sparse Column(CSC) representation of sparse matrix A.

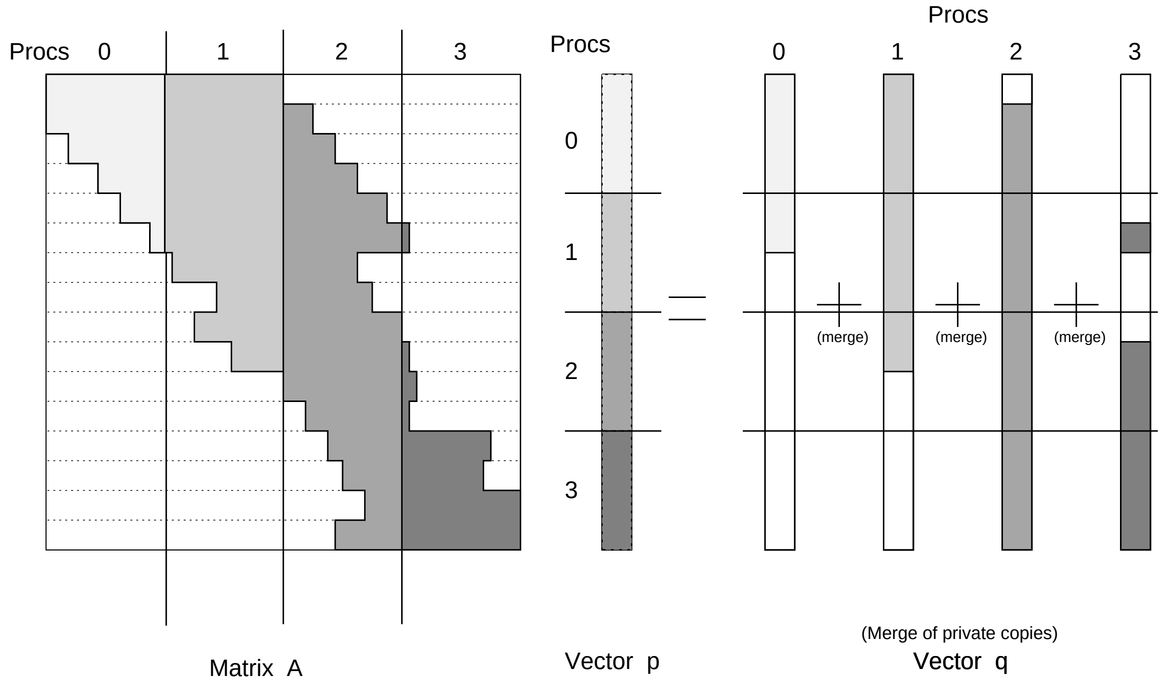
A(nz) contains the nonzero elements stored in the order of their columns from 1 to n; **row(nz)** stores the row numbers of each nonzero element; jth entry of **col(n+1)** points to the first entry of the j'th column in **A** and **row**.

Communications for Matrix Vector Multiply (Row)



Communication requirements of (sparse)Matrix vector multiplication where A is distributed in a (BLOCK, *) fashion.

Communications for Matrix Vector Multiply (Column):



Communication requirements of (sparse)Matrix vector multiplication where A is distributed in a (*, BLOCK) fashion.

HPF version of sparse storage CG (CSR format):

```
      REAL, dimension(1:nz) :: A
      INTEGER, dimension(1:nz) :: col
      INTEGER, dimension(1:n+1)  :: row
      REAL, dimension(1:n) :: x, r, p, q
!HPF$ PROCESSORS :: PROCS(NP)
!HPF$ DISTRIBUTE (BLOCK) :: q, p, r, x
!HPF$ DISTRIBUTE A(BLOCK)
!HPF$ DISTRIBUTE col(BLOCK)
!HPF$ DISTRIBUTE row(CYCLIC((n+1)/np)

      (usual initialisation of variables)
```

HPF version of sparse storage CG (CSR format) - Ctd.

```
DO k=1,Niter
  rho0 = rho
  rho = DOT_PRODUCT(r, r)  ! sdot
  beta = rho / rho0
  p = beta * p + r    ! saypx

  q = 0.0             ! sparse mat-vect multiply
  FORALL( j=1:n )
    DO i = row(j), row(j+1)-1
      q(j) = q(j) + A(i) * p(col(i))
    END DO
  END FORALL

  alpha = rho / DOT_PRODUCT(p, q)
  x = x + alpha * p   ! saxpy
  r = r - alpha * q   ! saxpy
  IF ( stop_criterion ) EXIT
END DO
```

CG (Sparse Methods) Conclusions:

- key issue for a parallel CG algorithm is matrix-vector routine.
- this must be able to exploit data storage scheme - typically compressed storage of sparse array.
- Current HPF distribution directives only allow arrays to be distributed according to **regular** structures such as BLOCK and CYCLIC.
- Whilst this is adequate for dense or regularly structured problems it does not provide the necessary flexibility for the efficient storage and manipulation of arbitrarily sparse matrices.
- Syntactic additions for HPF-2 may address this deficiency.

NPAC Projects Exploring HPF Applicability:

- Blackhole Binaries
- Weather/Climate Optimal Data Interpolation / Assimilation
- Parallel Shear Flow Code and Acoustic Equations implementation in HPF
- Work with ICASE on TLNS3D code

Overall Conclusions:

- HPF (potentially) allows: faster computation on parallel and distributed computers;
- additional code portability and ease of maintainance by comparison with message-passing implementations.
- Disadvantages (in common with any parallel implementation) over serial implementations are additional temporary data-storage requirements of parallel algorithms.
- HPF compilation systems now actually available (Digital, Portland, APR,...)
- HPF-2 language definition may address many of deficiencies now being uncovered by applications.

HPF Applications Web Package - HPFA Project:

- to help clarify omissions in the (HPF1) language;
- to provide a set of applications which include particular features of the language and so can be used to test the performance of compilers on these language features;
- to give the user community a set of exemplars which will help them learn to use HPF effectively.

Contents of the HPFA Package:

- List of Available Compilers, translators,...
- List of Industrial and Academic application areas with an indication of appropriate software including suitability of High Performance Fortran.
- List of generic exemplar applications codes with discussion of issues of relevance to HPF and HPF+.
- List of papers and books on HPF, Fortran90 and associated parallel computing issues;
- On-line HPF Tutorial;
- Answers to commonly asked questions on HPF;
- Talks and Lectures on HPF.

HPF References and Citations:

- List held at: http://www.npac.syr.edu/hpfa/bibl.html
- Papers and References and citations for HPF and HPF Applications efforts;
- Other Web Repositories like this one;
- Books on HPF or Relevant Data Parallel Applications (Koelbel's et al 'HPF Handbook'; Foster's 'Designing and Building Parallel Programs'; Fox's et al 'Parallel Computing Works!')

Summary of Online Internet Resources:

- **http://www.npac.syr.edu/hpfa** HPFA Project material at NPAC.
- **http://www.netlib.org/nse/home.html** The National HPCC Software Exchange.
- email: hpfaman@npac.syr.edu