Skip to main content

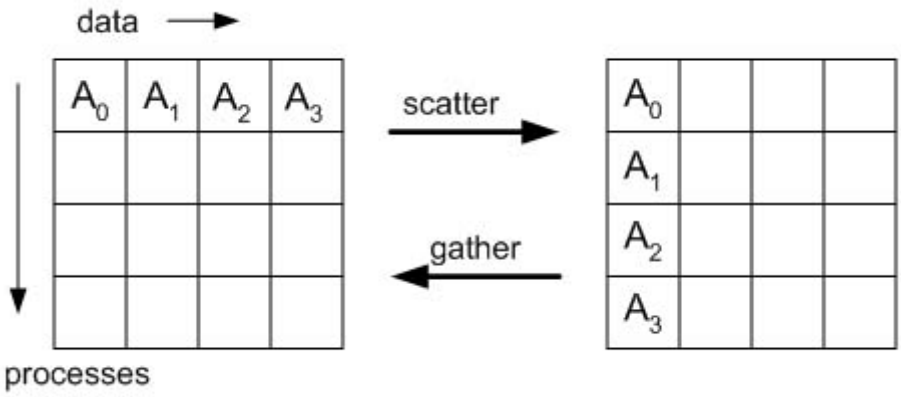
- Home
- Topics
- Reference
- Glossary
- Help
- Notebook

Virtual Workshop

Welcome guest
Log in (Globus)
Log in (other)
Try the quiz before you start
MPI Collective Communications
Introduction Goals Prerequisites
Characteristics Three Types of Routines Barrier Synchronization Data Movement • Broadcast • Gather and Scatter • Gather/Scatter Effect • Gatherv and Scatterv • Allgather • All to All Global Computing • Reduce • Scan • Operations and Example • Allreduce Mini-Exercise Nonblocking Routines • Nonblocking Example Performance Issues • Two Ways to Broadcast • Two Ways to Scatter Application Example • Scatter vs. Scatterv • Scatterv Syntax
Exercise Quiz
Short survey

MPI Collective Communications: Gather/Scatter Effect

In order to illustrate the gather and scatter functions, we give a matrix-style depiction below:



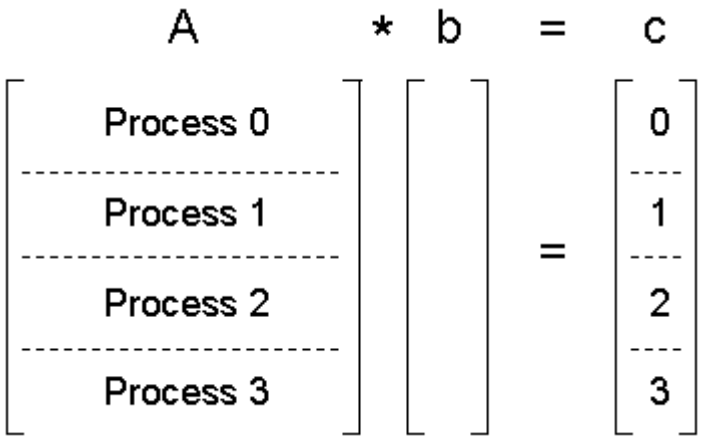
Let's consider in detail how MPI_Gather might be used to faciliate a distributed matrix computation.

Example: matrix-vector multiplication

- Matrix is distributed by rows (i.e. row-major order)
- Product vector is needed in entirety by one process
- MPI_Gather will be used to collect the product from processes

Description of Sample Code

The problem associated with the following sample code is the multiplication of a matrix A, size 100x100, by a vector b of length 100. The example uses four MPI processes, so each process will work on its own chunk of 25 rows of A. Since b is the same for each process, it will simply be replicated across processes. The vector c will therefore have 25 elements calculated by each process; these are stored in cpart. Here is a a picture of how the overall computation is distributed:



A: matrix distributed by rows
b: vector shared by all processes
c: vector updated by each process independently

The MPI_Gather routine will retrieve cpart from each process and store the result in ctotal, which is the complete vector c.

Sample Code in C

```
double a[25,100],b[100],cpart[25],ctotal[100];
int root;
root=0;
for(i=0;i<25;i++)
{
   cpart[i]=0;
   for(k=0;k<100;k++)
   {
      cpart[i]=cpart[i]+a[i,k]*b[k];
   }
}
MPI_Gather(cpart,25,MPI_REAL,ctotal,25,MPI_REAL,root,
     MPI_COMM_WORLD);
```

Sample Code in FORTRAN

```
! Fortran, unlike C, stores matrices in column-major order,
! so the A below is the transpose of the actual A matrix.
DIMENSION A(100,25), b(100), cpart(25), ctotal(100)
INTEGER root
DATA root/0/
DO I=1,25
     cpart(I) = 0.
     DO K=1,100
```

```
          cpart(I) = cpart(I) + A(K,I)*b(K)
      END DO
END DO
CALL MPI_GATHER(cpart, 25, MPI_REAL, ctotal, 25, MPI_REAL, &
      root, MPI_COMM_WORLD, ierr)
```

<= previous next =>

Add my notes
Mark (M) my place in this topic