Tutorial on Scatter

Here is a serial program to calculate the sum of numbers from a file:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define BUFSIZE 256

int main(int argc, char *argv[])
{ int *data, result;
  int size, i;
  FILE *infile;
  char buf[BUFSIZE];

  printf("What is the name of the data file?\n");
  fgets(buf,BUFSIZE,stdin);
  buf[strlen(buf)-1]='\0';    // Remove the carriage return from the file
name
  infile = fopen(buf,"r");
  if (infile==NULL)
  { perror ("Opening file");
    exit(1);
  }
  fscanf(infile,"%d", &size); // Find out how big the data is
  printf("Size = %d\n",size);
  data = (int *)malloc(sizeof(int)*size); // Allocate the space
  for (i=0; i<size; i++)      // Read the data
  { fscanf(infile,"%d",&data[i]);
    printf("%d ",data[i]);
  }
  printf("\n");
  result = 0;
  for (i=0; i<size; i++)
    result += data[i];
  printf("Result: %d\n",result);
}
```

Convert this into MPI program:

- Add the standard MPI administrative functions.
- Make new variables eachSize, eachData and eachResult in order for each processor to calculate their own part of the calculation.
- Process 0 needs to ask the user for the input file and read in the data.
- Broadcast the size of the data.
- Figure out how much of the data will divide evenly between the number of processors. (eachSize).
- Scatter the data into eachData.
- Change the calculation to calculate eachResult from eachData and eachSize.
- Reduce eachResult into result.
- Process 0 needs to calculate the leftover data.
- Process 0 needs to tell the result to the user.

Tutorial on Gather

Given this serial program:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define BUFSIZE 256

int main()
{ int i, size;
  char filename[BUFSIZE];
  FILE *outfile;

  printf("What is the file name you want to write to? ");
  scanf("%s", filename);
  printf("How many long integers to generate? ");
  scanf("%d", &size);
  outfile = fopen(filename,"w");
  if (!outfile)
  { fprintf(stderr,"Unable to open %s for writing.\n", filename);
    exit(1);
  }
  fprintf(outfile,"%d\n",size);
  srandom(time(0));
  for (i=0; i<size; i++)
    fprintf(outfile,"%ld\n", random());
  fclose(outfile);
}
```

Convert this to an MPI program. You will need to generate the random numbers into an array, and send them back to process 0 to write to the file. You will also need to have process 0 take care of any left-overs in case the number of long integers to generate does not divide exactly by the number of processes.