```
Script started on 2021-10-07 13:41:34-04:00
[TERM="xterm-256color" TTY="/dev/pts/1" COLUMNS="80"
LINES="24"]
]0;bjk47@remotel2:
~/Desktop/proj04/mandel/chunks[01;32mbjk47@remotel2[00
m:[01;34m~/Desktop/proj04/mandel/chunks[00m$ cat
mandel.c
/* Compute/draw mandelbrot set using MPI/MPE commands,
paralellized in chunks
 *
 * Written Winter, 1998, W. David Laverell.
 *
 * Main edited by Brea Koenes on 10/7/21
 *
 * Refactored Winter 2002, Joel Adams.
 */

#include <stdio.h>
#include <stdbool.h>
#include <math.h>
#include <sys/time.h>
#include <mpi.h>
#include <mpe.h>
#include "display.h"
#include <stdlib.h>

/* compute the Mandelbrot-set function for a given
 *  point in the complex plane.
 *
 * Receive: doubles x and y,
 *          complex c.
 * Modify: doubles ans_x and ans_y.
 * POST: ans_x and ans_y contain the results of the
mandelbrot-set
 *       function for x, y, and c.
 */
void compute(double x, double y, double c_real, double
c_imag,
             double *ans_x, double *ans_y)
{
       *ans_x = x*x - y*y + c_real;
       *ans_y = 2*x*y + c_imag;
}

/* compute the 'distance' between x and y.
 *
 * Receive: doubles x and y.
 * Return: x^2 + y^2.
 */
double distance(double x, double y)
{
       return x*x + y*y;
}


int main(int argc, char* argv[])
{
   const int  WINDOW_HEIGHT = 900;
   const int  WINDOW_WIDTH  = 1200;
   const double SPACING     = 0.003;

   int        n        = 0,
              ix       = 0,
              iy       = 0,
              button   = 0,
              id       = 0;
   double     x        = 0.0,
              y        = 0.0,
              c_real   = 0.0,
              c_imag   = 0.0,
              x_center = 1.16,
              y_center = -0.10;

   MPE_XGraph graph;

   // initialize new varibales
   int * myChunk;
   int* totalChunks;
   int numProcesses, start, stop = -1;
   int count, size;
   double startTime, endTime, chunkSize2 = 0;

   MPI_Init(&argc,&argv);
   MPI_Comm_rank(MPI_COMM_WORLD, &id);
   MPI_Comm_size(MPI_COMM_WORLD,
&numProcesses);

   // allocate memory
   myChunk = (int
*)malloc(WINDOW_HEIGHT*WINDOW_WIDTH*sizeof(int));
   totalChunks = (int
*)malloc(WINDOW_HEIGHT*WINDOW_WIDTH*sizeof(int));

   // begin time
   startTime = MPI_Wtime();

   // calculate chunk size
   int chunkSize1 = WINDOW_WIDTH/numProcesses;
   int remainder = WINDOW_WIDTH % numProcesses;

   // if remainder == 0 or remainder > id, go ahead
   // otherwise spread chunk size among rest of processes
and reduce chunk size by 1
   if ((remainder == 0) || (remainder != 0 && id < remainder))
{
              start = id * chunkSize1;
              stop = start + chunkSize1;
   } else {
      int chunkSize2 = chunkSize1 + 1;
              start = (remainder * chunkSize1) + (chunkSize2 *
(id - remainder));
              stop = start + chunkSize2;
   }

   // if not chunkSize2, size uses chunkSize1
   // otherwise size uses chunksize2
   if (!chunkSize2) {
              size = chunkSize1 * WINDOW_HEIGHT;
   } else {
              size = chunkSize2 * WINDOW_HEIGHT;
   }

   for (iy = start; iy < stop; iy++)
   {
     for (ix = 0; ix < WINDOW_WIDTH; ix++)
     {
       c_real = (ix - 400) * SPACING - x_center;
       c_imag = (iy - 400) * SPACING - y_center;

       x = y = 0.0;
       n = 0;

       while (n < 50 && distance(x, y) < 4.0)
       {
         compute(x, y, c_real, c_imag, &x, &y);
         n++;
       }
       if (n < 50) {
         myChunk[ix + (iy-start)*WINDOW_WIDTH] = 0;
       } else {
```

```
            myChunk[ix + (iy-start)*WINDOW_WIDTH] = 1;
          }
          count++;
        }
    }
    // gather chunks into an array
    MPI_Gather(myChunk, size, MPI_INT, totalChunks, size,
            MPI_INT, 0, MPI_COMM_WORLD);

    // if id == 0 open graphics
    if (id == 0) {
      count = 0;
      MPE_Open_graphics( &graph, MPI_COMM_WORLD,
                    getDisplay(),
                    -1, -1,
                    WINDOW_WIDTH, WINDOW_HEIGHT, 0 );
      for (ix = 0; ix < WINDOW_WIDTH; ix++)
      {
         for (iy = 0; iy < WINDOW_HEIGHT; iy++)
         {
            // draw graphics
            if (totalChunks[ix + (iy*WINDOW_WIDTH)])
            {
               MPE_Draw_point(graph, ix, iy, MPE_BLACK);
            }
            else
            {  MPE_Draw_point(graph, ix, iy, MPE_PINK);
            }
            count++;
         }
       }
       // calculate time
       endTime = MPI_Wtime() - startTime;

       // print to console
       printf("\nClick in the window to continue...\n");
       printf("\nTotal time: %f\n", endTime);

       MPE_Get_mouse_press( graph, &ix, &iy, &button );
       MPE_Close_graphics(&graph);
    }
    free(myChunk);
    free(totalChunks);

    MPI_Finalize();
    return 0;
}

]0;bjk47@remotel2:
~/Desktop/proj04/mandel/chunks[01;32mbjk47@remotel2[00
m:[01;34m~/Desktop/proj04/mandel/chunks[00m$ make
make: 'mandel' is up to date.
]0;bjk47@remotel2:
~/Desktop/proj04/mandel/chunks[01;32mbjk47@remotel2[00
m:[01;34m~/Desktop/proj04/mandel/chunks[00m$ makecat
mandel.cd [Kmake run NP=16[K
mpirun.mpich -np 1  ./mandel

Click in the window to continue...

Total time: 0.175760
X connection to :10.0 broken (explicit kill or server
shutdown).

make: *** [Makefile:23: run] Error 1
]0;bjk47@remotel2:
~/Desktop/proj04/mandel/chunks[01;32mbjk47@remotel2[00
m:[01;34m~/Desktop/proj04/mandel/chunks[00m$ make run
NP=1 16
mpirun.mpich -np 16  ./mandel

Click in the window to continue...

Total time: 0.720765
X connection to :10.0 broken (explicit kill or server
shutdown).

make: *** [Makefile:23: run] Error 1
]0;bjk47@remotel2:
~/Desktop/proj04/mandel/chunks[01;32mbjk47@remotel2[00
m:[01;34m~/Desktop/proj04/mandel/chunks[00m$ cd ..
]0;bjk47@remotel2:
~/Desktop/proj04/mandel[01;32mbjk47@remotel2[00m:[01;3
4m~/Desktop/proj04/mandel[00m$ cd slices
]0;bjk47@remotel2:
~/Desktop/proj04/mandel/slices[01;32mbjk47@remotel2[00
m:[01;34m~/Desktop/proj04/mandel/slices[00m$ cat sli
mandel.c
/* Compute/draw mandelbrot set using MPI/MPE commands,
paralellized in slices
 *
 * Written Winter, 1998, W. David Laverell.
 *
 * Main edited by Brea Koenes on 10/7/21
 *
 * Refactored Winter 2002, Joel Adams.
 */

#include <stdio.h>
#include <stdbool.h>
#include <math.h>
#include <sys/time.h>
#include <mpi.h>
#include <mpe.h>
#include "display.h"
#include <stdlib.h>

/* compute the Mandelbrot-set function for a given
 *  point in the complex plane.
 *
 * Receive: doubles x and y,
 *          complex c.
 * Modify: doubles ans_x and ans_y.
 * POST: ans_x and ans_y contain the results of the
mandelbrot-set
 *       function for x, y, and c.
 */
void compute(double x, double y, double c_real, double
c_imag,
             double *ans_x, double *ans_y)
{
       *ans_x = x*x - y*y + c_real;
       *ans_y = 2*x*y + c_imag;
}

/* compute the 'distance' between x and y.
 *
 * Receive: doubles x and y.
 * Return: x^2 + y^2.
 */
double distance(double x, double y)
{
       return x*x + y*y;
}


int main(int argc, char* argv[])
{
    const int  WINDOW_HEIGHT = 900;
    const int  WINDOW_WIDTH  = 1200;
```

```
   const double SPACING     = 0.003;

   int        n        = 0,
              ix       = 0,
              iy       = 0,
              button   = 0,
              id       = 0;
   double     x        = 0.0,
              y        = 0.0,
              c_real   = 0.0,
              c_imag   = 0.0,
              x_center = 1.16,
              y_center = -0.10;

   MPE_XGraph graph;

   // initialize new varibales
   int * mySlices;
   int* totalSlices;
   int numProcesses = -1;
   double startTime, endTime = 0;

   MPI_Init(&argc,&argv);
   MPI_Comm_rank(MPI_COMM_WORLD, &id);
   MPI_Comm_size(MPI_COMM_WORLD,
&numProcesses);

   // allocate memory
   mySlices = (int
*)malloc(WINDOW_HEIGHT*WINDOW_WIDTH*sizeof(int));
   totalSlices = (int
*)malloc(WINDOW_HEIGHT*WINDOW_WIDTH*sizeof(int));

   // begin time
   startTime = MPI_Wtime();

   for (iy = id; iy < WINDOW_HEIGHT; iy+= numProcesses)
   {
     for (ix = 0; ix < WINDOW_WIDTH; ix++)
     {
       c_real = (ix - 400) * SPACING - x_center;
       c_imag = (iy - 400) * SPACING - y_center;

       x = y = 0.0;
       n = 0;

       while (n < 50 && distance(x, y) < 4.0)
       {
         compute(x, y, c_real, c_imag, &x, &y);
         n++;
       }
       if (n < 50) {
         mySlices[ix + (iy*WINDOW_WIDTH)] = 0;
       } else {
         mySlices[ix + (iy*WINDOW_WIDTH)] = 1;
       }
     }
   }
   // reduce slices into array
   MPI_Reduce(mySlices, totalSlices,
WINDOW_HEIGHT*WINDOW_WIDTH,
            MPI_INT, MPI_SUM, 0, MPI_COMM_WORLD);

   // if id == 0 open graphics
   if (id == 0) {
     MPE_Open_graphics( &graph, MPI_COMM_WORLD,
                        getDisplay(),
                        -1, -1,
                        WINDOW_WIDTH, WINDOW_HEIGHT, 0 );
     for (ix = 0; ix < WINDOW_WIDTH; ix++)
     {
       for (iy = 0; iy < WINDOW_HEIGHT; iy++)
       {
         // draw graphics
         if (totalSlices[ix + (iy*WINDOW_WIDTH)])
         {
           MPE_Draw_point(graph, ix, iy, MPE_BLACK);
         }
         else
         {  MPE_Draw_point(graph, ix, iy, MPE_PINK);
         }
       }
     }
     // calculate time
     endTime = MPI_Wtime() - startTime;

     //print to console
     printf("\nClick in the window to continue...\n");
     printf("\nTotal time: %f\n", endTime);

     MPE_Get_mouse_press( graph, &ix, &iy, &button );
     MPE_Close_graphics(&graph);
   }
   free(mySlices);
   free(totalSlices);

   MPI_Finalize();
   return 0;
}

]0;bjk47@remotel2:
~/Desktop/proj04/mandel/slices[01;32mbjk47@remotel2[00
m:[01;34m~/Desktop/proj04/mandel/slices[00m$ c\  make
mpicc.mpich -DMPE_GRAPHICS -c -Wall -ansi -pedantic
-std=c99  ./mandel.c
mpicc.mpich ./mandel.o display.o -o ./mandel -lmpe -lX11 -lm
]0;bjk47@remotel2:
~/Desktop/proj04/mandel/slices[01;32mbjk47@remotel2[00
m:[01;34m~/Desktop/proj04/mandel/slices[00m$ makecat
mandel.c[3Pd slices[4P..make run NP=16[K
mpirun.mpich -np 1  ./mandel

Click in the window to continue...

Total time: 0.167805
X connection to :10.0 broken (explicit kill or server
shutdown).

make: *** [Makefile:23: run] Error 1
]0;bjk47@remotel2:
~/Desktop/proj04/mandel/slices[01;32mbjk47@remotel2[00
m:[01;34m~/Desktop/proj04/mandel/slices[00m$ make run
NP=1[Kcat mandel.c[8Pmake run NP=16
mpirun.mpich -np 16  ./mandel

Click in the window to continue...

Total time: 1.378152
X connection to :10.0 broken (explicit kill or server
shutdown).

make: *** [Makefile:23: run] Error 1
]0;bjk47@remotel2:
~/Desktop/proj04/mandel/slices[01;32mbjk47@remotel2[00
m:[01;34m~/Desktop/proj04/mandel/slices[00m$ cd ..
]0;bjk47@remotel2:
~/Desktop/proj04/mandel[01;32mbjk47@remotel2[00m:[01;3
4m~/Desktop/proj04/mandel[00m$ cd MW
```

```
]0;bjk47@remotel2:
~/Desktop/proj04/mandel/MW[01;32mbjk47@remotel2[00m:[
01;34m~/Desktop/proj04/mandel/MW[00m$ cat mandel.c
/* Compute/draw mandelbrot set using MPI/MPE commands,
paralellized using a master-worker pattern
 *
 * Written Winter, 1998, W. David Laverell.
 *
 * Main edited by Brea Koenes on 10/7/21
 *
 * Refactored Winter 2002, Joel Adams.
 */

#include <stdio.h>
#include <stdbool.h>
#include <math.h>
#include <sys/time.h>
#include <mpi.h>
#include <mpe.h>
#include "display.h"
#include <stdlib.h>

/* compute the Mandelbrot-set function for a given
 *  point in the complex plane.
 *
 * Receive: doubles x and y,
 *          complex c.
 * Modify: doubles ans_x and ans_y.
 * POST: ans_x and ans_y contain the results of the
mandelbrot-set
 *       function for x, y, and c.
 */
void compute(double x, double y, double c_real, double
c_imag,
             double *ans_x, double *ans_y)
{
       *ans_x = x*x - y*y + c_real;
       *ans_y = 2*x*y + c_imag;
}

/* compute the 'distance' between x and y.
 *
 * Receive: doubles x and y.
 * Return: x^2 + y^2.
 */
double distance(double x, double y)
{
       return x*x + y*y;
}


int main(int argc, char* argv[])
{
   const int  WINDOW_HEIGHT = 900;
   const int  WINDOW_WIDTH  = 1200;
   const double SPACING     = 0.003;

   int        n       = 0,
              ix      = 0,
              iy      = 0,
              button  = 0,
              id      = 0;
   double     x       = 0.0,
              y       = 0.0,
              c_real  = 0.0,
              c_imag  = 0.0,
              x_center = 1.16,
              y_center = -0.10;

   MPE_XGraph graph;

   // initialize new varibales
   int * myMW;
   int numProcesses, next;
   double startTime, endTime = 0;

   MPI_Init(&argc,&argv);
   MPI_Comm_rank(MPI_COMM_WORLD, &id);
   MPI_Comm_size(MPI_COMM_WORLD,
&numProcesses);
   MPI_Status status;

   // allocate memory
   myMW = (int
*)malloc(WINDOW_HEIGHT*WINDOW_WIDTH*sizeof(int));

   // begin time
   startTime = MPI_Wtime();

  // if id is 0, perform process using the default approach
  if (id == 0) {
    MPE_Open_graphics( &graph, MPI_COMM_WORLD,
                                    getDisplay(),
                                    -1, -1,
                                    WINDOW_WIDTH,
WINDOW_HEIGHT, 0);

   for (int i = 0; i < WINDOW_HEIGHT; i++)
   {
     for (ix = 0; ix < WINDOW_WIDTH; ix++)
     {
       c_real = (ix - 400) * SPACING - x_center;
       c_imag = (iy - 400) * SPACING - y_center;

       x = y = 0.0;
       n = 0;

       while (n < 50 && distance(x, y) < 4.0)
       {
         compute(x, y, c_real, c_imag, &x, &y);
         n++;
       }
       if (n < 50) {
         myMW[ix] = 1;
         MPE_Draw_point(graph, ix, iy, MPE_PINK);
       } else {
         myMW[ix] = 0;
         MPE_Draw_point(graph, ix, iy, MPE_BLACK);
       }
     }
     // if id is not 0, calculate using the master-worker
approach
     if (id != 0) {
       // send to worker
            MPI_Send(myMW, WINDOW_WIDTH, MPI_INT,
0, iy, MPI_COMM_WORLD);
            // wait to recieve from worker
            MPI_Recv(&iy, 1, MPI_INT, 0, MPI_ANY_TAG,
MPI_COMM_WORLD, &status);
     }else{
            iy++;
     }
     // if status.MPI_TAG equals 0 or the processes are
greater than 1, break loop
     if(status.MPI_TAG == 0 || (id == 0 && numProcesses >
1)) {
            break;
     }
   }
```

```
   // if it is not process 1, calculate using the master-worker
approach
   if (numProcesses != 1) {

         for (next = numProcesses; next <
WINDOW_HEIGHT+(numProcesses-1); next++) {
               // wait to recieve from worker
                        MPI_Recv(myMW, WINDOW_WIDTH,
MPI_INT, MPI_ANY_SOURCE, MPI_ANY_TAG,
                                 MPI_COMM_WORLD, &status);

                        if (next >= WINDOW_HEIGHT) {
                              // send next to worker
                                    MPI_Send(&next, 1, MPI_INT,
status.MPI_SOURCE, 0, MPI_COMM_WORLD);
                        } else {
                              // send next to worker
                                    MPI_Send(&next, 1, MPI_INT,
status.MPI_SOURCE, 1, MPI_COMM_WORLD);
               }

               for (int x = 0; x < WINDOW_WIDTH; x++) {

               if (myMW[x]==1) {
                        MPE_Draw_point(graph, x,
status.MPI_TAG, MPE_PINK);
               } else {
                        MPE_Draw_point(graph, x,
status.MPI_TAG, MPE_BLACK);
               }
               }
            }
         }
         // calculate time
         endTime = MPI_Wtime() - startTime;

         // print to console
         printf("\nClick in the window to continue...\n");
         printf("\nTotal time: %f\n", endTime);

         MPE_Get_mouse_press( graph, &ix, &iy, &button );
         MPE_Close_graphics(&graph);
   }
   free(myMW);

   MPI_Finalize();
   return 0;
}

]0;bjk47@remotel2:
~/Desktop/proj04/mandel/MW[01;32mbjk47@remotel2[00m:[
01;34m~/Desktop/proj04/mandel/MW[00m$ c make
make: 'mandel' is up to date.
]0;bjk47@remotel2:
~/Desktop/proj04/mandel/MW[01;32mbjk47@remotel2[00m:[
01;34m~/Desktop/proj04/mandel/MW[00m$ makecat
mandel.c[7Pd MW..make run NP=16
mpirun.mpich -np 1  ./mandel

Click in the window to continue...

Total time: 0.118968
X connection to :10.0 broken (explicit kill or server
shutdown).

Script done on 2021-10-07 13:43:27-04:00
[COMMAND_EXIT_CODE="130"]
```