Parallel LB Diffusion Code

In which, as a counterpoint to abstract virtual generic templated meta-programming, we present a concrete *code to simulate on precisely two computational nodes using a 1D lattice Boltzmann model with $\omega = 1$ the heating of one end of a 1m copper bar for 0.1ms.*

```
#include <iostream>
#include <vector>
#include "mpi.h"
int main (int argc, char *argv[])
{
  MPI::Init(argc, argv);
  MPI_Status status;
  if (MPI::COMM_WORLD.Get_size()!=2) exit(0);
  int rank = MPI::COMM_WORLD.Get_rank();
  int rank_other = 1-rank;
  const int iterations = 2406;
  const int nx = 50; //size of each node's domain
  double swap; //propagation conduit
  std::vector<double> data(nx);
  std::vector<double> temp(nx);
  for (int it=0; it<iterations; ++it) { //run simulation; swap edge data
    if (rank==0)
      swap=data[nx-1]/6;
    else
      swap=data[0]/6;
    MPI_Send(&swap,1,MPI_DOUBLE,rank_other,1,MPI_COMM_WORLD);
    MPI_Recv(&swap,1,MPI_DOUBLE,rank_other,1,MPI_COMM_WORLD,&status);
    if (rank==0)
      temp[nx-1]=data[nx-2]/6+data[nx-1]*4/6+swap;
    else
      temp[0]=swap+data[0]*4/6+data[1]/6;
    for (int i=1;i<nx-1;++i)
      temp[i]=data[i-1]/6+data[i]*4/6+data[i+1]/6;
    if (rank==0) temp[0]=100; //constant temperature boundary
    data = temp;
  }
  if (rank==0) { //assemble and report entire domain's values
    std::vector<double> rec(nx);
    MPI_Recv(&rec[0],nx,MPI_DOUBLE,rank_other,1,MPI_COMM_WORLD,&status);
    std::cout << "x␣(cm)\tT(degrees␣C)\n";
    for (int i=0; i<nx; ++i)
      std::cout << i << '\t' <<  data[i] << '\n';
    for (int i=0; i<nx; ++i)
      std::cout << i+nx << '\t' <<  rec[i] << '\n';
  }
  else {
    MPI_Send(&data[0],nx,MPI_DOUBLE,rank_other,1,MPI_COMM_WORLD);
  }
  MPI::Finalize();
}
```

iainhaslam@exolete.com