# MPI and the C++ STL

# Parallel Matrix - Vector Product

- (See Live Octave Demo)

# C++ Implementation

- A Sparse Matrix Class

```
class matrix : public std::vector<std::map<int, double>>
{
public:
  matrix (int N) { (*this).resize (N); };

  std::vector<double>
  operator* (std::vector<double> x)
  {
    std::vector<double> out (x.size (), .0);
    for (int icol = 0; icol < (*this).size (); ++icol)
      for (auto jrow = (*this)[icol].begin (); jrow != (*this)[icol].end (); ++jrow)
        out[(*jrow).first] += (*jrow).second * x[(*jrow).first];
  };
};
```

# Parallelization

- MPI does not support C++ objects
  - We must write our own code to transmit (part of) a matrix
  - Write the method

    matrix::send (int target, int tag, int firstcol, int lastcol, MPI_Comm comm)
  - Write the static method

    matrix

    matrix::recv (int source, int tag, MPI_Comm comm)
  - Implement the parallel product algorithm in C++