# High Performance Computing Applications for Material Physics

Matthew Johnson
REMRSEC REU Program
Summer 2011

Advisor:
Dr. Timothy Kaiser
Director, Golden Energy Computing Organization

"Well, Mr. Frankel, … , began to suffer from the computer disease that anybody who works with computers now knows about. It's a very serious disease and it interferes completely with the work. The trouble with computers is you 'play' with them. They are so wonderful.
…
But if you've ever worked with computers, you understand the disease - the 'delight' in being able to see how much you can do. But he got the disease for the first time, the poor fellow who invented the thing."
~Richard Feynman

GECO
Golden Energy Computing Organization

## SIESTA

### Powerful Modeling Tool

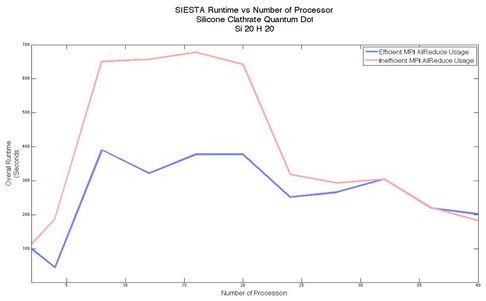
- Spanish Initiative for Electronic Simulations with Thousands of Atoms.
- Analyzes the electro-dynamics of a molecule or solid.
- Can evaluate the photo-electric characteristics of N-body systems, like Si clathrate QD's.

### Multiple Solution Methods

- SIESTA has two solvers for finding Eigenvalues.
- Default solver chosen based on the number of atoms in the molecule or solid being characterized.
- **Order-N Solver** – default solver for problems with more than 1000 atoms.
- **Diagon Solver** – standard diagonalization algorithm.

### Applications in Renewable Energy Materials

- SIESTA can be used to characterize the electrical properties of many different molecular structures.
- By offering multiple solution algorithms SIESTA is able to efficiently characterize molecules of many different sizes.
- This allows SIESTA to be scaled based on the size of the molecule it is characterizing.




## Anatomy of an MPI_AllReduce Call

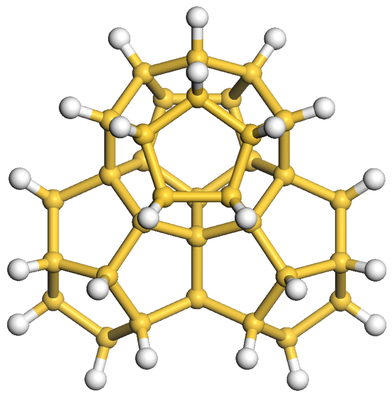
### MPI_AllReduce

MPI_AllReduce is a global data reduction operation

- Step 1: Each processor in the specified communicator calls MPI_Reduce, with its portion of the data. The specified operation is performed on the data and the result is sent to the processor which has been designated root.
- Step 2: The processor designated as root performs the specified reduction operation on the data sent to from the individual processors.
- Step 3: The result of the reduction operation performed in step 2 is disseminated across the communicator by a call to MPI_Broadcast.

### Use in SIESTA

- An analysis of SIESTA's runtime behavior was performed using Tuning and Analysis Utilities (TAU).
- The results show that SIESTA spends significantly more time in two communication routines than it does in other portion of the program. The offending routines were: MPI_AllReduce and MPI_Broadcast.
- Examination of SIESTA's source code revealed that two specific functions were calling MPI_AllReduce more frequently than necessary.
- Below are two examples of how to use MPI_AllReduce to find the sum of two number.



## Communication with MPI

### Message Passing Interface

- Used to pass messages between processors on parallel systems.
- Contains many commonly used parallel communication routines, considered a staple on HPC systems.
- Open source and platform/compiler specific distributions are available.
- MPI was designed with an emphasis on minimizing the overhead required for communication.

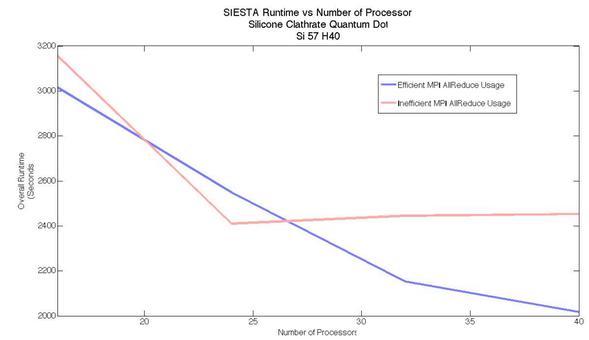
### Types of Operations

- Passing data between processing elements i.e. send, and receive.
- Dissemination of data across groups of processing elements i.e. broadcast, scatter.
- Data reductions, combinations, and simplifications i.e. summations, products, locating maxima and minima, logical operations (AND, OR, XOR, NOT).
- Most complex operations are simply combinations of the simpler send/receive operations and other system operations.

## MPI_AllReduce Call Examples

```
C MPI_AllReduce Calls
C Language: Fortran 77
C Efficient MPI_AllReduce Call
      integer var1,var2,combine(2),combine2(2),MPIerror
      combine(1)=var1
      combine(2)=var2
      call MPI_AllReduce(combine,combine2,2,MPI_integer,MPI_Sum,
     . MPI_Comm_World,MPIerror)
      var1=combine2(1)
      var2=combine2(2)

C Inefficient MPI_AllReduce Call
      integer var1,var2,MPIerror,temp
      call MPI_AllReduce(var1,temp,1,MPI_integer,MPI_Sum,
     . MPI_Comm_World,MPIerror)
      var1=temp
      call MPI_AllReduce(var2,temp,1,MPI_integer,MPI_Sum,
     . MPI_Comm_World,MPIerror)
      var2=temp
```




## Results

### Scalability Results

- The data collected from the Si 57 and Si 20 quantum dots suggests that the optimal number of processors is higher for the version of SIESTA which uses the more efficient MPI_AllReduce calls.
- In the case of the Si 20 quantum dot the optimal number of processors was doubled.
- For the Si 57 quantum dot, the maximum number of processors was increased to 166% of the initial maximum number.

### Overall Performance Improvements

- In addition to improving the scalability of SIESTA, more efficient use of MPI_AllReduce also improved SIESTA's overall runtime.
- For the Si 57 quantum dot overall runtime was approximately 83% of the initial (unmodified) value.
- The Si 20 quantum dot runtime showed the greatest improvement, running in approximately 25% of the time required to run the same model on an unmodified version of SIESTA .

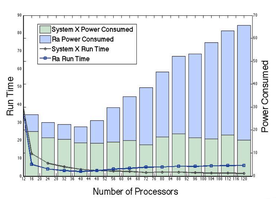
## Scalability and Communication

### Scalability

- Scalability is the measure of a computer programs ability to improve its performance as more resources (CPU's, GPU's, RAM, etc) are allocated to a problem.
- Many factors can affect a programs scalability, including: hardware, operating system, programming language.
- Information about a programs scalability is particularly important on HPC systems where resources are both indispensable and costly.

### Communication

- Scalability is also a way to determine how efficiently a program uses communication.
- Programs that scale well are often very effective communicators, passing only the necessary values between processors
- This allows the processors to spend most of their time calculating values and less time dealing with communication overhead.
- If a problem size remains fixed and the number of processors used to solve the problem is increased linearly, eventually the time spent in communication routines will drastically outweigh the amount of time spent in computation routines.




## Fixed SIESTA Source Code

```
#ifdef MPI
C Global reduction of Uscf/DUscf/Uatm
C Fixed to call all reduce once
C     call MPI_AllReduce(Uscf,Eloc,1,MPI_double_precision,MPI_Sum,
C    .  MPI_Comm_World,MPIerror)
C     Uscf = Eloc
C     call MPI_AllReduce(DUscf,Eloc,1,MPI_double_precision,MPI_Sum,
C    .  MPI_Comm_World,MPIerror)
C     DUscf = Eloc
C     call MPI_AllReduce(Uatm,Eloc,1,MPI_double_precision,MPI_Sum,
C    .  MPI_Comm_World,MPIerror)
C    Uatm = Eloc
      combine(1)=Uscf
      combine(2)=DUscf
      combine(3)=Uatm
      call MPI_AllReduce(combine,combine2,3,MPI_double_precision,
     . MPI_Sum, MPI_Comm_World,MPIerror)
      Uscf=combine2(1)
      DUscf=combine2(2)
      Uatm=combine2(3)
#endif
```

## Resources

| | |
|---|---|
| MPI Source Code | http://trac.mcs.anl.gov/projects/mpich2/browser/mpich2/trunk/src/mpi/coll/allreduce.c?rev=3733 |
| MPI_AllReduce Reference | http://www.mcs.anl.gov/research/projects/mpi/www/www3/MPI_Allreduce.html |
| MPI General Reference | http://switzernet.com/people/emin-gabrielyan/060708-thesis-ref/papers/Snir96.pdf |
| | http://www-uxsup.csx.cam.ac.uk/courses/MPI/paper_07.pdf |

MINES