FEVS: A Functional Equivalence Verification Suite

Stephen F. Siegel, Timothy K. Zirkel, Yi Wei

Supported by the National Science Foundation
under Grant No. CCF-0733035

Functional Equivalence

Definition: Suppose P_1 and P_2 are two computational programs with the same input/output signatures defining functions $f, g : X \to Y$. P_1 and P_2 are **functionally equivalent** if

$$\forall x \in X. f(x) = g(x)$$

Functional Equivalence

- Implementations might have additional inputs

$$f : X \to Y, g : X \times B \to Y$$

  - Equivalent if $\forall x \in X, b \in B . f(x) = g(x, b)$
- Nondeterminism
  - Internal is fine
  - Output should be deterministic

Functional Equivalence Verifiers

- Take two programs as input
- Return whether the outputs agree on all possible inputs
- Tools that target functional equivalence
  - TASS [http://vsl.cis.udel.edu/tass]
  - KLEE [http://klee.llvm.org]
  - TVOC [http://cs.nyu.edu/acsys/tv/]
  - isa [http://www.kotnet.org/~skimo/loop/isa-0.08.tar.bz2]

Scientific Computing

- Drug design
- Climate modeling
- Nanotechnology
- BioInformatics
- Cryptography
- Seismic processing
- Particle physics
- Data mining

Blue Gene/P

Source: Wikimedia Commons
Courtesy of Argonne National Laboratory

Scientific Computing

- MPI
  - Distributed memory via message passing
  - Dominant parallel programming framework in scientific computing
- OpenMP
  - Shared memory
- CUDA
  - NVIDIA's general purpose GPU programming API
- OpenCL
  - Heterogeneous processing environments

Blue Gene/P

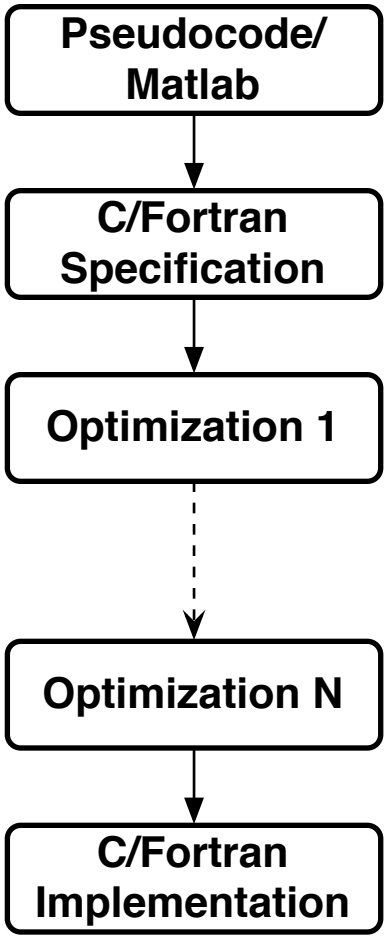
Source: Wikimedia Commons
Courtesy of Argonne National Laboratory

Why a Functional Equivalence Verification Suite is Needed

- Verify the verifiers
- Compare performance of verifiers
- Evaluate optimizations to a verifier
- Determine scope of applicability

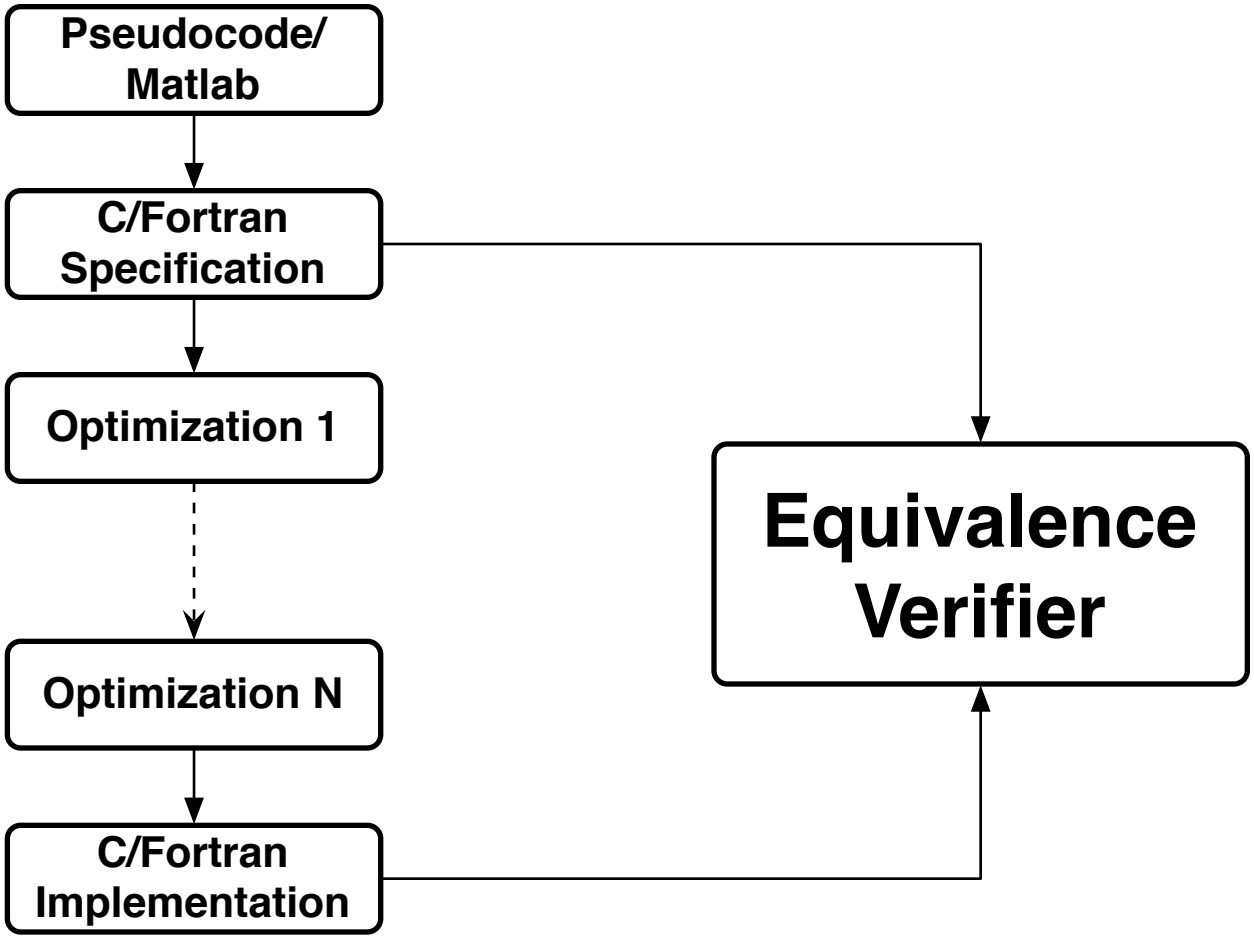
Criteria

- Categories
  - Grid-based
    - Finite difference techniques
  - Particle-based
  - Matrix algorithms
- Transformations
  - Loop Tiling
  - Parallelization (MPI)
    - manager-worker
    - non-blocking communication
    - domain decomposition
- Level of complexity
- One language for now
- Limited set of libraries

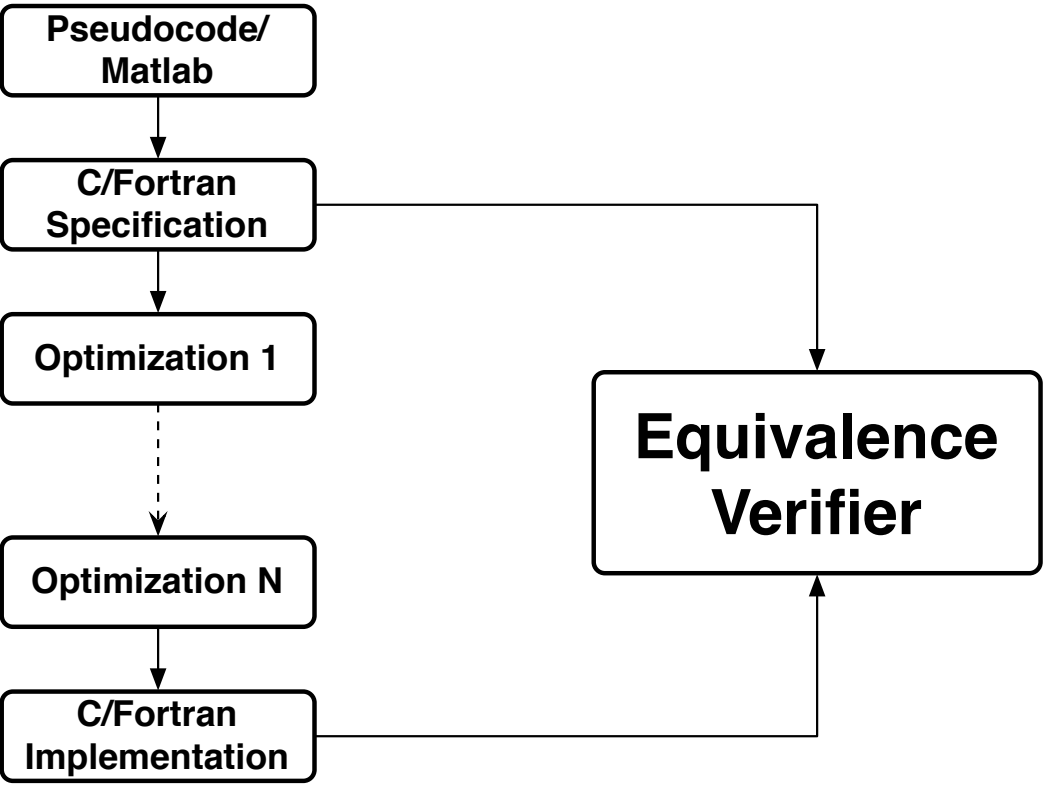


Typical Development Process

- Start simple
- Apply optimizations
  - Loop transformations
  - Parallelization
  - Usually done by hand
- Compare


Pseudocode/
Matlab
C/Fortran
Specification
Optimization 1
Optimization N
C/Fortran
Implementation
Equivalence
Verifier

This is hard!



- Undecidable
- Code is complicated
  - Pointer arithmetic
  - Parallelism
- Techniques
  - Symbolic execution
  - Model checking
  - Theorem proving

Types of Equivalence

- Floating-point
  - IEEE 754
  - Rounding errors
  - Not associative
- Real arithmetic
- Herbrand equivalent
  - Treat all functions as uninterpreted
  - No simplification
    - 0*x != 0

FEVS Program Groups

Each group consists of one specification and multiple implementations

- Adder
- Diffusion 1D
- Diffusion 2D
- Factorial
- Fibonacci Phi Calculation
- Gauss-Jordan Elimination
- Numerical Integration
- Laplace Equation
- Matrix Multiplication
- Mean
- N-body Simulation

Libraries

assert	math	stdlib
float	mpi	string
gd	stdio	time

Libraries

assert	math	stdlib
float	mpi	string
gd	stdio	time

Statistics

Filename	LoC	CC	MPI
adder_spec.c	14	3	
adder_par.c	40	6	S,R
adder_bad.c	40	6	S,R
mean_spec.c	18	3	
mean_impl.c	18	3	
mean_bad.c	18	3	
factorial_spec.c	11	3	
factorial_iter.c	9	2	
factorial_spec.c	11	3	
fib_spec.c	14	3	
fib_impl.c	14	3	
fib_bad.c	13	2	
matmat_spec.c	32	12	
matmat_vec.c	38	12	
matmat_mw.c	67	20	S,R,BC
matmat_tile.c	46	18	
matmat_bad.c	67	20	S,R,BC
diffusion1d_spec.c	149	32	
diffusion1d_par.c	253	57	S,R,SR,BC
diffusion1d_nb.c	263	63	S,R,IS,IR,WA
diffusion1d_bad.c	243	57	S,R,SR,BC

Filename	LoC	CC	MPI
diffusion2d_spec.c	172	24	
diffusion2d_par.c	252	42	S,R,SR
diffusion2d_bad.c	172	24	
laplace_spec.c	72	14	
laplace_rowdist.c	114	30	S,R,SR,AR
laplace_bad.c	114	30	S,R,SR,AR
gausselim_spec.c	101	27	
gausselim_rowdist.c	156	38	S,R,AR
gausselim_bad.c	101	27	
nbody_seq.c	289	46	
nbody_par.c	525	73	S,R,BC,IS,AR,B
nbody_bad.c	525	73	S,R,BC,IS,AR,B
integrate_spec.c	37	6	
integrate_mw.c	105	18	S,R
integrate_nb.c	122	17	IS,IR,W,BC,WN,WA
integrate_bad.c	105	18	S,R

S: MPI_Send
R: MPI_Recv
IS: MPI_Isend
IR: MPI_Irecv
SR: MPI_Sendrecv
BA: MPI_Barrier

BC: MPI_Bcast
AR: MPI_Allreduce
W: MPI_Wait
WA: MPI_Waitall
WN:MPI_Waitany

Examples

- Matrix Multiplication
- Diffusion 1D
- Gauss-Jordan Elimination
- N-body Simulation

Matrix Multiplication Specification

```
double A[L][M], B[M][N], C[L][N];
                    :
                    :
                    :
int i,j,k;
for (i = 0; i < L; i++)
  for (j = 0; j < N; j++) {
    C[i][j] = 0.0;
    for (k = 0; k < M; k++)
            C[i][j] += A[i][k] * B[k][j];
 }
```

Matrix Multiplication Tiled Version

```
int i, j, k, ii, jj, kk, hi1, hi2, hi3;
                        :
                        :
for (ii = 0; ii < L; ii+=TILE_SIZE) {
  for (jj = 0; jj < N; jj+=TILE_SIZE) {
    for (kk = 0; kk < M; kk+=TILE_SIZE) {
      hi1 = (ii + TILE_SIZE < L ? ii+TILE_SIZE : L);
      for (i = ii; i < hi1; i++) {
        hi2 = (jj + TILE_SIZE < N ? jj + TILE_SIZE : N);
        for (j = jj; j < hi2; j++) {
          hi3 = (kk + TILE_SIZE < M ? kk + TILE_SIZE : M);
          for (k = kk; k < hi3; k++)
            C[i][j] = C[i][j] + A[i][k] * B[k][j];
} } } } }
```

Matrix Multiplication
Factoring Out a Function

```
void vecmat(double vector[L], double matrix[L][M], double result[M]) {
  int j, k;
  for (j = 0; j < M; j++)
    for (k = 0, result[j] = 0.0; k < L; k++)
      result[j] += vector[k]*matrix[k][j];
}


int main(int argc, char *argv[]) {
            :
            :
            :
  for (i = 0; i < N; i++)
    vecmat(A[i], B, C[i]);
}
```

Matrix Multiplication Manager-Worker

```
MPI_Init(&argc, &argv);
MPI_Comm_rank(comm, &rank);
MPI_Comm_size(comm, &nprocs);
if (rank == 0) {
  double A[N][L], B[L][M], C[N][M], tmp[M];
  int count;


  MPI_Bcast(b, L*M, MPI_DOUBLE, 0, comm);
  for (count = 0; count < nprocs-1 && count < N; count++)
    MPI_Send(&A[count][0], L, MPI_DOUBLE, count+1, count+1, comm);
  for (i = 0; i < N; i++) {
    MPI_Recv(tmp, M, MPI_DOUBLE, MPI_ANY_SOURCE, MPI_ANY_TAG, comm, &status);
    for (j = 0; j < M; j++) C[status.MPI_TAG-1][j] = tmp[j];
    if (count < N) {
      MPI_Send(&A[count][0], L, MPI_DOUBLE, status.MPI_SOURCE, count+1, comm);
      count++;
    }
  }
  for (i = 1; i < nprocs; i++) MPI_Send(NULL, 0, MPI_INT, i, 0, comm);
  printMatrix(N, M, &C[0][0]);
} else {
  double B[L][M], in[L], out[M];

  MPI_Bcast(B, L*M, MPI_DOUBLE, 0, comm);
  while (1) {
    MPI_Recv(in, L, MPI_DOUBLE, 0, MPI_ANY_TAG, comm, &status);
    if (status.MPI_TAG == 0) break;
    vecmat(in, B, out);
    MPI_Send(out, M, MPI_DOUBLE, 0, status.MPI_TAG, comm);
  }
}
MPI_Finalize();
```

Manager

$$\begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \\ a_{20} & a_{21} \\ a_{30} & a_{31} \end{bmatrix} \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} = \begin{bmatrix} & \\ & \\ & \\ & \end{bmatrix}$$

Worker 1

$$\begin{bmatrix} & \end{bmatrix} \begin{bmatrix} & \\ & \end{bmatrix} = \begin{bmatrix} & \end{bmatrix}$$

Worker 2

$$\begin{bmatrix} & \end{bmatrix} \begin{bmatrix} & \\ & \end{bmatrix} = \begin{bmatrix} & \end{bmatrix}$$

Manager

$$\begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \\ a_{20} & a_{21} \\ a_{30} & a_{31} \end{bmatrix} \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} = \begin{bmatrix} & \\ & \\ & \\ & \end{bmatrix}$$

Worker 1

$$\begin{bmatrix} & \end{bmatrix} \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} = \begin{bmatrix} & \end{bmatrix}$$

Worker 2

$$\begin{bmatrix} & \end{bmatrix} \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} = \begin{bmatrix} & \end{bmatrix}$$

Manager

$$\begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \\ a_{20} & a_{21} \\ a_{30} & a_{31} \end{bmatrix} \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} = \begin{bmatrix} & \\ & \\ & \\ & \end{bmatrix}$$

Worker 1

$$\begin{bmatrix} a_{00} & a_{01} \end{bmatrix} \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} = \begin{bmatrix} & \end{bmatrix}$$

Worker 2

$$\begin{bmatrix} & \end{bmatrix} \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} = \begin{bmatrix} & \end{bmatrix}$$

Manager

$$\begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \\ a_{20} & a_{21} \\ a_{30} & a_{31} \end{bmatrix} \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} = \begin{bmatrix} & \\ & \\ & \\ & \end{bmatrix}$$

Worker 1

$$\begin{bmatrix} a_{00} & a_{01} \end{bmatrix} \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} = \begin{bmatrix} & \end{bmatrix}$$

Worker 2

$$\begin{bmatrix} a_{10} & a_{11} \end{bmatrix} \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} = \begin{bmatrix} & \end{bmatrix}$$

Manager

$$\begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \\ a_{20} & a_{21} \\ a_{30} & a_{31} \end{bmatrix} \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} = \begin{bmatrix} & \\ & \\ & \\ & \end{bmatrix}$$

Worker 1

$$\begin{bmatrix} a_{00} & a_{01} \end{bmatrix} \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} = \begin{bmatrix} c_{00} & c_{01} \end{bmatrix}$$

Worker 2

$$\begin{bmatrix} a_{10} & a_{11} \end{bmatrix} \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} = \begin{bmatrix} c_{10} & c_{11} \end{bmatrix}$$

Manager

$$\begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \\ a_{20} & a_{21} \\ a_{30} & a_{31} \end{bmatrix} \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} = \begin{bmatrix} c_{10} & c_{11} \\ & \\ & \\ & \end{bmatrix}$$

Worker 1

$$\begin{bmatrix} a_{00} & a_{01} \end{bmatrix} \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} = \begin{bmatrix} c_{00} & c_{01} \end{bmatrix}$$

Worker 2

$$\begin{bmatrix} & \end{bmatrix} \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} = \begin{bmatrix} & \end{bmatrix}$$

Manager

$$\begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \\ a_{20} & a_{21} \\ a_{30} & a_{31} \end{bmatrix} \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} = \begin{bmatrix} c_{10} & c_{11} \\ & \\ & \\ & \end{bmatrix}$$

Worker 1

$$\begin{bmatrix} a_{00} & a_{01} \end{bmatrix} \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} = \begin{bmatrix} c_{00} & c_{01} \end{bmatrix}$$

Worker 2

$$\begin{bmatrix} a_{20} & a_{21} \end{bmatrix} \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} = \begin{bmatrix} & \end{bmatrix}$$

Manager

$$\begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \\ a_{20} & a_{21} \\ a_{30} & a_{31} \end{bmatrix} \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} = \begin{bmatrix} c_{00} & c_{01} \\ c_{10} & c_{11} \\ & \\ & \end{bmatrix}$$

Worker 1

$$\begin{bmatrix} & \end{bmatrix} \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} = \begin{bmatrix} & \end{bmatrix}$$

Worker 2

$$\begin{bmatrix} a_{20} & a_{21} \end{bmatrix} \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} = \begin{bmatrix} & \end{bmatrix}$$

Manager

$$\begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \\ a_{20} & a_{21} \\ a_{30} & a_{31} \end{bmatrix} \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} = \begin{bmatrix} c_{00} & c_{01} \\ c_{10} & c_{11} \\ & \\ & \end{bmatrix}$$

Worker 1

$$\begin{bmatrix} a_{30} & a_{31} \end{bmatrix} \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} = \begin{bmatrix} & \end{bmatrix}$$

Worker 2

$$\begin{bmatrix} a_{20} & a_{21} \end{bmatrix} \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} = \begin{bmatrix} & \end{bmatrix}$$

Manager

$$\begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \\ a_{20} & a_{21} \\ a_{30} & a_{31} \end{bmatrix} \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} = \begin{bmatrix} c_{00} & c_{01} \\ c_{10} & c_{11} \\ & \\ & \end{bmatrix}$$

Worker 1

$$\begin{bmatrix} a_{30} & a_{31} \end{bmatrix} \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} = \begin{bmatrix} c_{30} & c_{31} \end{bmatrix}$$

Worker 2

$$\begin{bmatrix} a_{20} & a_{21} \end{bmatrix} \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} = \begin{bmatrix} & \end{bmatrix}$$

Manager

$$\begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \\ a_{20} & a_{21} \\ a_{30} & a_{31} \end{bmatrix} \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} = \begin{bmatrix} c_{00} & c_{01} \\ c_{10} & c_{11} \\ & \\ c_{30} & c_{31} \end{bmatrix}$$

Worker 1

$$\begin{bmatrix} \quad & \quad \end{bmatrix} \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} = \begin{bmatrix} \quad & \quad \end{bmatrix}$$

Worker 2

$$\begin{bmatrix} a_{20} & a_{21} \end{bmatrix} \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} = \begin{bmatrix} \quad & \quad \end{bmatrix}$$

Manager

$$\begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \\ a_{20} & a_{21} \\ a_{30} & a_{31} \end{bmatrix} \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} = \begin{bmatrix} c_{00} & c_{01} \\ c_{10} & c_{11} \\ & \\ c_{30} & c_{31} \end{bmatrix}$$

Worker 1

$$\begin{bmatrix} \quad & \quad \end{bmatrix} \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} = \begin{bmatrix} \quad & \quad \end{bmatrix}$$

Worker 2

$$\begin{bmatrix} a_{20} & a_{21} \end{bmatrix} \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} = \begin{bmatrix} c_{20} & c_{21} \end{bmatrix}$$

Manager

$$\begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \\ a_{20} & a_{21} \\ a_{30} & a_{31} \end{bmatrix} \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} = \begin{bmatrix} c_{00} & c_{01} \\ c_{10} & c_{11} \\ c_{20} & c_{21} \\ c_{30} & c_{31} \end{bmatrix}$$

Worker 1

$$\begin{bmatrix} \quad & \quad \end{bmatrix} \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} = \begin{bmatrix} \quad & \quad \end{bmatrix}$$

Worker 2

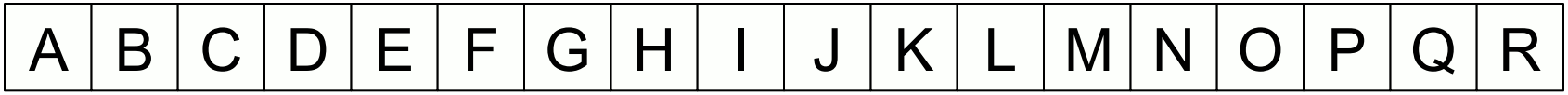
$$\begin{bmatrix} \quad & \quad \end{bmatrix} \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} = \begin{bmatrix} \quad & \quad \end{bmatrix}$$

Diffusion 1D

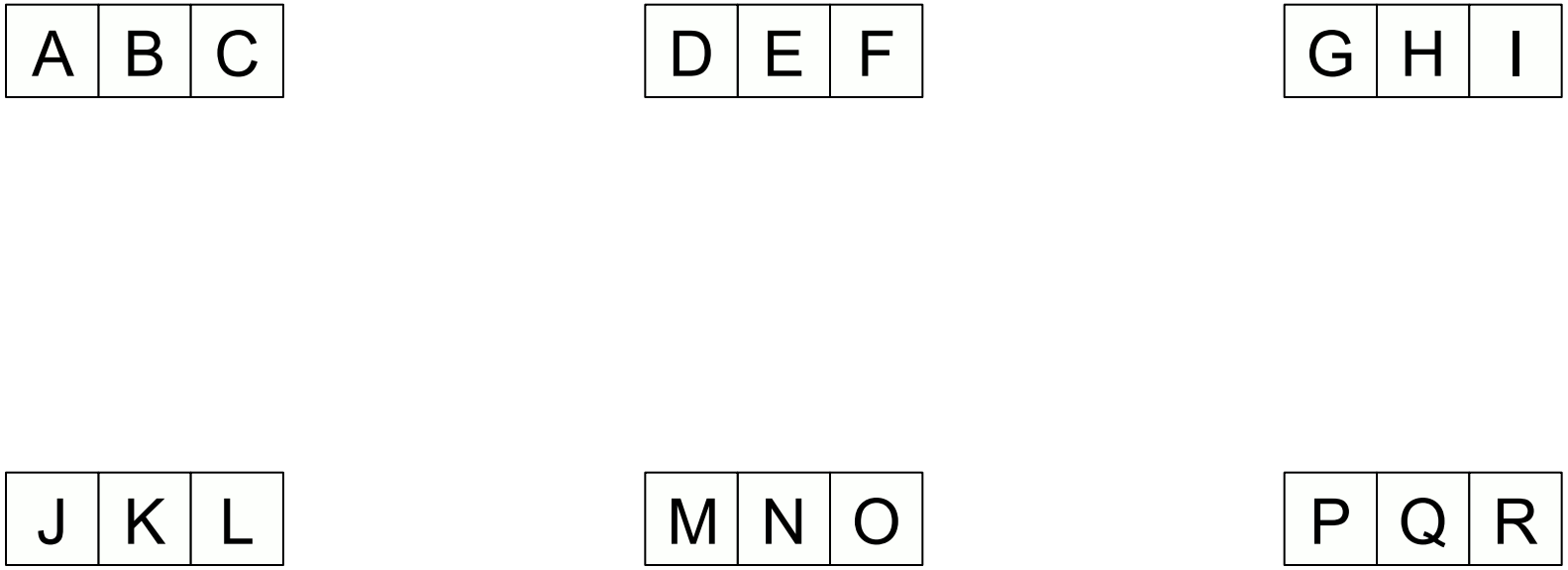
$$\frac{\partial u}{\partial t} = k \frac{\partial^2 u}{\partial x^2}$$

```
for (i = 1; i < nx-1; i++)
  u_new[i] =  u[i] + k*(u[i+1] + u[i-1] -2*u[i]);
```

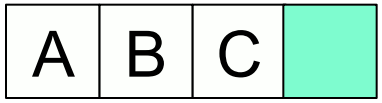
Diffusion 1D Sequential Version

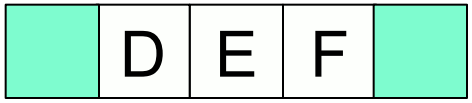


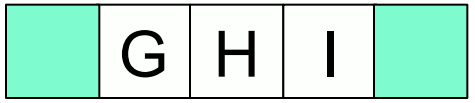
Diffusion 1D Parallel Version

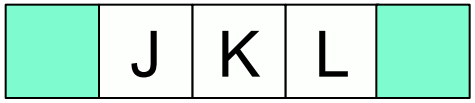


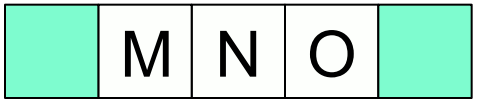
Diffusion 1D Parallel Version

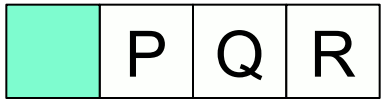




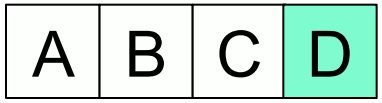


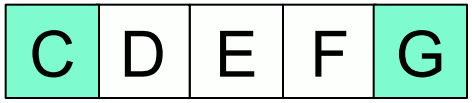


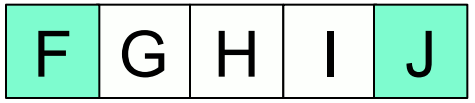


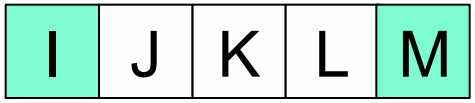


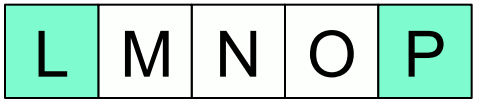
Diffusion 1D Parallel Version

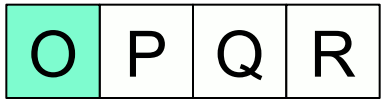




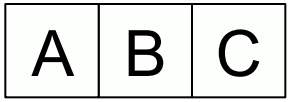




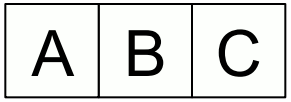




Diffusion 1D Sequential Version

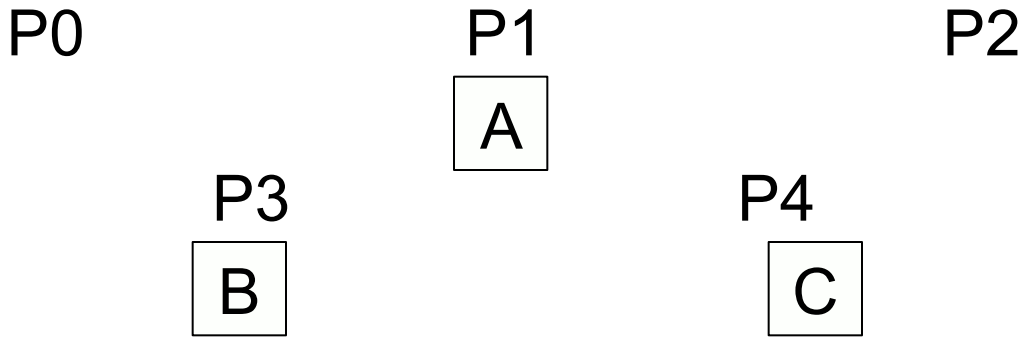


Diffusion 1D Parallel Version



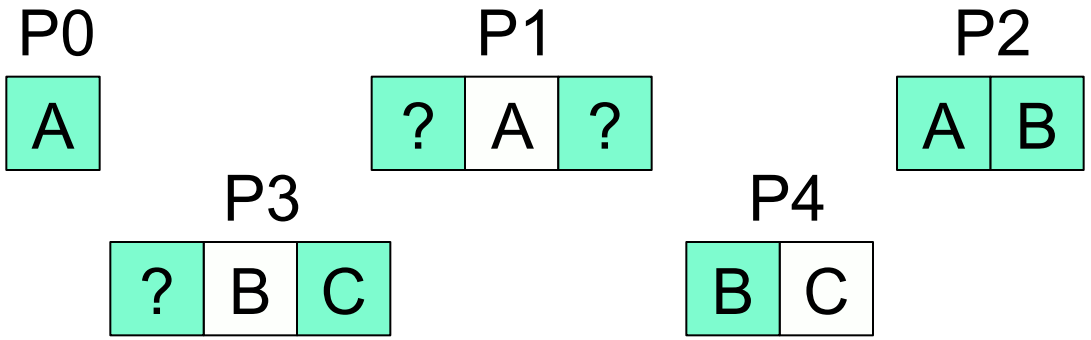

```
int owner(int index) {
  return (nprocs*(index+1)-1)/nx;
}
```

Diffusion 1D Parallel Version




```
int owner(int index) {
  return (nprocs*(index+1)-1)/nx;
}
```

Diffusion 1D Parallel Version




```
int owner(int index) {
  return (nprocs*(index+1)-1)/nx;
}
```

Guass-Jordan Elimination

Sequential

$$\begin{bmatrix} 2 & -1 & 1 & 3 \\ -1 & 2 & 1 & 1 \\ 3 & 2 & 3 & 0 \end{bmatrix}$$

Guass-Jordan Elimination

Sequential

$$\begin{bmatrix} 2 & -1 & 1 & 3 \\ -1 & 2 & 1 & 1 \\ 3 & 2 & 3 & 0 \end{bmatrix}$$

Parallel

P0 P1 P2

$$\begin{bmatrix} 2 & -1 & 1 & 3 \end{bmatrix} \quad \begin{bmatrix} -1 & 2 & 1 & 1 \end{bmatrix} \quad \begin{bmatrix} 3 & 2 & 3 & 0 \end{bmatrix}$$

Guass-Jordan Elimination

Sequential

```
pivot = 0.0;
for (; col < numCols; col++) {
  for (pivotRow = top; pivotRow < numRows; pivotRow++) {
    pivot = matrix[pivotRow*numCols + col];
    if (pivot) break;
  }
  if (pivot) break;
}
```

Parallel

```
for (; col < numCols; col++) {
  if (matrix[col] != 0.0 && rank >= top) {
    MPI_Allreduce(&rank, &pivotRow, 1, MPI_INT, MPI_MIN, MPI_COMM_WORLD);
  }
  else {
    MPI_Allreduce(&nprocs, &pivotRow, 1, MPI_INT, MPI_MIN, MPI_COMM_WORLD);
  }
  if (pivotRow < nprocs){
    break;
  }
}
```

Guass-Jordan Elimination

Sequential

```
if (pivotRow != top) {
  for (j = 0; j < numCols; j++) {
    tmp = matrix[top*numCols + j];
    matrix[top*numCols + j] = matrix[pivotRow*numCols + j];
    matrix[pivotRow*numCols + j] = tmp;
  }
}
```

Parallel

```
if (pivotRow != top) {
  if (rank == top) {
    MPI_Sendrecv_replace(matrix, numCols, MPI_DOUBLE, pivotRow, 0,
                         pivotRow, 0, MPI_COMM_WORLD, &status);
  }
  else if (rank == pivotRow) {
    MPI_Sendrecv_replace(matrix, numCols, MPI_DOUBLE, top, 0,
                         top, 0,  MPI_COMM_WORLD, &status);
  }
}
if (rank == top) {
  pivot = matrix[col];
}
```

Guass-Jordan Elimination

Sequential

```
for (j = col; j < numCols; j++) {
  matrix[top*numCols + j] /= pivot;
}
```

Parallel

```
if (rank == top) {
  for (j = col; j < numCols; j++) {
    matrix[j] /= pivot;
    toprow[j] = matrix[j];
  }
}
MPI_Bcast(toprow, numCols, MPI_DOUBLE, top, MPI_COMM_WORLD);
```

Guass-Jordan Elimination

Sequential

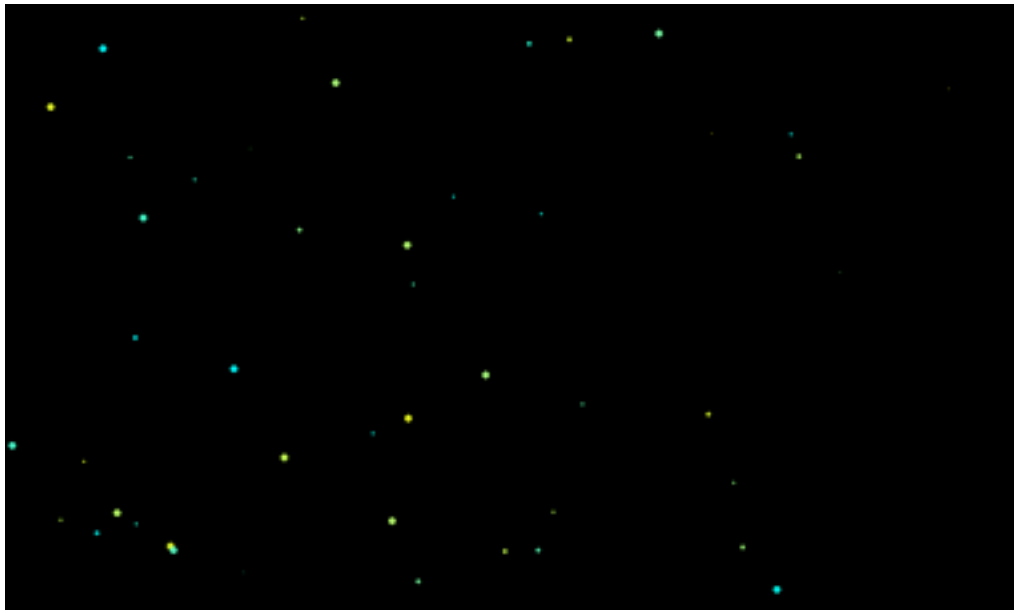
```
for (i = 0; i < numRows; i++) {
  if (i != top){
    tmp = matrix[i*numCols + col];
    for (j = col; j < numCols; j++) {
      matrix[i*numCols + j] -= matrix[top*numCols + j]*tmp;
    }
  }
}
```

Parallel

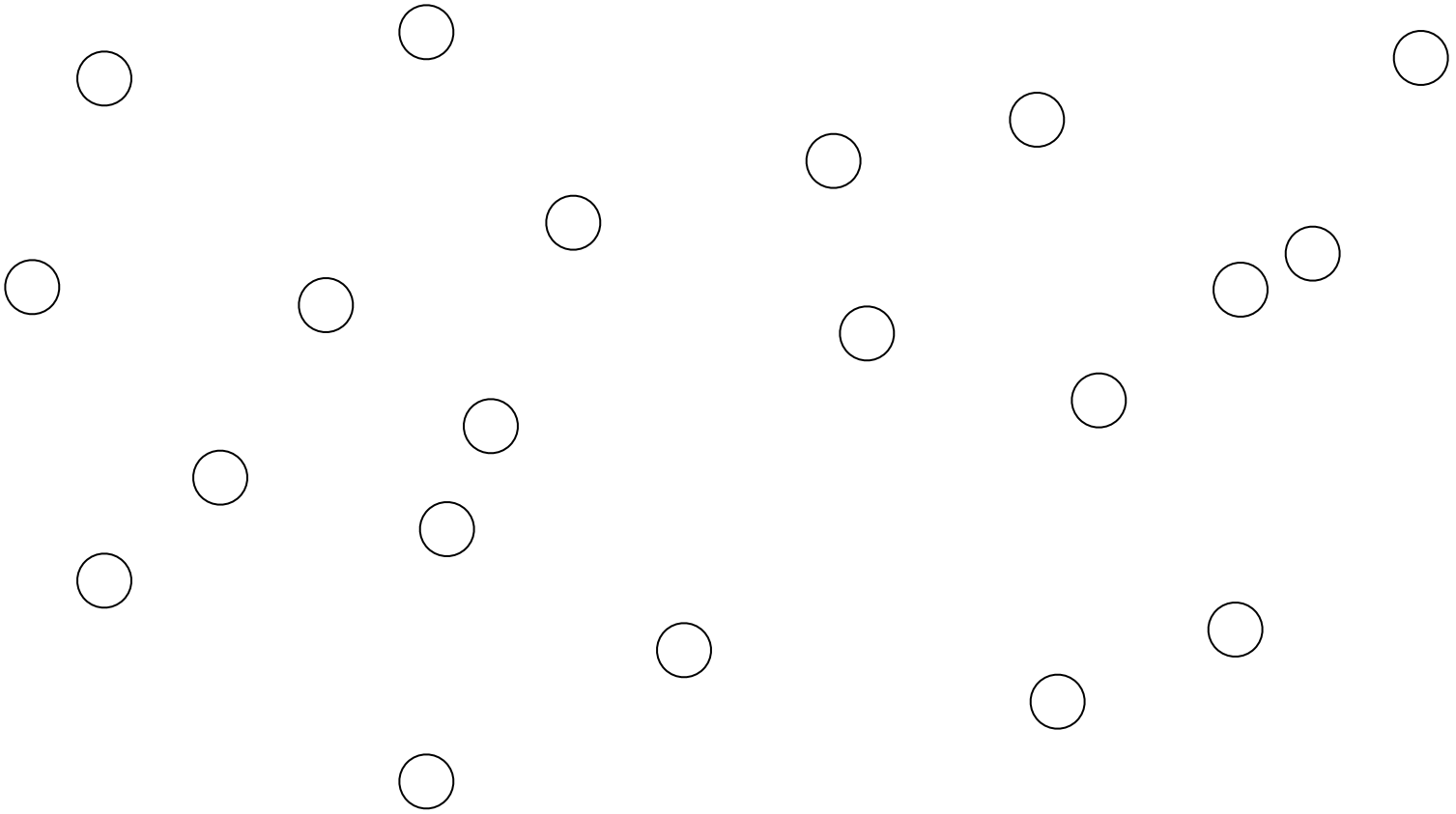
```
if (rank != top) {
  tmp = matrix[col];
  for (j = col; j < numCols; j++) {
    matrix[j] -= toprow[j]*tmp;
  }
}
```

N-body Simulation

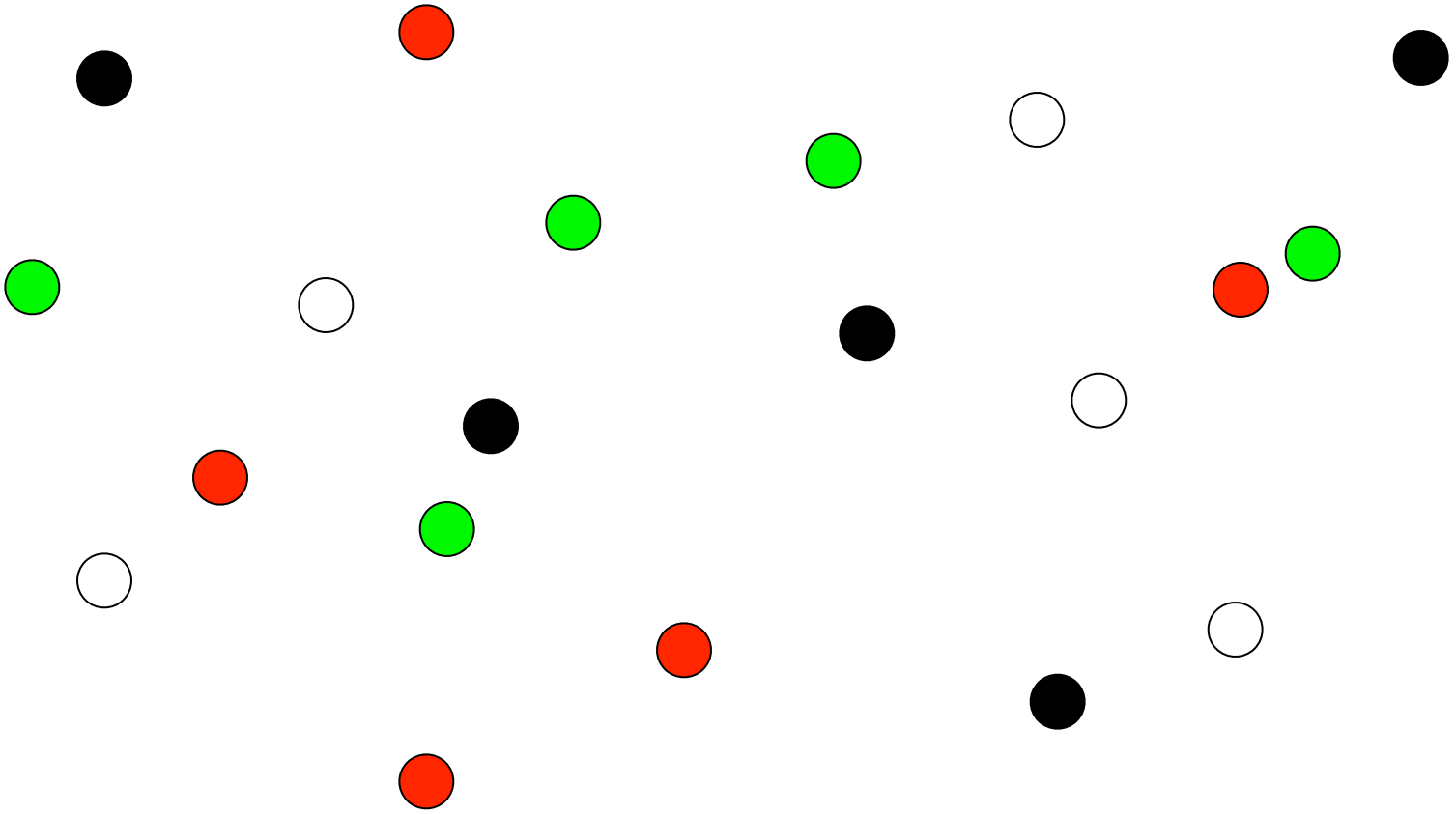
- Gravitational simulation
- Bodies distributed to processes
- Non-blocking and blocking communication



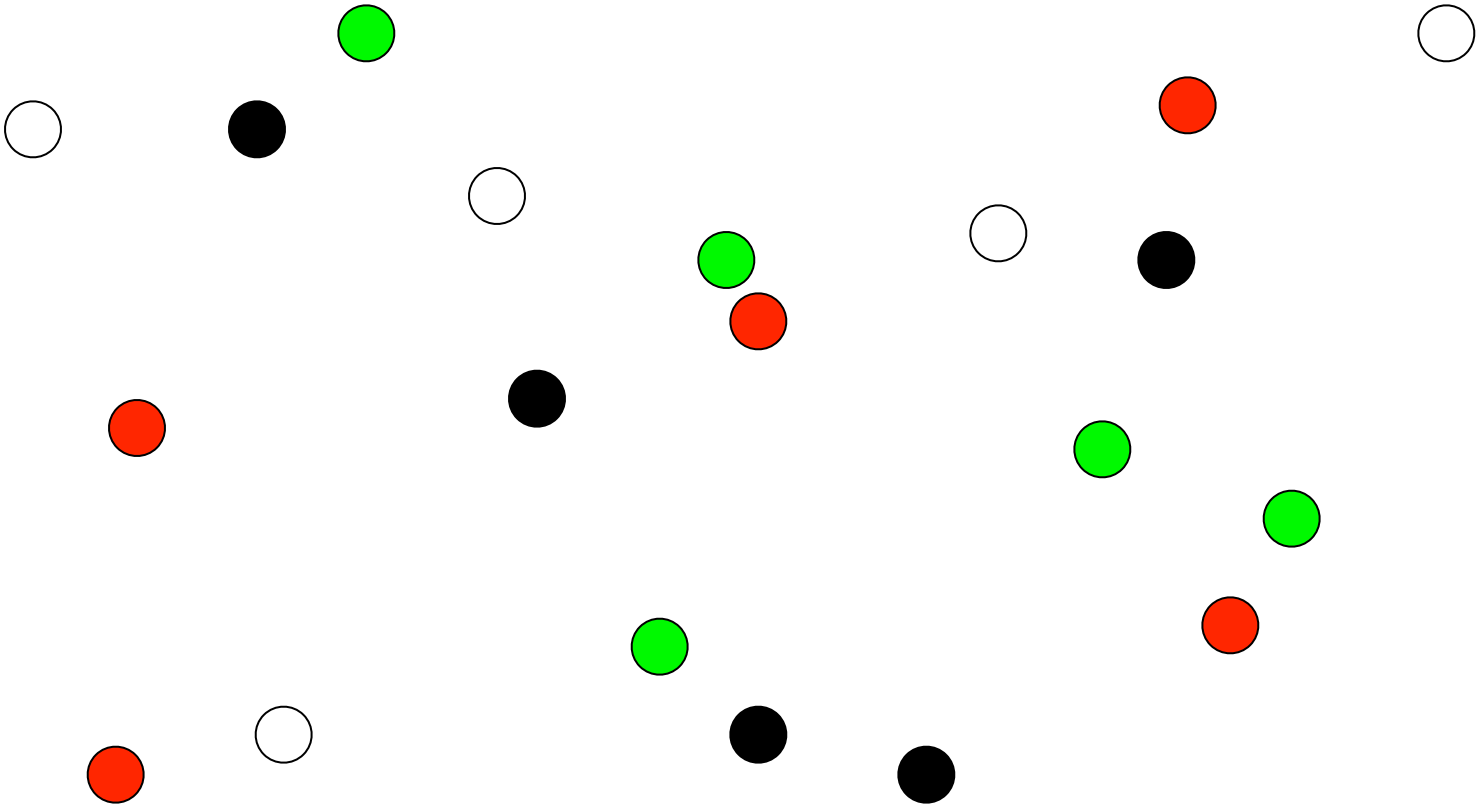
N-body Simulation



N-body Simulation



N-body Simulation



Future Work

- More complex examples
- OpenMP
- CUDA
- Subtle erroneous examples
- Fortran
- We welcome contributions!

For more information, visit:
http://vsl.cis.udel.edu/fevs