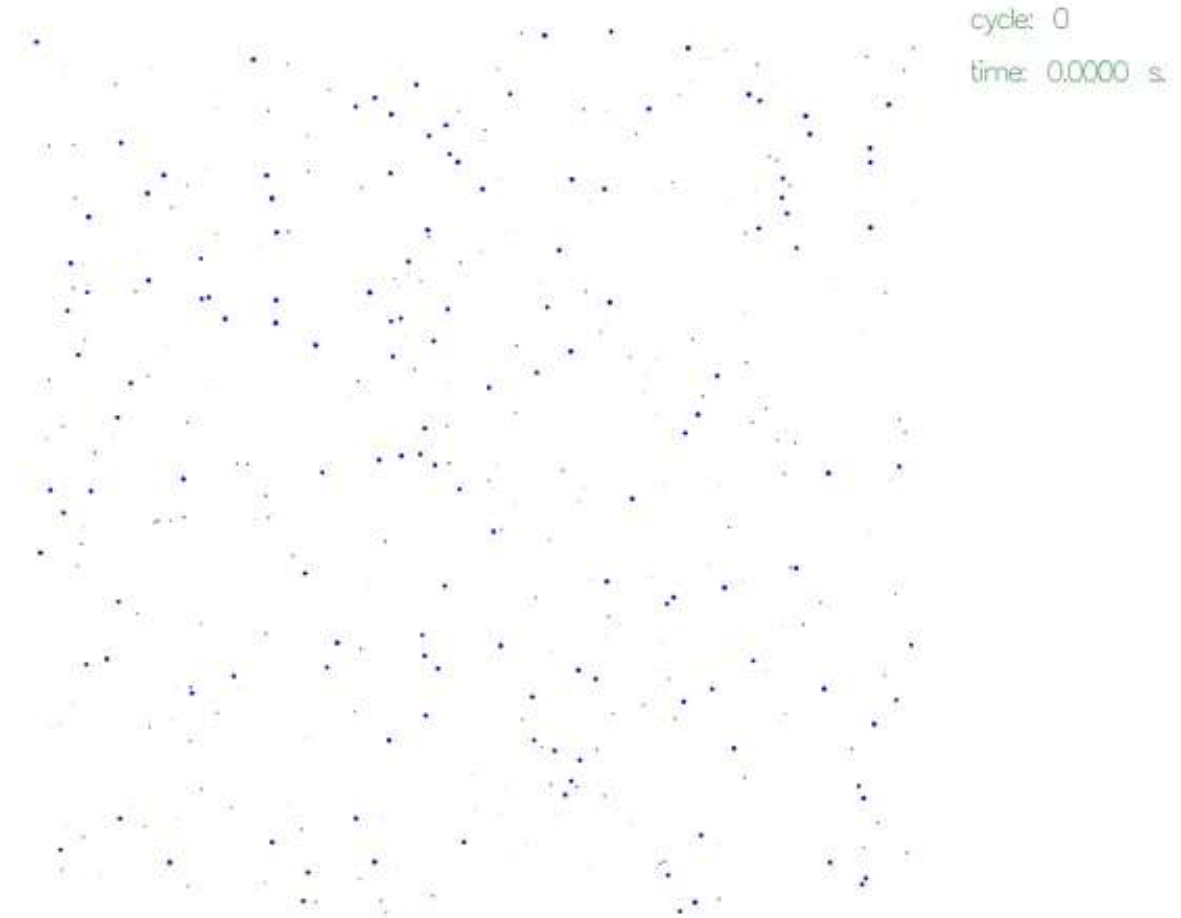


Cluster Computing

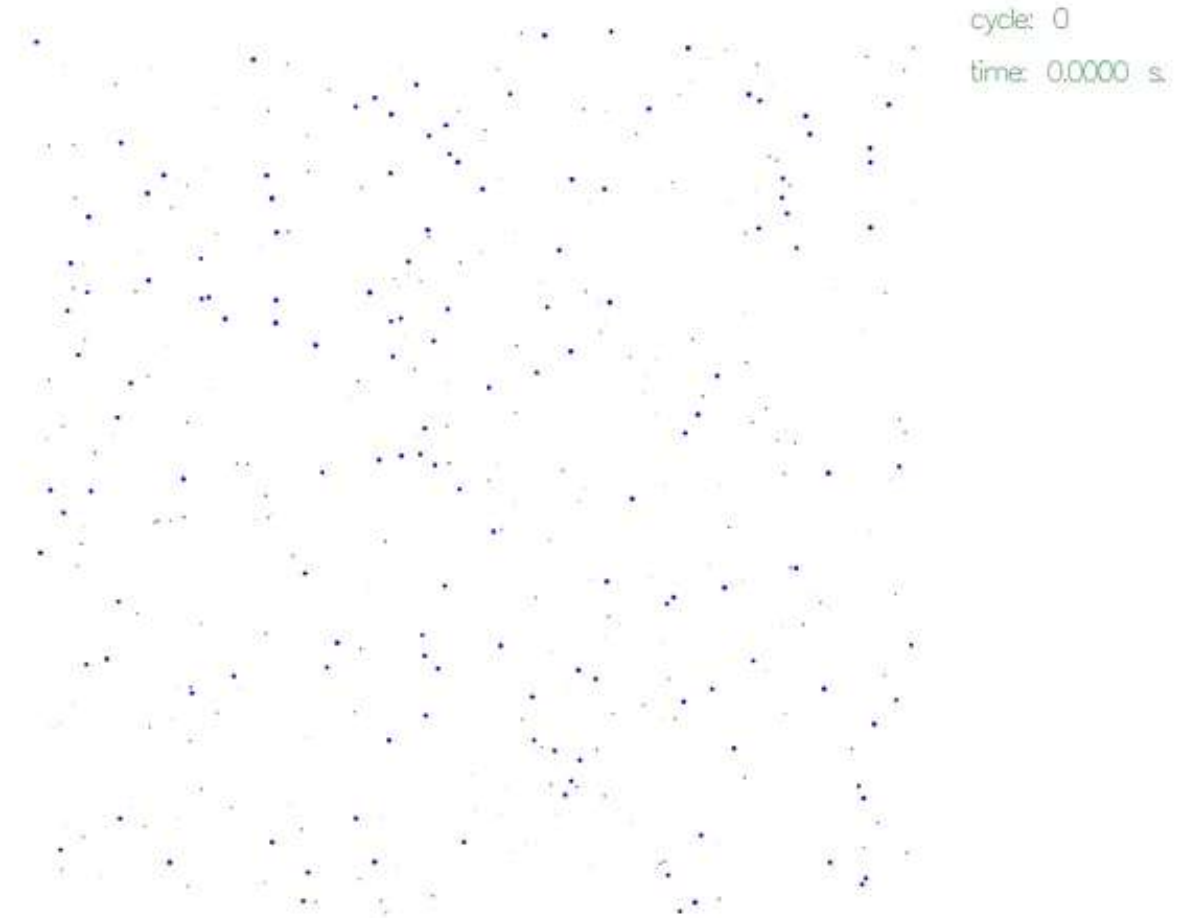
N-Body Problem

- Given N objects
 - Mass
 - Velocity
- Compute the status of each object
- Universal Gravitation
 - $O(N^2)$ forces



N-Body Problem

- Given N objects
 - Mass
 - Velocity
- Compute the status of each object
- Universal Gravitation
 - $O(N^2)$ forces
- We need to scale!
- (Or have a better algorithm)



Cluster Computing

- Putting many (cheap) computers in a cluster
 - The computers in the same cluster do not have to be the same
 - Communication topology
 - All nodes are fully connected
 - Hubs
- Eventually, the communication will be the bottleneck
- For most cases, a network filesystem is employed

Message Passing Interface (MPI)

- Introduced in early 90's
- Each process may have multiple threads
- Each process has its own address space
- Inter-process communication

MPI Example

```
#include <stdio.h>
#include <mpi.h>
int main(int argc, char *argv[])
{
    MPI_Init(&argc, &argv);
    printf("hello world!\n");
    MPI_Finalize();
    return 0;
}
```

MPI Example

```
#include <stdio.h>
#include <mpi.h>
int main(int argc, char *argv[])
{
    int rank, size;
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    printf("hello world from %d of %d!\n", rank, size);
    MPI_Finalize();
    return 0;
}
```

MPI Communications

```
int MPI_Send(  
    void* data,  
    int count,  
    MPI_Datatype datatype,  
    int destination,  
    int tag,  
    MPI_Comm communicator)
```

```
int MPI_Recv(  
    void* data,  
    int count,  
    MPI_Datatype datatype,  
    int source,  
    int tag,  
    MPI_Comm communicator,  
    MPI_Status* status)
```

MPI Communications

```
int MPI_Probe(  
    int source,  
    int tag,  
    MPI_Comm comm,  
    MPI_Status* status)  
int MPI_Get_count(  
    MPI_Status* status,  
    MPI_Datatype datatype,  
    int* count)
```

MPI Communications

```
int MPI_Isend(  
    const void *buf,  
    int count,  
    MPI_Datatype datatype,  
    int dest,  
    int tag,  
    MPI_Comm comm,  
    MPI_Request *request)
```

MPI Communications

```
int MPI_Wait(  
    MPI_Request *request,  
    MPI_Status *status)
```

```
int MPI_Test(  
    MPI_Request *request,  
    int *flag,  
    MPI_Status *status)
```

Communicator

```
int MPI_Comm_split(  
    MPI_Comm comm,  
    int color,  
    int key,  
    MPI_Comm * newcomm)
```

```
int MPI_Comm_free(MPI_Comm *comm)
```

Compilation and Execution

- MPICH, OpenMPI
 - mpicc, mpiCC, mpic++
 - mpiexec, mpirun
 - mpiexec -np 4 ./a.out
 - mpiexec --showme
-
- SLURM
 - sbatch
 - srun

Tensor Computing in Deep Learning

- Scalar
- Vector
- Matrix
- Tensor
 - Rank
 - Dimension

Tensor Computing in Deep Learning

Matrix

~~Tensor~~ Computing in Deep Learning

- Scalar
- Vector
- Matrix
- Tensor
 - Rank
 - Dimension

Matrix ~~Tensor~~ Computing in Deep Learning

- Scalar
- Vector
- Matrix
- Tensor
 - Rank
 - Dimension

- Matrix multiplication
- Non-linear activation
- Gradient descent

Matrix ~~Tensor~~ Computing in Deep Learning

- Scalar
- Vector
- Matrix
- Tensor
 - Rank
 - Dimension

- Matrix multiplication
- Non-linear activation
- ~~•Gradient descent~~

Graduate student descent

Tensor Operations

- Element-wise add
- Element-wise plus
- Element-wise division
- Hadamard product
- Matrix multiplication
- Batched matrix multiplication
- More linear algebra operations...
- Collect, Scatter, Reduce...

Libraries

- Numpy
- Blas
- cuBlas
- cuSparse
- MKL
- TensorFlow
- PyTorch
- MXNet
- ...

Lazy Evaluation and Code Generation

$c = a + b$

$d = c * 2$

for $i = 1$ to n do

$c[i] = a[i] + b[i]$

for $i = 1$ to n do

$d[i] = c[i] * 2$

for $i = 1$ to n do

$d[i] = (a[i] + b[i]) * 2$

Differentiation

- Numerical differentiation
- Symbolic differentiation
 - Chain rules
- Forward mode auto differentiation
- Reverse mode auto differentiation

Tensor

- Dense
 - Column major
 - Row major
 - Stride
- Sparse
 - Compressed representation
 - Set intersection