

# Strong Scaling Studies with UPIC

Viktor K. Decyk

UCLA

# UPIC Framework

Framework: a unified environment containing all components needed for writing code for a specific problem domain

Goal is rapid construction of new PIC codes by reusing tested modules: “Lego” pieces for developing new codes

Designed to support different programming styles

- Simple Fortran77 projects
- Complex, object-oriented, multi-author projects
- Student programmers, with many error checks

Supports multiple numerical methods, optimizations, different physics models, different types of hardware

Above all, hides parallel processing

# Codes using UPIC Framework

**QuickPIC:** Quasi-static code for plasma based accelerators

**QPIC:** Quantum PIC code

**Parallel DRACO:** 3D code for modeling ion propulsion.

**RECON3D:** 3D EM code for studying magnetic reconnection

**BEPS:** UPIC Test code

# Benchmarks Parameters

Benchmarks on Atlas, an AMD 2.4 GHz Opteron Cluster at LLNL, Infiniband Network, 9216 Cpus.

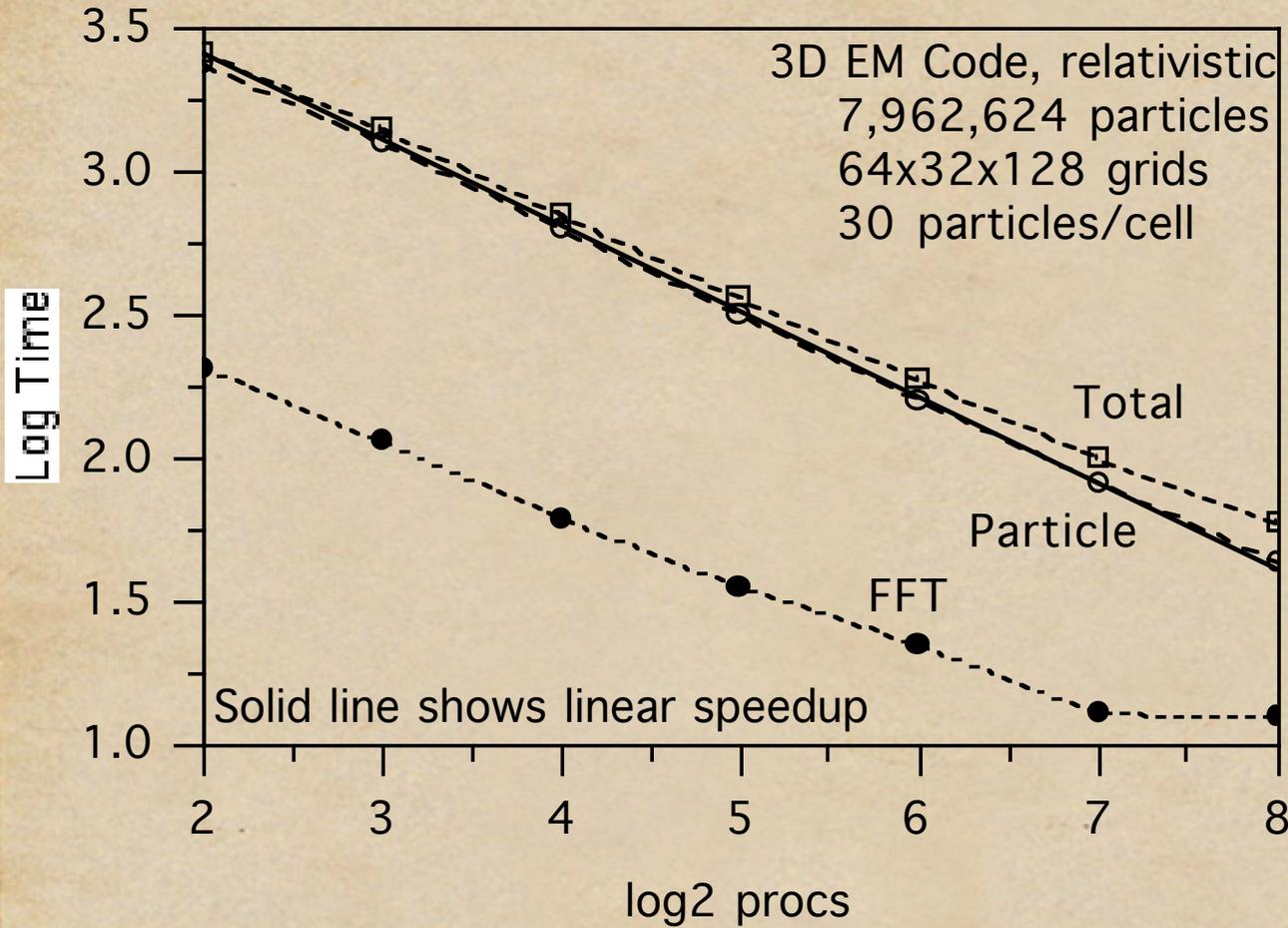
Small, Medium, Large, and Extra-Large Strong Scaling Studies (Fixed problem size) for 3400 time steps

3D Electromagnetic, Relativistic Code, Linear Interpolation. Uniform Plasma, 2D Partition. Most results double precision

Periodic Spectral Code, 10 FFTs per time step

Purpose: to identify where current algorithms break down

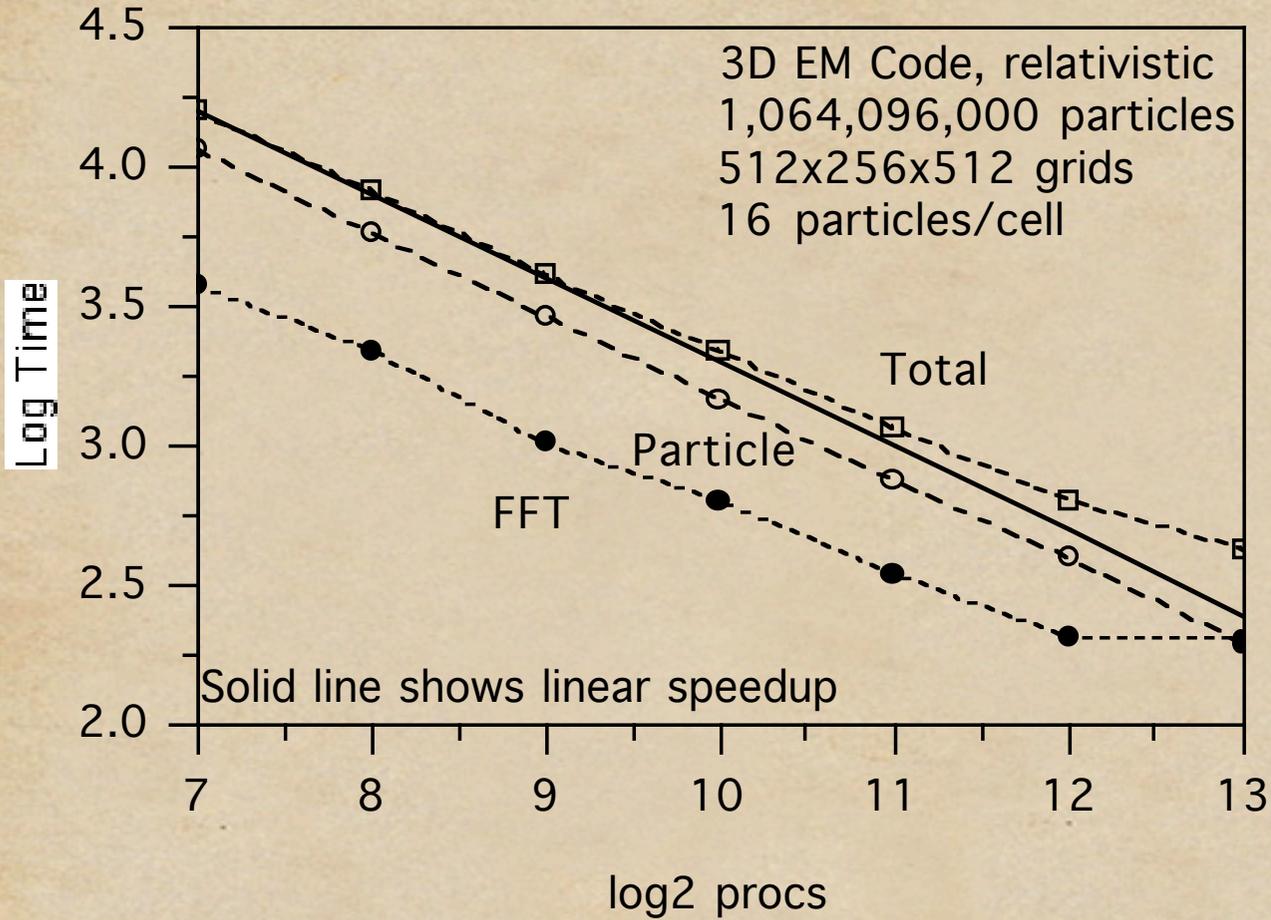
Log Time vs log2 procs



Measurements of Particle Push includes:

- Charge and Current Deposit
- Particle Acceleration and Sorting
- Particle Manager

### Log Time vs log2 procs



Large

Procs	Push Time	FFT Time	Loop Time
8192 proc:	53 psec.	199.2 sec.	421.9 sec.
4096 proc:	108 psec.	203.2 sec.	630.0 sec.
2048 proc:	204 psec.	342.5 sec.	1140.1 sec.
1024 proc:	401 psec.	625.8 sec.	2164.8 sec.
512 proc:	794 psec.	1003.6 sec.	4023.1 sec.
256 proc:	1.6 ns.	2174.9 sec.	8137.0 sec.
128 proc:	3.1 ns.	3698.6 sec.	15587.7 sec.

# Observations

Particle Push scales very well in all cases (better than 90%)

Particle Manager takes about 25% of the time, but scales

- Checking which particles to move takes as much time as moving them

Most of the FFT time (75%) is spent doing transpose

- Eventually message size in transpose becomes small and latency dominated

- For vector data, transposes for each component combined

# Double Precision FFT Benchmark

2048x2048x2048 real to complex UPIC FFT on Atlas

Procs	FFT Time
4096 proc:	0.74 sec.
2048 proc:	1.74 sec.
1024 proc:	3.36 sec.
512 proc:	7.79 sec.
256 proc:	15.02 sec.

2048x2048x2048 complex to complex Steve Plimpton's FFT  
on Franklin

Procs	FFT Time
4096 proc:	5.6 sec.
2048 proc:	9.2 sec.
1024 proc:	15.9 sec.
512 proc:	26.2 sec.
256 proc:	43.3 sec.

# Transpose Routines

Many algorithms can be parallelized by operating on one coordinate which is local (not distributed), then transposing and operating on the other coordinate, which is now local

FFT can be parallelized this way

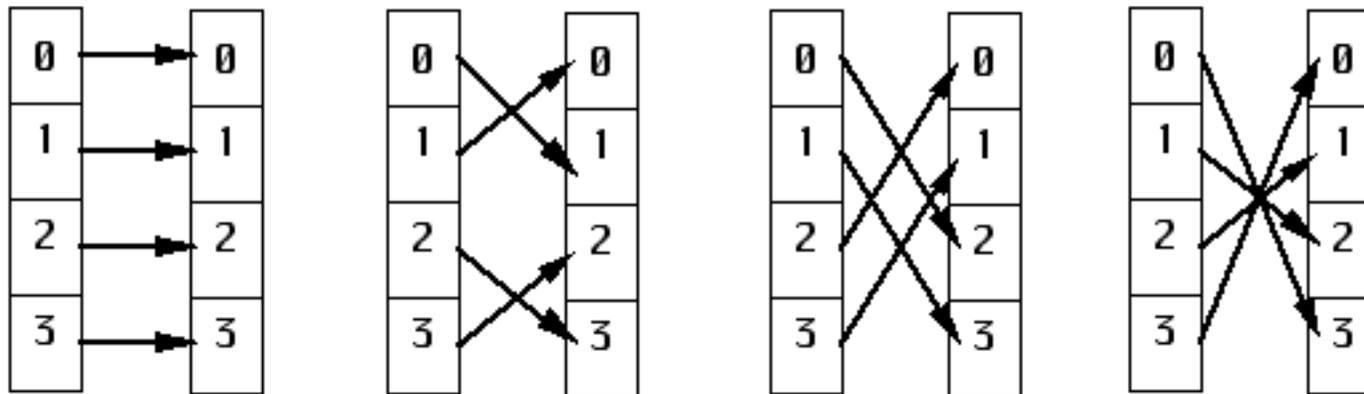
Also, mixed spectral and finite-difference methods

## Limitations

- Cannot use more processors than grids
- Assumes domains are regular

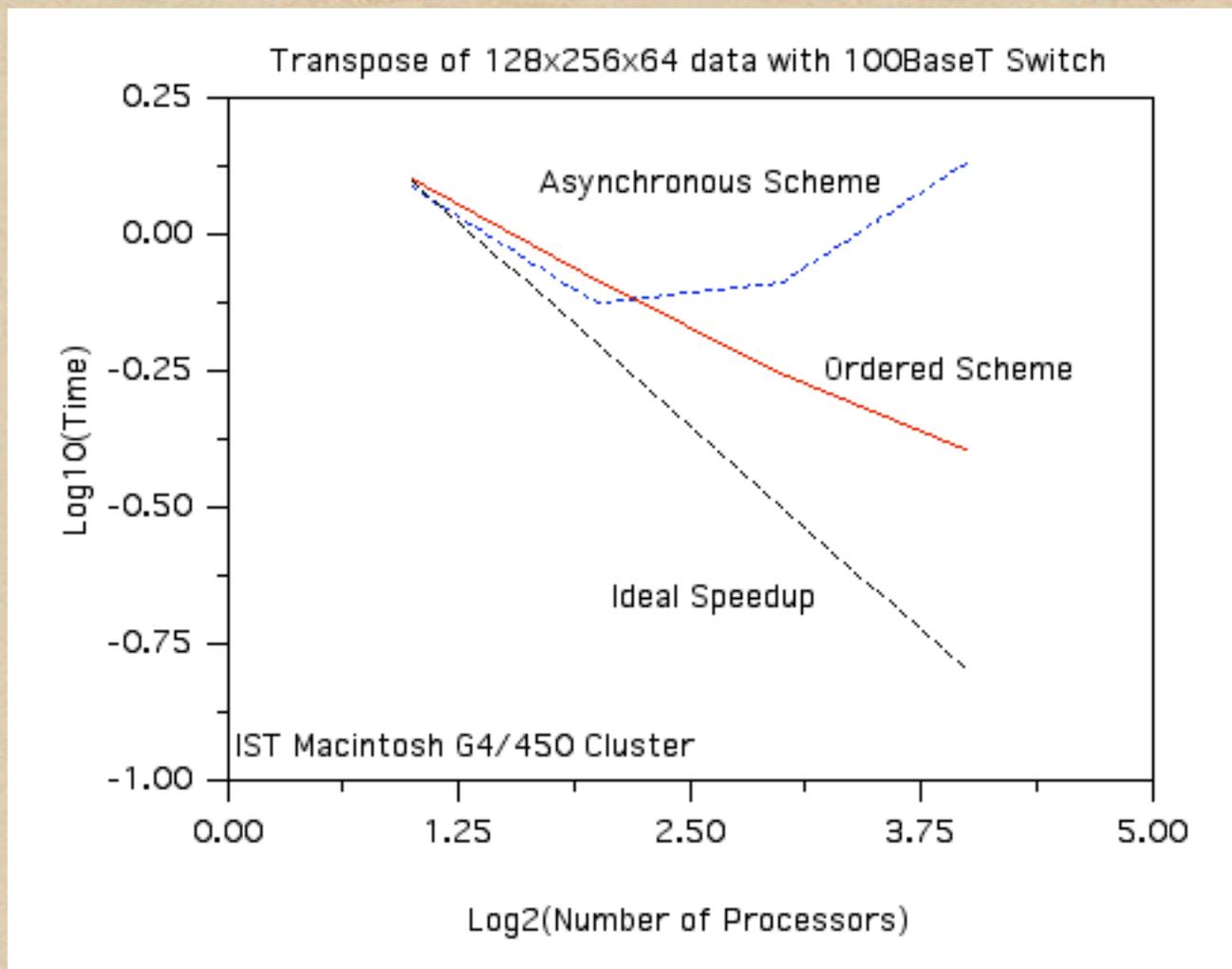
Transpose is very communication intensive, all-to-all

Collision avoidance important for some hardware

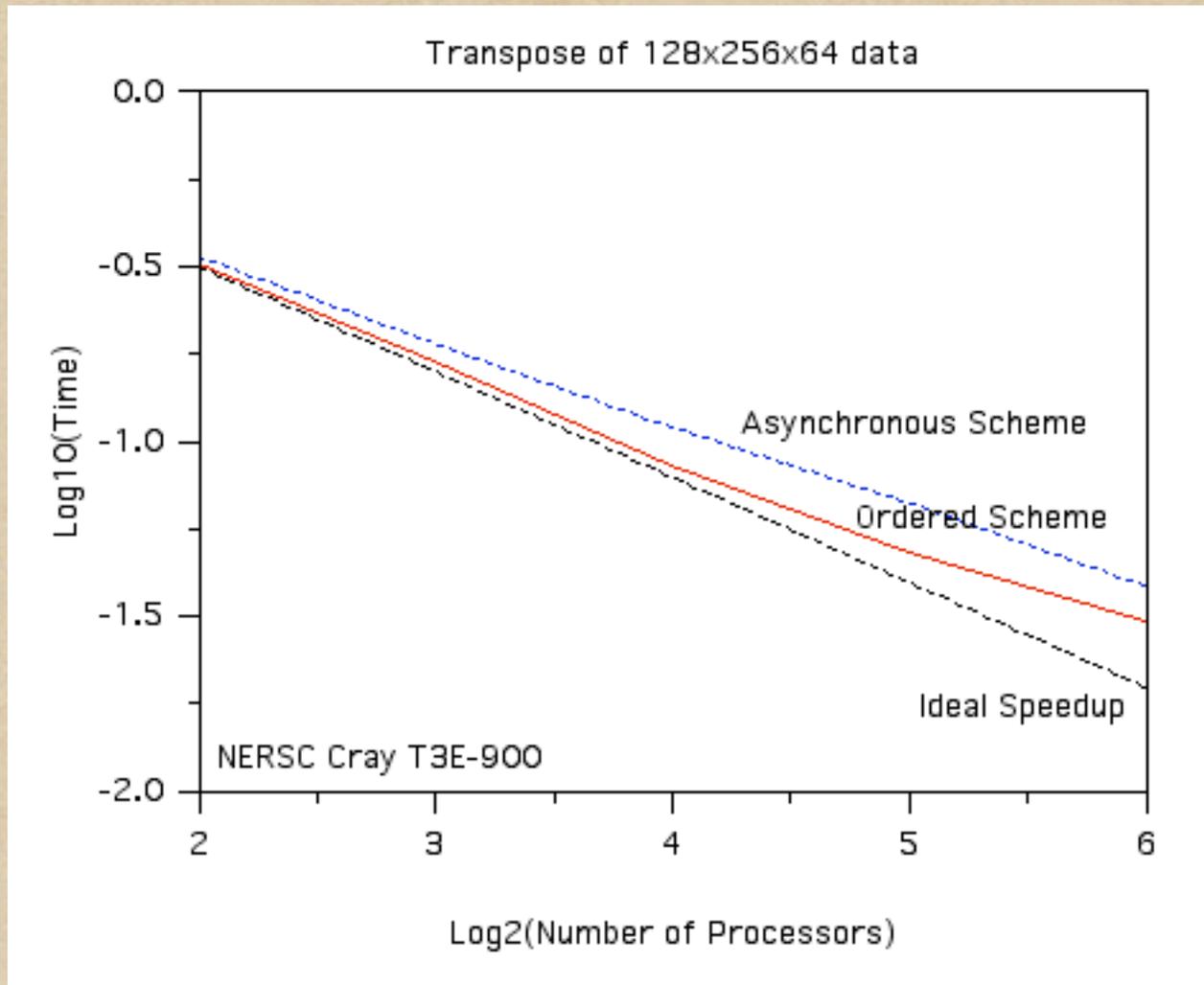


**Controlled Data Transpose**  
**Full Duplex, No collisions**

At any given pass, each process sends and receives to unique processor. Pause at each pass before continuing.

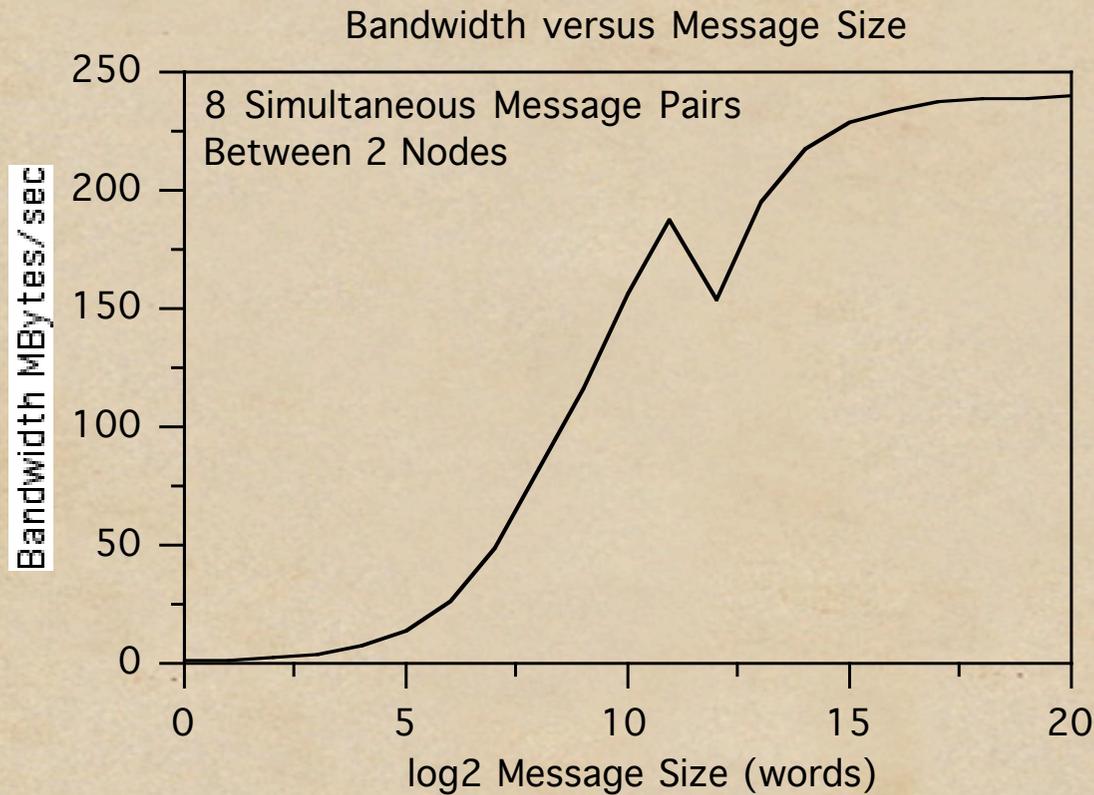


Ordered vs. Asynchronous Transpose on  
Commodity Switches



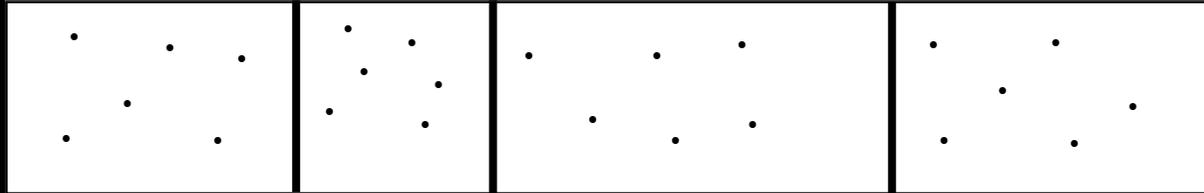
Ordered vs. Asynchronous Transpose on High Performance Computers

# Bandwidth Test on Atlas

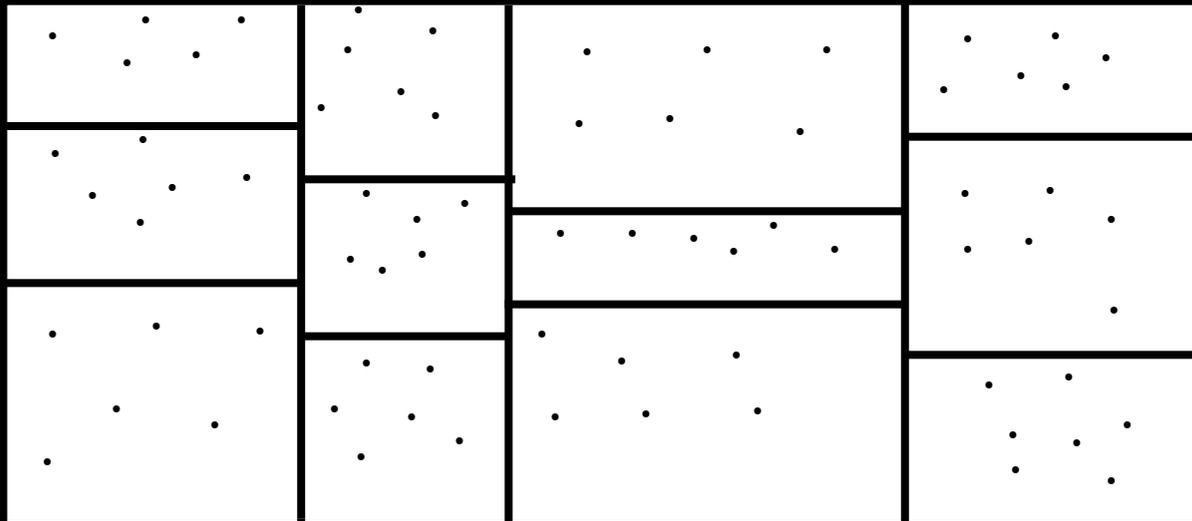


Message size in FFT:  
$$NX * NY * NZ / nproc^2$$

Measured Bandwidth for one message pair between physical nodes while multiple messages are being sent. Measured latency time is 8 microseconds. Half maximum is achieved for messages of about 512 words (2 KBytes). Note glitch at 4096 words (16 KBytes).



**1D Domain decomposition**

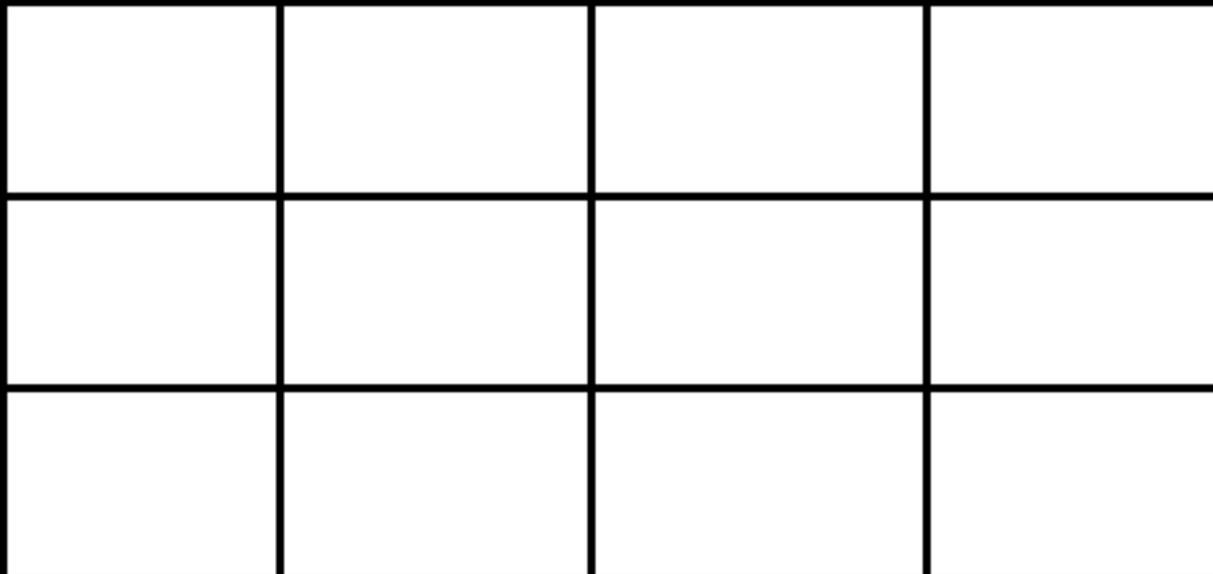


**2D Domain decomposition**

**Each partition has equal number of particles**



1D Domain decomposition



2D Domain decomposition

Each partition has equal number of grids

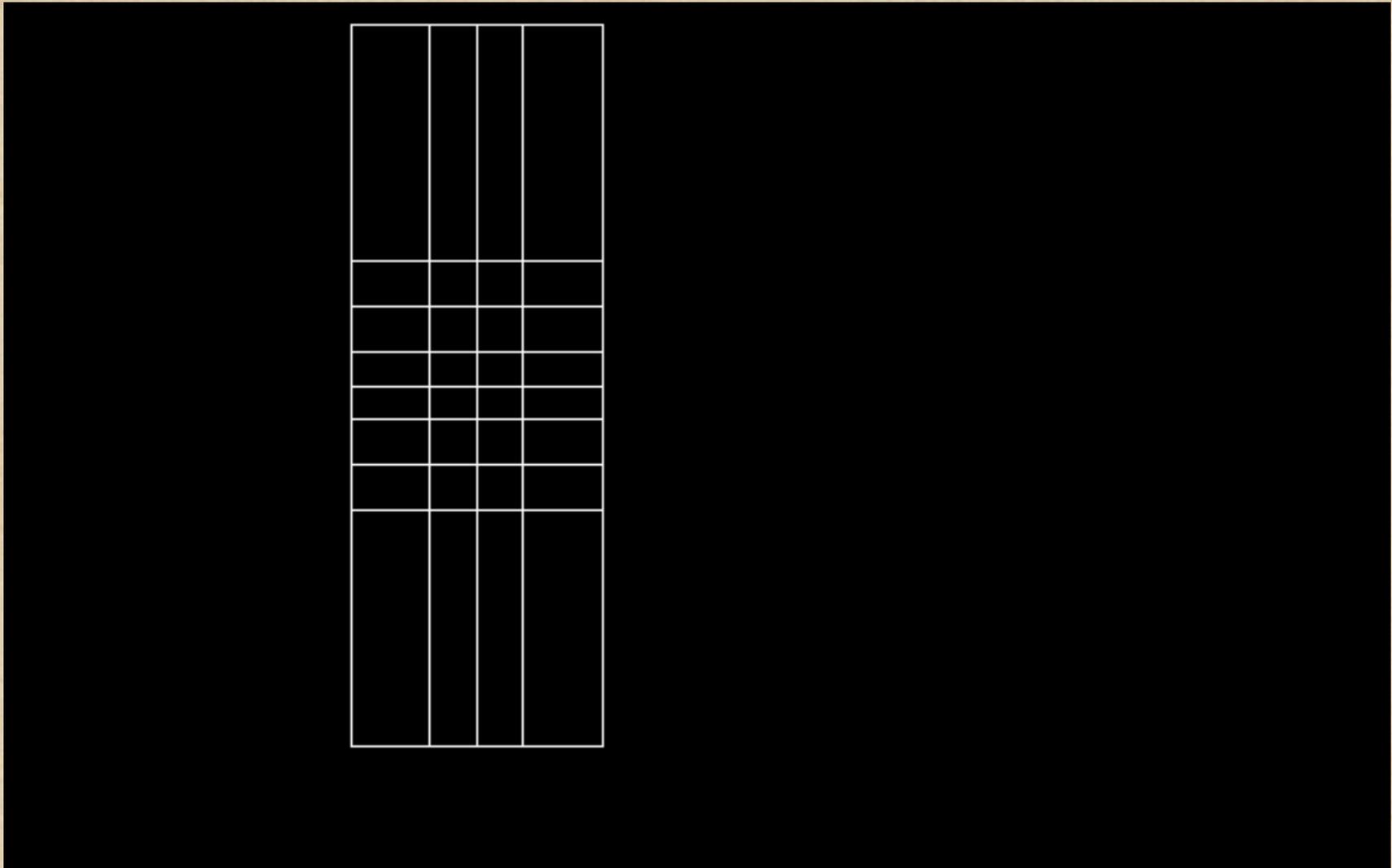
# Partition Manager

Moves data between non-uniform and uniform partitions

Fields which do not belong, passed to appropriate neighbor, then passed again if necessary. Remembers how many passes are required.

Finds new partitions based on number of particles per cell, accurate to nearest cell. Can also find new partitions from known distribution.

Partition manager is called every time step, since fields needed by particles are in non-uniform partitions, but FFT based solvers require uniform partitions.



Dynamic load balancing: changing partitions

# Future Architectures

Will multicore processors change our strategy?

