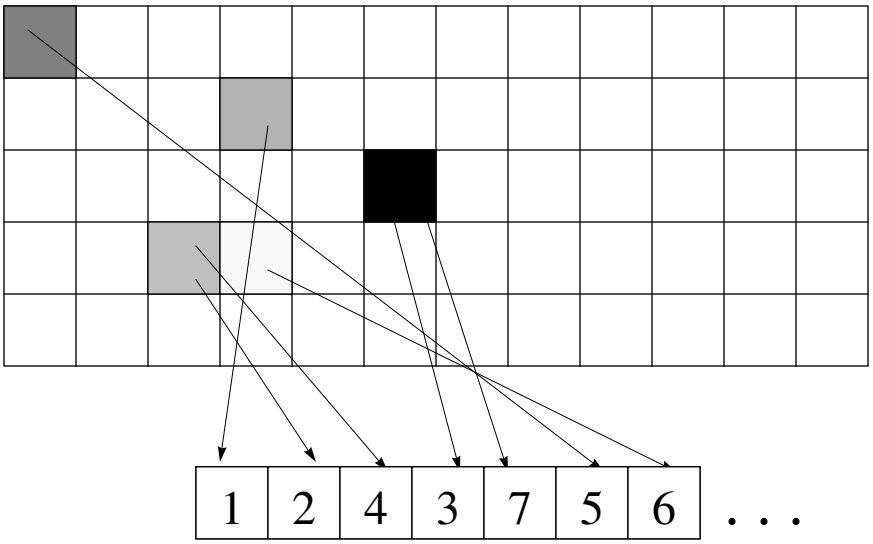
Explicit Message-Passing Implementation

Straight forward In-processor Cell method, because of the size of 2D gridded array is small --- Embarrassingly Parallel Approach.

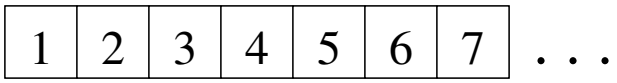
3. Initial Experimental Performance Data on CM5 and Intel iPSC/860

2) Particle Sorting:

2D Virtual Processor Array (Gridded Velocity)



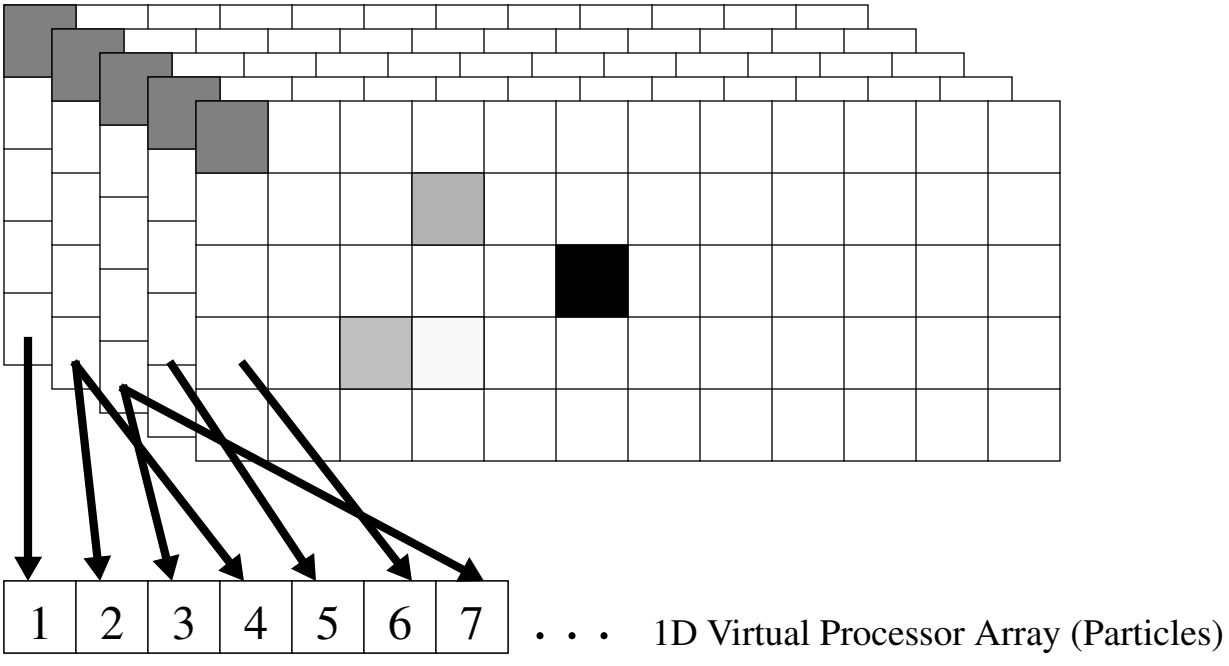
Sorted 1D Virtual Processor Array (Particles)



Unsorted 1D Virtual Processor Array (Particles)

3) In-Processor Cells:

A 2D Serial Array Copy on Each Physical Processor (Gridded Velocity)



1D Virtual Processor Array (Particles)

2. Major Computational Issues in Parallel Implementation

1) Data-parallel vs. Message-passing

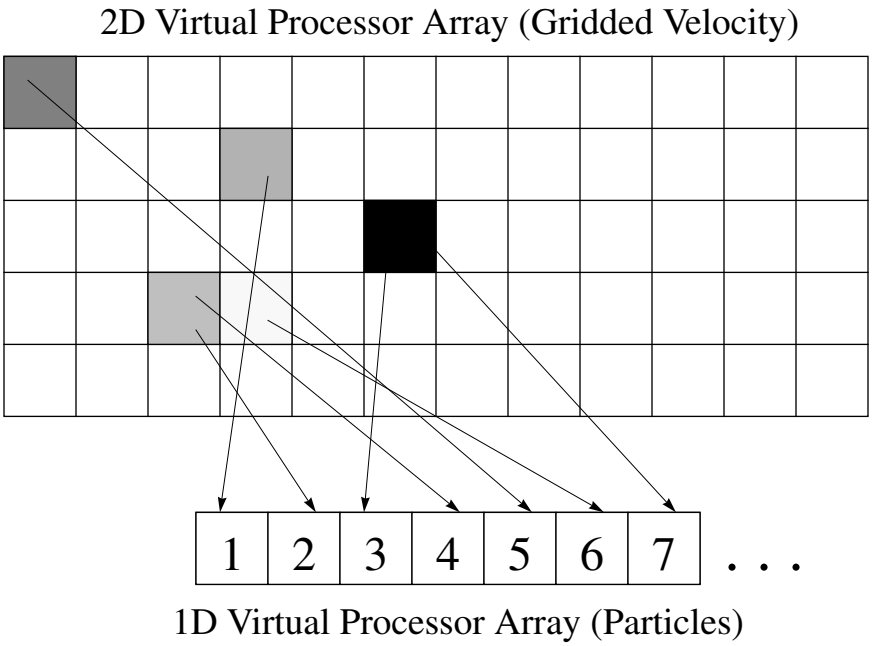
2) In *interpolation 3*, data-parallel algorithms for "Particle-in-Cell" interpolation

3) I/O --- Potential bottleneck for large datasets

2. Current Approaches to the Particle-in-Cell Interpolation

Data Parallel Implementation

1) General Gathering Operation from 2D Array to 1D:



Data Structure:

3D grid data --- temperature, velocity for large(in days) timeset
2D grid data --- velocity of the constant surface for large time frame and small # (in hours/minutes) time frame.
1D parcel velocity --- parcel velocity in lon.-lat. direction
1D parcel locations --- parcel location in lon.-lat. direction at past, current, and future timesteps.
1D pressure --- 18 standard pressure levels.

3D grid size: 72x91x18
2D grid size: 72x91
1D particle set size: 1000 - 100,000
1D pressure set size: 18

Data dependancy:

Interpolation 1:
3D grid data + 1D pressure data --> 2D grid data.

Interpolation 2:
2D grid data --> 2D grid data, linear.

Interpolation 3:
2D grid data + 1D particle location --> 1D particle velocity.

Leapfrog or Matsuno time integration:
1D particle location + 1D parcel velocity at current (and past) timestep --> 1D particle location at next future timestep.

1. Problem Structure

Control Structure:

Do **i** = 1, # of timesets

I/O of temperature, velocity for current two timesets # (3D grid data)

Interpolation 1: interpolate 3D grid data (temperature, pressure, and velocity) to 2D constant theta surface (velocity). (3D --> 2D grid interpolation)

Do **j** = 1, # of timesteps

Interpolation 2: interpolate linear grid velocity at large timeframe to the desired smaller timeframe. (2D grid in days --> 2D grid in hours/minutes interpolation)

Interpolation 3: interpolate grid velocity to parcel locations. (2D grid velocity --> 1D particle velocity)

Leapfrog or Matsuno time integration

Data update for next timestep

End do **j**

Data update for next timeset

End do **i**

Four-Dimensional Data Assimilation (NASA HPCC):

Parallel Trajectory Model

Gang Cheng,[*] Lynn Sparling,[**] Kim Mills,[*] and Peter Lyster[***]

[*]Northeast Parallel Architectures Center, Syracuse University
[**]NASA/Goddard Space Flight Center
[***]Jet Propulsion Laboratory

❑ Objectives

- ❍ Evaluate Computational Models and Hardware
- ❍ Define Operational MPP System for NASA's Data-Assimilation Mission

❑ Lagrangian Trajectory Model

- ❍ Data-parallel CMFortran vs. Message-passing CMMD
- ❍ Message-passing on CM 5 and iPSC/860

❑ Data-parallel Particle-in-Cell Methods Used

- ❍ General Gathering
- ❍ Particle Sorting
- ❍ In-processor Cell Method

❑ Parallel I/O Compared

- ❍ CM5 DataVault
- ❍ Intel iPSC/860 CFS