IBM TJ Watson Research Center

Visual Simulation of Fluid Dynamics on the Cell

Theodore Kim
Postdoctoral Fellow

6/19/07

© 2007 IBM Corporation

Visual Simulation

- **Accuracy is necessary**
- **Speed and stability are *required*.**
- **Unconditional stability – the user cannot break it**
- **Imagine if the Photoshop blur filter blew up …**

© 2007 IBM Corporation

Stable Fluids

- **Jos Stam, SIGGRAPH 1999**
- **An explosion of papers since then**
- **Conferences now have (sometimes multiple) ‘Fluids’ sessions**
- **Most FX-heavy movies use this**
  - Pirates of the Caribbean
  - Terminator 3
  - Star Wars 1-3
  - etc … (videos)

© 2007 IBM Corporation

Why On Cell?

- **Draws on Computational Fluid Dynamics (CFD) techniques**
  - Chorin projection
  - Semi-Lagrangian advection
- **3rd party implementations available**
  - Stam has posted CPU code
  - Nvidia has posted GPU code
- **Cell code designed for comparisons**

© 2007 IBM Corporation

Navier-Stokes Equations

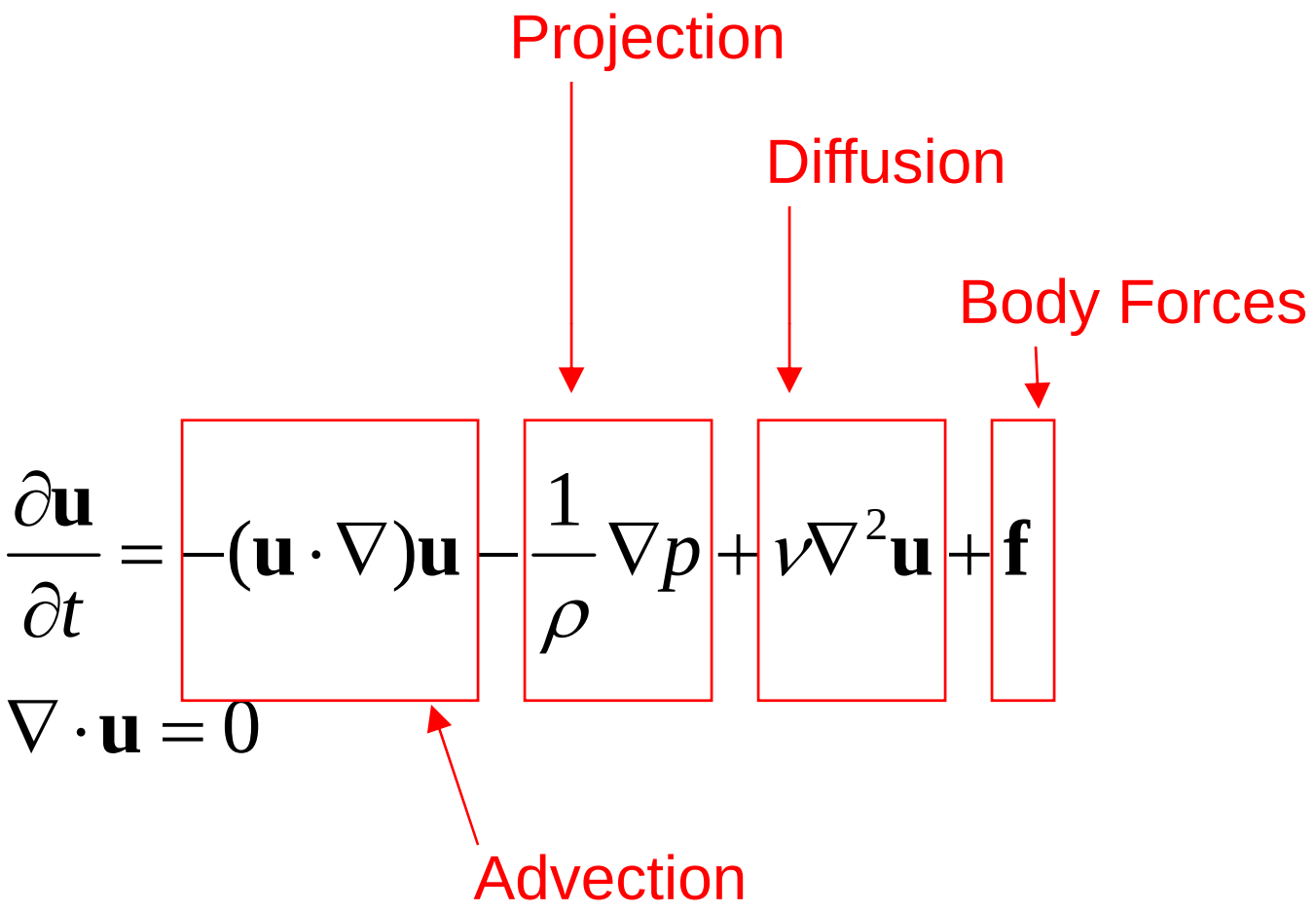
$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla)\mathbf{u} - \frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{f}$$

$$\nabla \cdot \mathbf{u} = 0$$

- **u = velocity (x,y,z)**
- **p = pressure**
- **t = time**
- **f = body forces (gravity, buoyancy, etc.)**

© 2007 IBM Corporation

Navier-Stokes Equations



© 2007 IBM Corporation

Regular Grid Simulation

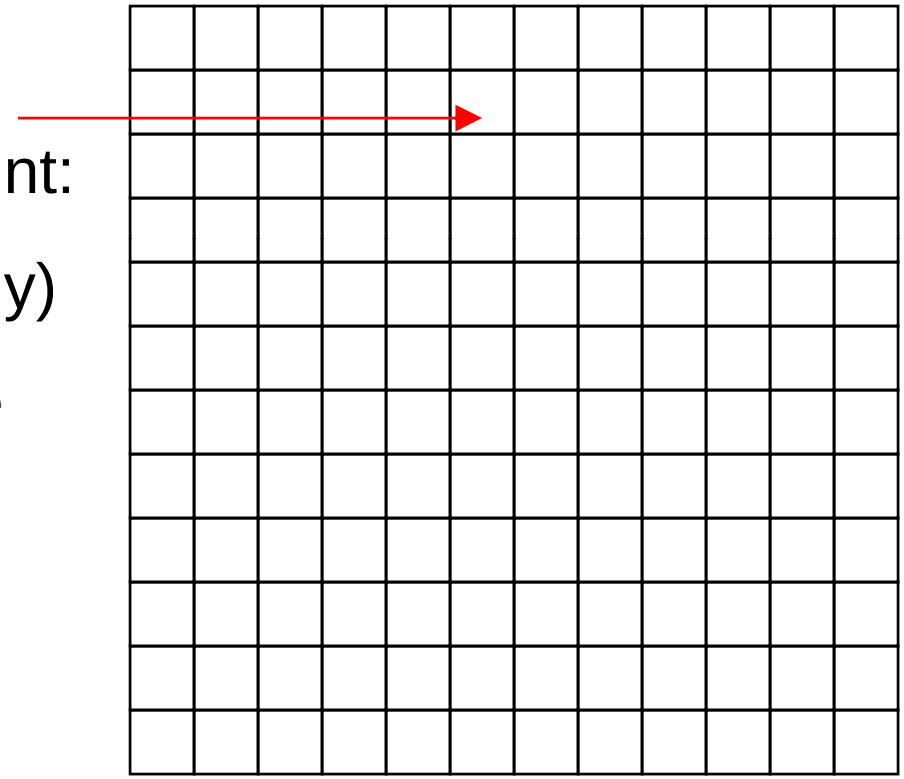
Per grid point:

Velocity (x,y)

Pressure

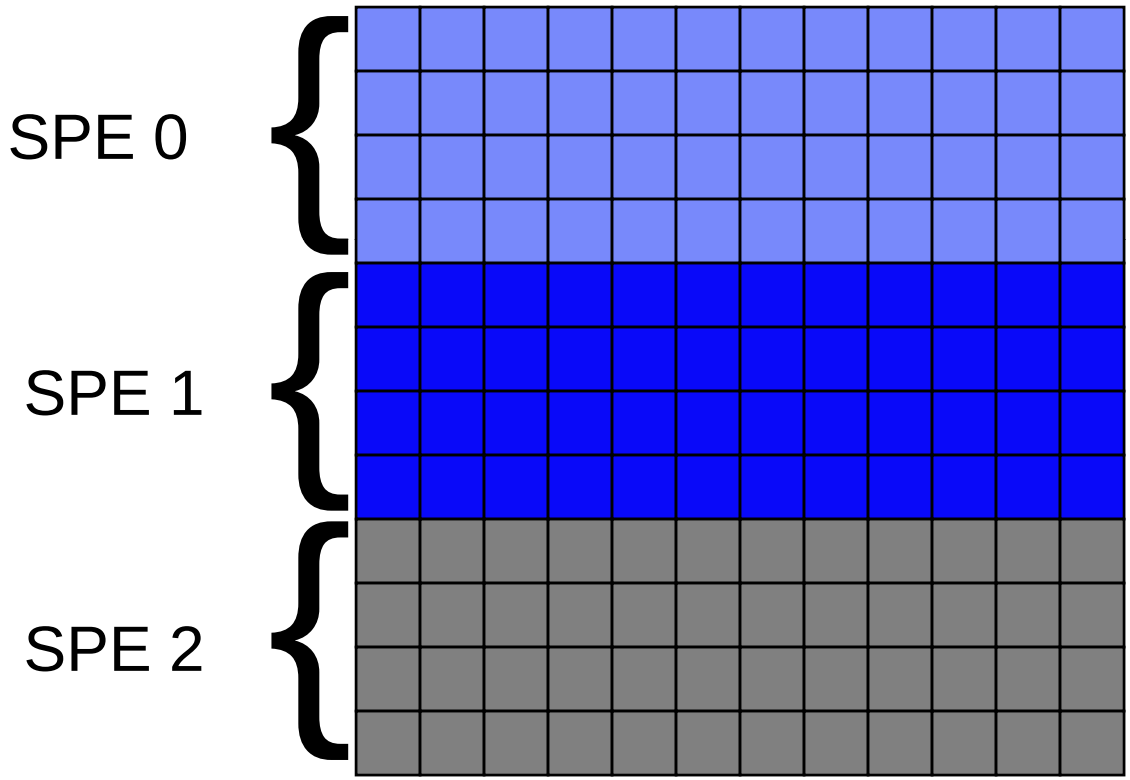
Density

Force



© 2007 IBM Corporation

Regular Grid Simulation



© 2007 IBM Corporation

Some Notation

$\mathbf{u}^{t}$ **- velocity at current timestep**

$\mathbf{u}^{t+1}$ **- velocity at next timestep**

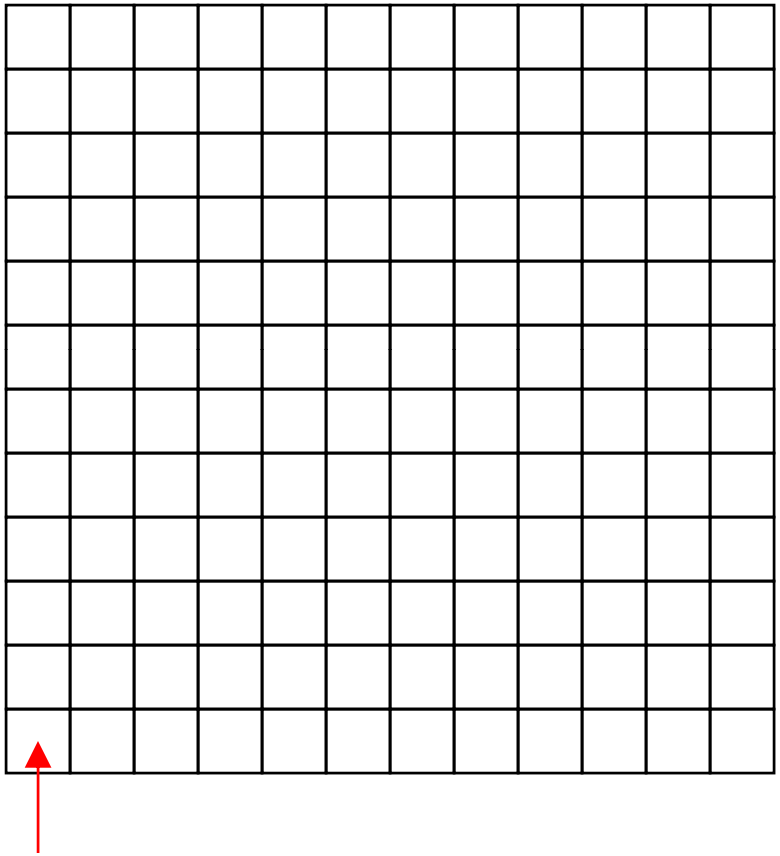
$\mathbf{u}'$ **- velocity after force**

$\mathbf{u}''$ **- velocity after diffusion**

$\mathbf{u}'''$ **- velocity after projection**

© 2007 IBM Corporation

Force Computation



For each grid point $\mathbf{u}' = \mathbf{u}^t + dt \times \mathbf{f}$

© 2007 IBM Corporation

Diffusion

$$\mathbf{u}''_{2,2} = \mathbf{u}'_{2,2} + \begin{array}{|c|c|c|}\hline \mathbf{u}'_{1,3} & \mathbf{u}'_{2,3} & \mathbf{u}'_{3,3} \\ \hline \mathbf{u}'_{1,2} & \mathbf{u}'_{2,2} & \mathbf{u}'_{3,2} \\ \hline \mathbf{u}'_{1,1} & \mathbf{u}'_{2,1} & \mathbf{u}'_{3,1} \\ \hline \end{array} \otimes \frac{dt}{\Delta x^2} \begin{array}{|c|c|c|}\hline & 1 & \\ \hline 1 & -4 & 1 \\ \hline & 1 & \\ \hline \end{array}$$

$$= \mathbf{u}'_{2,2} + \frac{dt}{\Delta x^2}\left(-4\mathbf{u}'_{2,2} + \mathbf{u}'_{1,2} + \mathbf{u}'_{3,2} + \mathbf{u}'_{2,3} + \mathbf{u}'_{2,1}\right)$$

© 2007 IBM Corporation

Diffusion

$$\mathbf{u}''_{2,2} = \mathbf{u}'_{2,2} + \begin{array}{|c|c|c|}\hline \mathbf{u}''_{1,3} & \mathbf{u}''_{2,3} & \mathbf{u}''_{3,3} \\ \hline \mathbf{u}''_{1,2} & \mathbf{u}''_{2,2} & \mathbf{u}''_{3,2} \\ \hline \mathbf{u}''_{1,1} & \mathbf{u}''_{2,1} & \mathbf{u}''_{3,1} \\ \hline \end{array} \otimes \frac{dt}{\Delta x^2} \begin{array}{|c|c|c|}\hline & 1 & \\ \hline 1 & -4 & 1 \\ \hline & 1 & \\ \hline \end{array}$$

$$= \mathbf{u}'_{2,2} + \frac{dt}{\Delta x^2}\left(-4\mathbf{u}''_{2,2} + \mathbf{u}''_{1,2} + \mathbf{u}''_{3,2} + \mathbf{u}''_{2,3} + \mathbf{u}''_{2,1}\right)$$

© 2007 IBM Corporation

Diffusion

- **Lots of possible solvers –**
  - Jacobi
  - Gauss-Seidel
  - Successive over-relaxation (SOR)
  - FFT
  - Conjugate gradient
  - Multigrid

© 2007 IBM Corporation

Gauss-Seidel Solver

```
for (int z = 0; z < 20; z++)
  for (int j = 1; j < N-1; j++)
    for (int i = 1; i < N-1; i++)
      u[i][j] = (uOld[i][j] + (u[i-1][j] + u[i+1][j] +
                               u[i][j-1] + u[i][j+1])*c0)*c1;
```


© 2007 IBM Corporation

Gauss-Seidel Solver

```
for (int z = 0; z < 20; z++)
  for (int j = 1; j < N-1; j++)
    for (int i = 1; i < N-1; i++)
      u[i][j] = (uOld[i][j] + (u[i-1][j] + u[i+1][j] +
                               u[i][j-1] + u[i][j+1])*c0)*c1;
```

- 5 flops
- 1 store
- 6 loads

© 2007 IBM Corporation

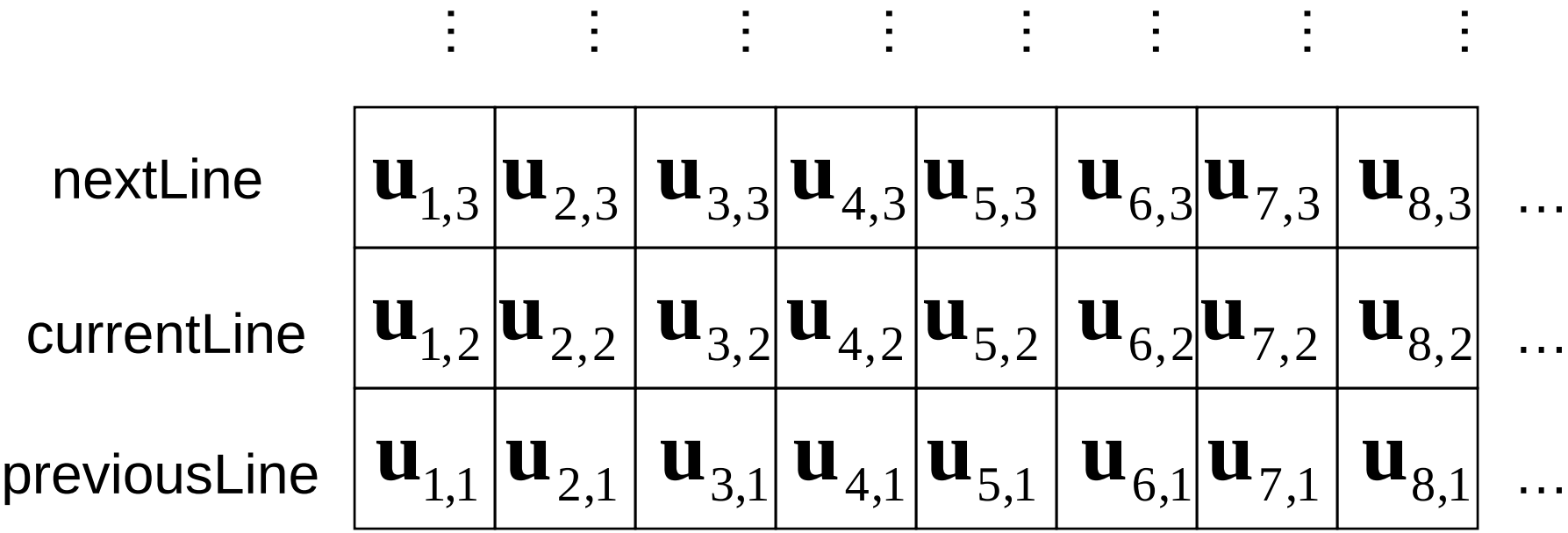
Gauss-Seidel Solver

- **On 512 x 512 grid,**
  - 204.8 GFlops / (512 x 512 x 20 iterations x 5 flops) =
    7812.5 frames per second
  - 25.6 GB/s / (512 x 512 x 20 iterations x 7 load-stores x
    4 bytes per float) =
    174.4 frames per second
  - Run 4 times (x velocity, y velocity, pressure, density)
    43.6 frames per second

© 2007 IBM Corporation

Gauss-Seidel Solver

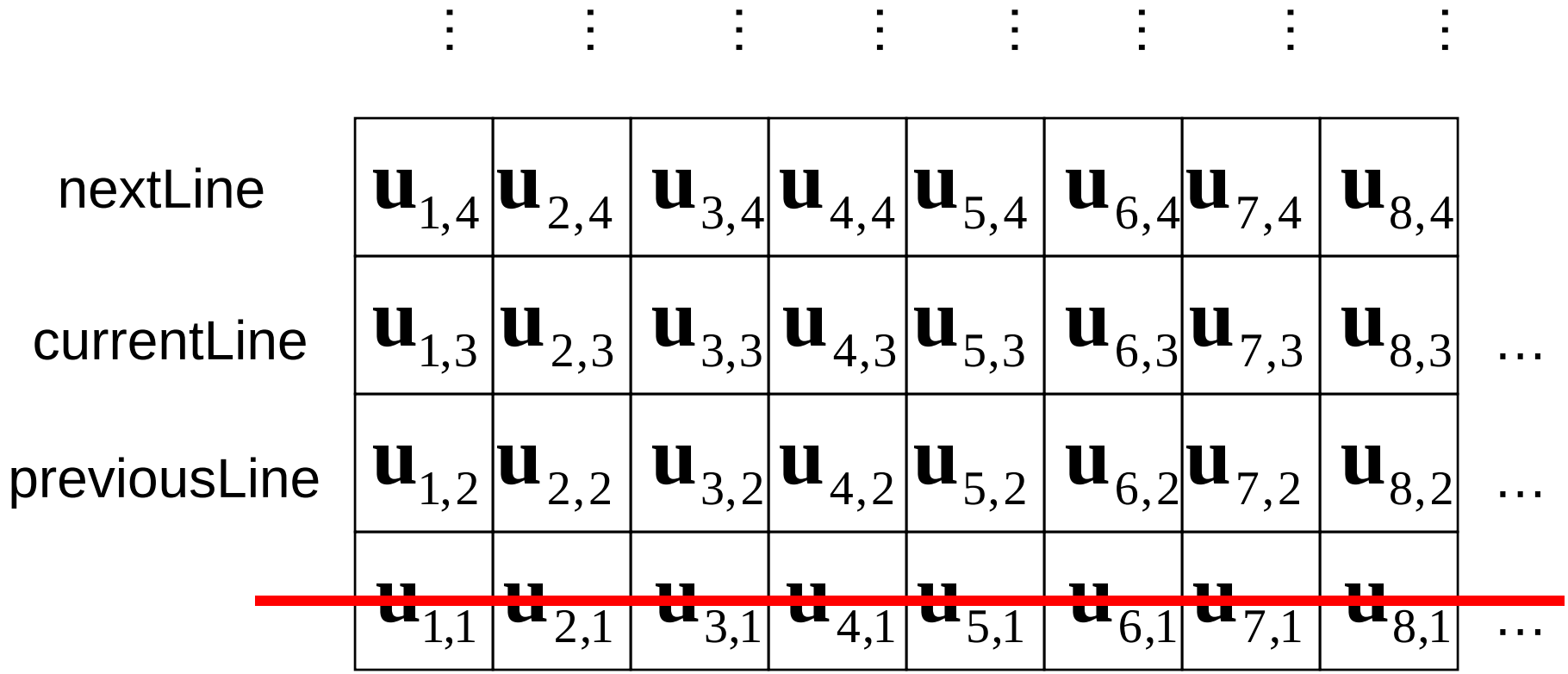
```
u[i][j] = (uOld[i][j] + (u[i-1][j] + u[i+1][j] +
                         u[i][j-1] + u[i][j+1])*c0)*c1;
```



© 2007 IBM Corporation

Gauss-Seidel Solver

```
u[i][j] = (uOld[i][j] + (u[i-1][j] + u[i+1][j] +
                         u[i][j-1] + u[i][j+1])*c0)*c1;
```



© 2007 IBM Corporation

Gauss-Seidel Solver

- **On 512 x 512 grid,**
  - 25.6 GB/s / (512 x 512 x 20 iterations x 3 load-stores x 4 bytes per float) =

    406.9 frames per second

  - Run 4 times (x velocity, y velocity, pressure, density)

    101.7 frames per second

© 2007 IBM Corporation

Higher Order Stencils

$$\mathbf{u}''_{2,2} = \mathbf{u}'_{2,2} + \begin{array}{|c|c|c|}\hline \mathbf{u}''_{1,3} & \mathbf{u}''_{2,3} & \mathbf{u}''_{3,3} \\ \hline \mathbf{u}''_{1,2} & \mathbf{u}''_{2,2} & \mathbf{u}''_{3,2} \\ \hline \mathbf{u}''_{1,1} & \mathbf{u}''_{2,1} & \mathbf{u}''_{3,1} \\ \hline \end{array} \otimes \frac{dt}{\Delta x^2} \begin{array}{|c|c|c|}\hline & 1 & \\ \hline 1 & -4 & 1 \\ \hline & 1 & \\ \hline \end{array}$$

$$= \mathbf{u}'_{2,2} + \frac{dt}{\Delta x^2}\left(-4\mathbf{u}''_{2,2} + \mathbf{u}''_{1,2} + \mathbf{u}''_{3,2} + \mathbf{u}''_{2,3} + \mathbf{u}''_{2,1}\right)$$

© 2007 IBM Corporation

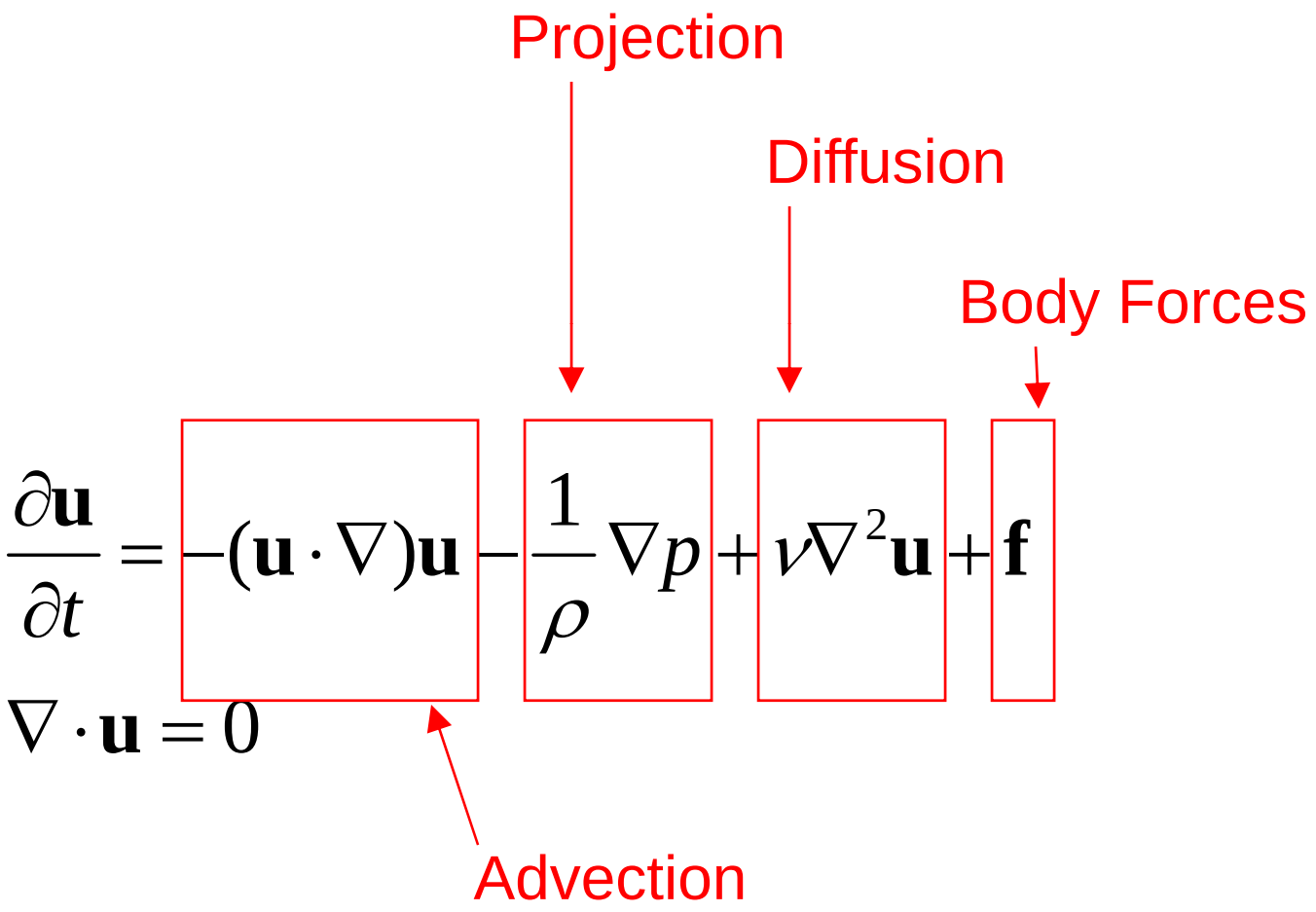
Higher Order Stencils

$$\mathbf{u}''_{2,2} = \mathbf{u}'_{2,2} + \begin{bmatrix} \mathbf{u}''_{1,3} & \mathbf{u}''_{2,3} & \mathbf{u}''_{3,3} \\ \mathbf{u}''_{1,2} & \mathbf{u}''_{2,2} & \mathbf{u}''_{3,2} \\ \mathbf{u}''_{1,1} & \mathbf{u}''_{2,1} & \mathbf{u}''_{3,1} \end{bmatrix} \otimes \frac{dt}{\Delta x^2} \begin{bmatrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{bmatrix}$$

$$= \mathbf{u}'_{2,2} + \frac{dt}{\Delta x^2} \begin{pmatrix} -20\mathbf{u}''_{2,2} + 4\mathbf{u}''_{1,2} + 4\mathbf{u}''_{3,2} + 4\mathbf{u}''_{2,3} + 4\mathbf{u}''_{2,1} + \\ \mathbf{u}''_{1,1} + \mathbf{u}''_{3,1} + \mathbf{u}''_{1,3} + \mathbf{u}''_{3,3} \end{pmatrix}$$

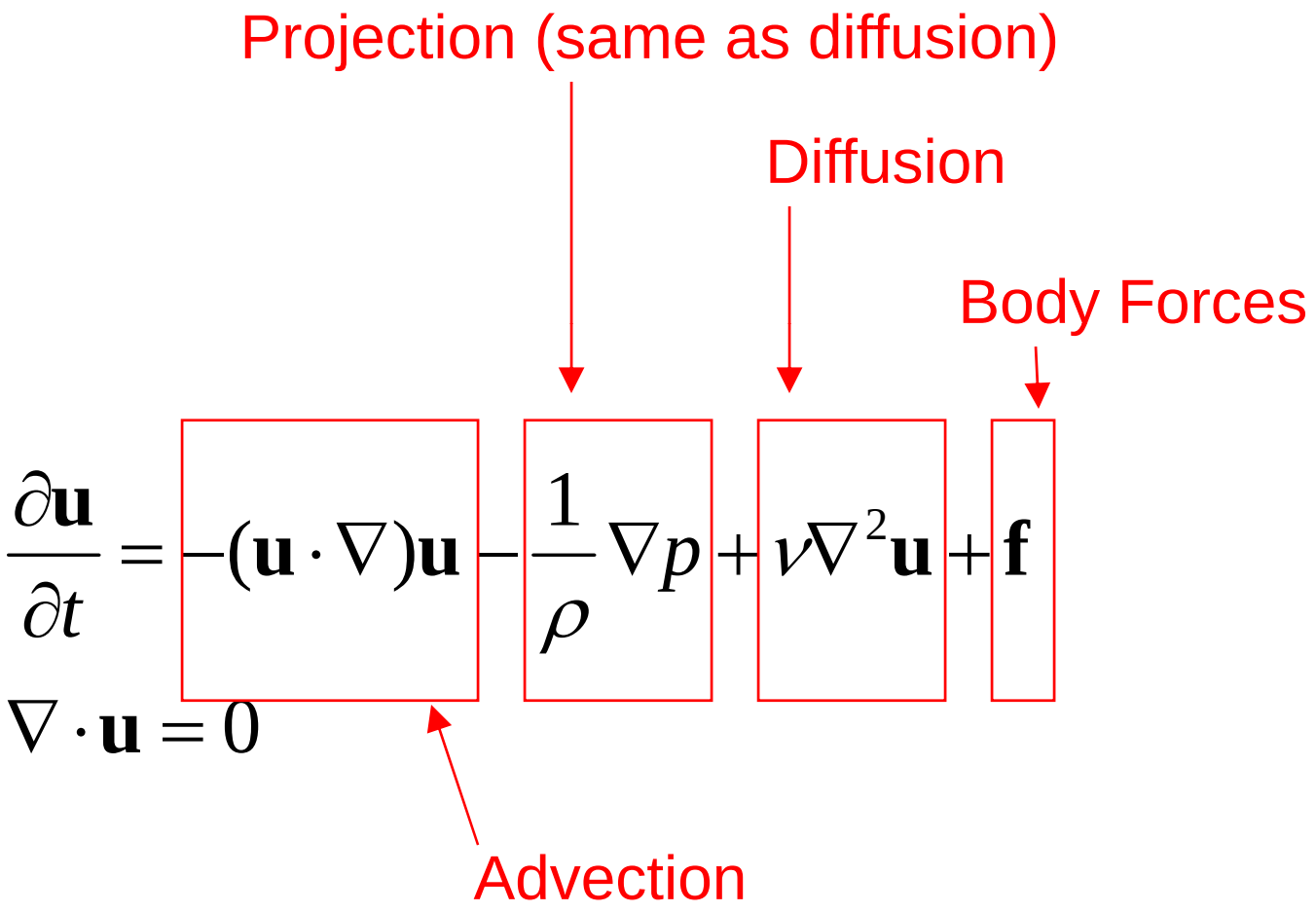
© 2007 IBM Corporation

Navier-Stokes Equations



© 2007 IBM Corporation

Navier-Stokes Equations



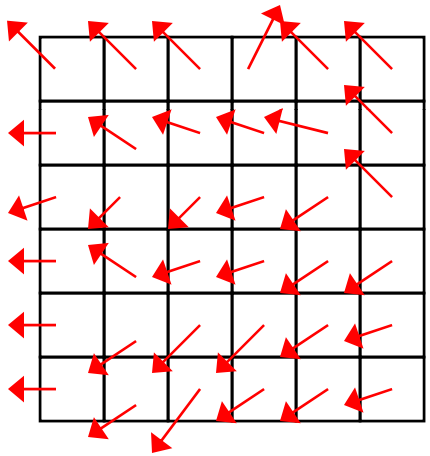
© 2007 IBM Corporation

Advection

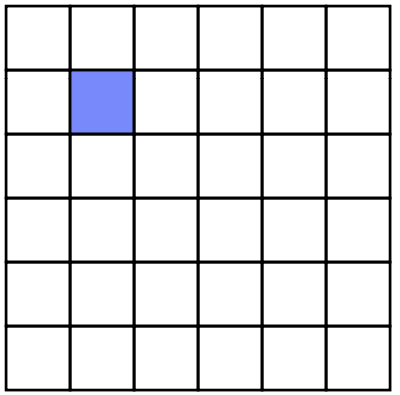
- **(show CPU version)**

© 2007 IBM Corporation

Advection



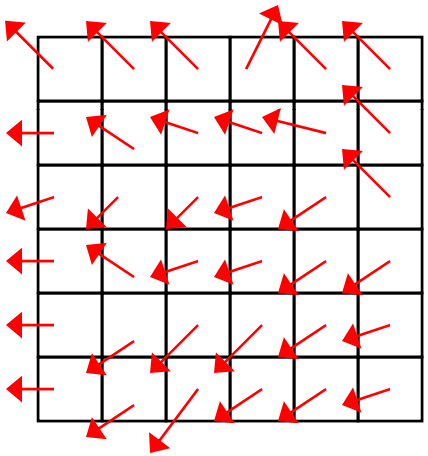
Velocity at time t



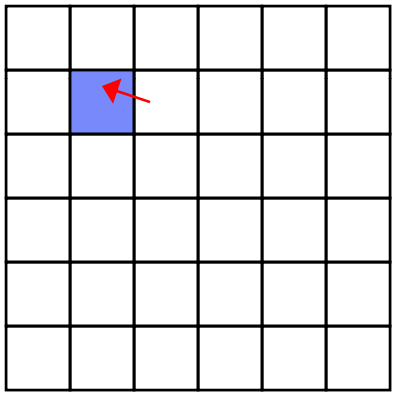
What stuff arrives
at time t+1?

© 2007 IBM Corporation

Advection – Forward Euler version



Velocity at time t

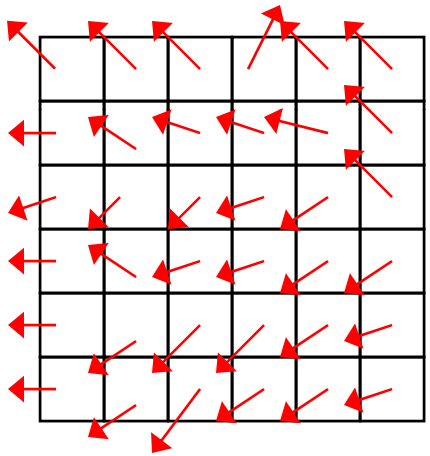


What stuff arrives at time t+1?

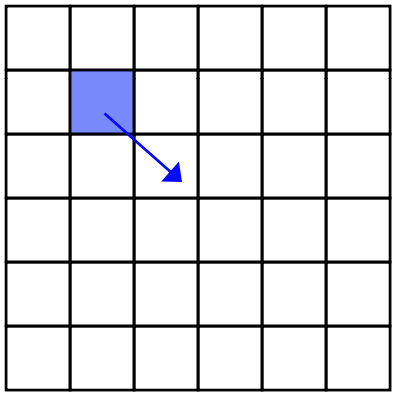
Who's pointing at me?

© 2007 IBM Corporation

Advection – Semi-Lagrangian version



Velocity at time t



What stuff arrives at time t+1?

Look backwards and interpolate

© 2007 IBM Corporation

Semi-Lagrangian Advection

- **On a 512x512 grid**
  - 204.8 GFlops / (512 x 512 x 16 flops) =

    48,828 frames per second

  - 25.6 GB/s / (512 x 512 x 7 load-stores x 4 bytes per float) =

    3487 frames per second

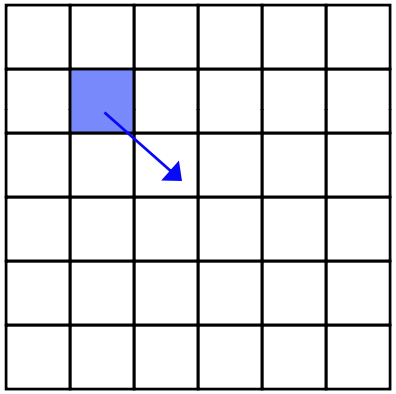
  - Run 3 times (x velocity, y velocity, density)

    1162 frames per second

© 2007 IBM Corporation

Semi-Lagrangian Advection

- **25.6 GB/s occurs with contiguous 16 KB DMAs**

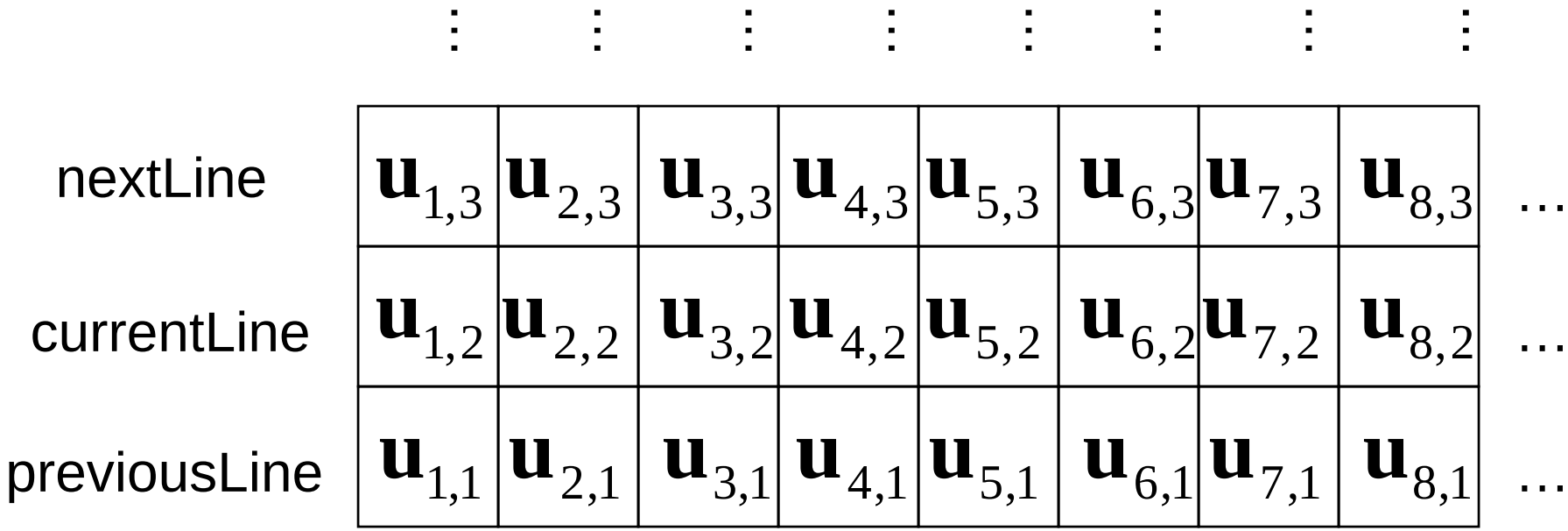


- **We have 512 x 512 x 3 *random, 32 byte* DMAs!**
- **Use software caching?**

© 2007 IBM Corporation

Semi-Lagrangian Advection

- **Most lookups are *relatively local***
- **Prefetch 3x3 neighborhood around each cell**



© 2007 IBM Corporation

Semi-Lagrangian Advection

Resolution	Without cache (GB/s)	With cache (GB/s)
64^2	8.01	15.91
128^2	7.98	16.45
256^2	7.98	16.54
512^2	8.18	16.5
1024^2	8.16	15.54

Average cache hit rate ~ 99%, pathological cases ~ 80%

© 2007 IBM Corporation

Solver Components

- **Force solver**
  - Maps well
- **Diffusion solver**
  - Maps well
- **Semi-Lagrangian solver**
  - Maps okay, with a little work
- **(videos)**

© 2007 IBM Corporation

3D case

- **Access patterns get hairier, LS exceeded quickly**
- **64^3 running at 40 Hz**
- **Volume rendering is the bottleneck**
- **Ongoing …**
- **(video)**

© 2007 IBM Corporation

Ongoing Work

- **Try out 16 bit floats**
- **Higher order methods**
  - SPEs are idle *75%* of the time!
  - Bigger diffusion stencils – How big is still free?
  - Back and forth error correction and compensation (BFECC) – Can we get it for free?
- **Better solvers**
  - Conjugate gradient
  - Multigrid

© 2007 IBM Corporation

- **asmVis …**
- **Questions?**

© 2007 IBM Corporation