

Digital Human: Simulation of the Heart and other Organs

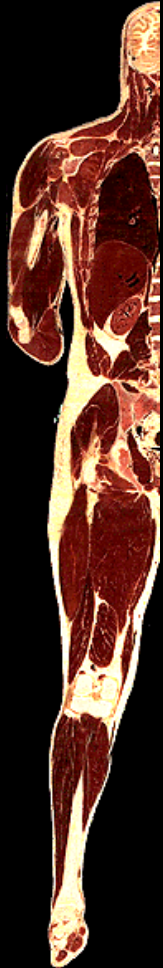
Kathy Yelick
EECS Department
U.C. Berkeley

The 20+ Year Vision

- Imagine a “digital body double”
 - 3D image-based medical record
 - Includes diagnostic, pathologic, and other information
- Used for:
 - Diagnosis
 - Less invasive surgery-by-robot
 - Experimental treatments
- Digital Human Effort
 - Lead by the Federation of American Scientists



Digital Human Today: Imaging



- The Visible Human Project
 - 18,000 digitized sections of the body
 - Male: 1mm sections, released in 1994
 - Female: .33mm sections, released in 1995
 - Goals
 - study of human anatomy
 - testing medical imaging algorithms
 - Current applications:
 - educational, diagnostic, treatment planning, virtual reality, artistic, mathematical and industrial
 - Used by > 1,400 licensees in 42 countries

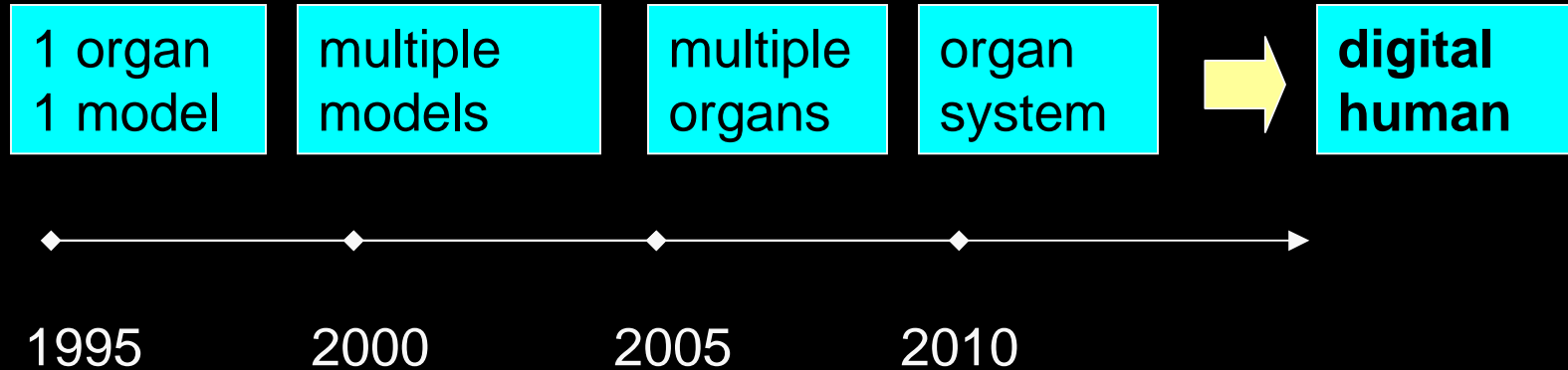


Image Source: www.madsci.org



THE VISIBLE HUMAN PROJECT®

Digital Human Roadmap



algorithms & mathematics

3D model construction

faster algorithms

coupled models

computer systems

scalable parallel implementations

faster computers

improved programmability

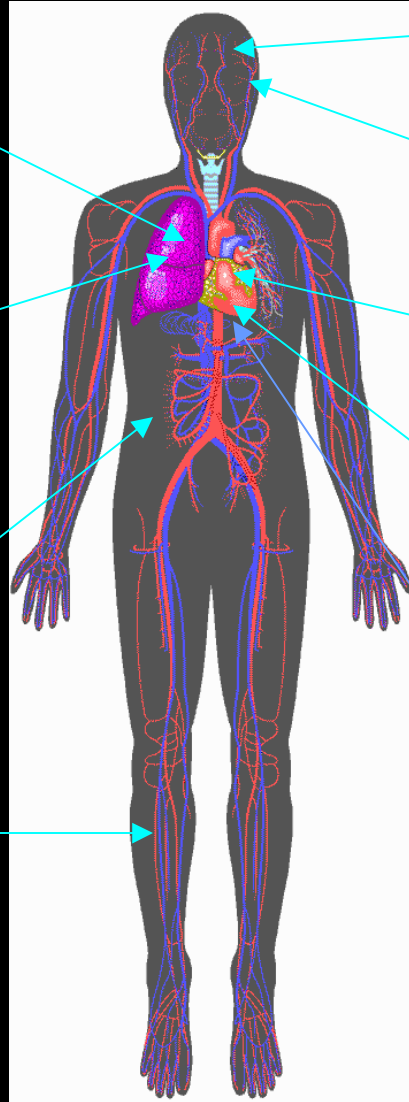
Organ Simulation

Lung transport
– Vanderbilt

Lung flow
– ORNL

Kidney mesh generation
– Dartmouth

Skeletal mesh generation



Brain
– Ellisman

Cochlea
– Caltech, UM, UCB

Cardiac flow
– NYU,...

Cardiac cells/muscles
– SDSC, Auckland, UW, Utah,

Electrocardiography
– Johns Hopkins,...

Just a few of the efforts at understanding and simulating parts of the human body

Immersed Boundaries within the Body

- Fluid flow within the body is one of the major challenges, e.g.,
 - Blood through the heart
 - Coagulation of platelets in clots
 - Effect of sound waves on the inner ear
 - Movement of bacteria
- A key problem is modeling an elastic structure immersed in a fluid
 - Irregular moving boundaries
 - Wide range of scales
 - Vary by structure, connectivity, viscosity, external forces, internally-generated forces, etc.

Heart Simulation

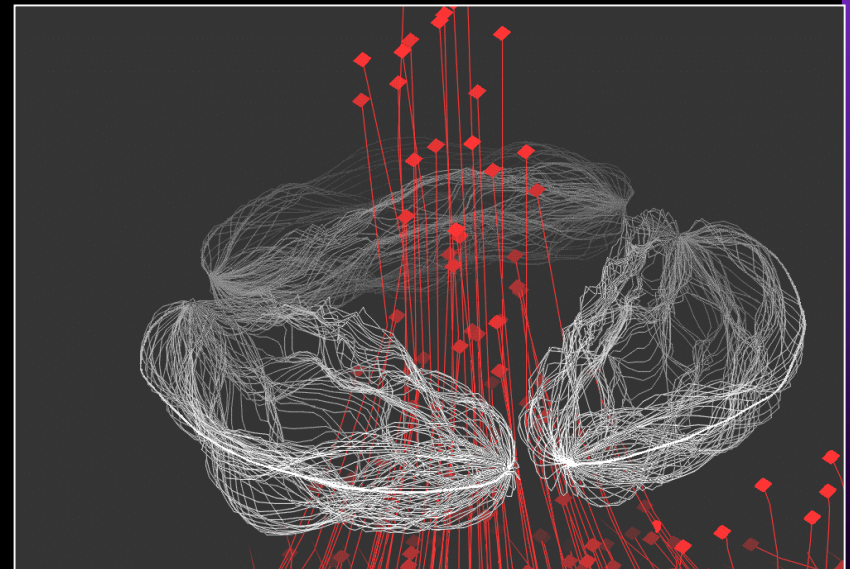
Developed by Peskin and McQueen at NYU

- Ran on vector and shared memory machines
- 100 CPU hours on a Cray C90
- Models blood flow in the heart
- Immersed boundaries are individual muscle fibers

– Rules for contraction, valves, etc. included

– Applications:

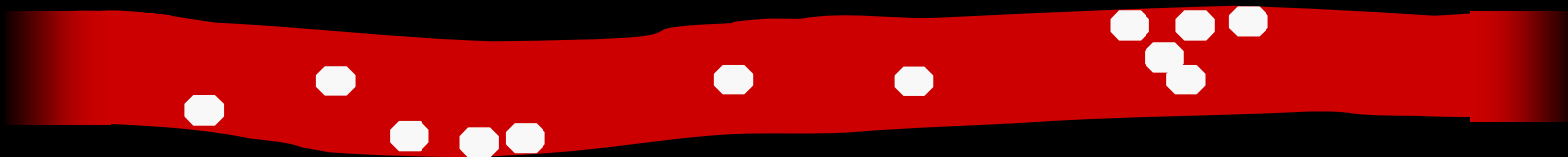
- Understanding structural abnormalities
- Evaluating artificial heart valves
- Eventually, artificial hearts



Source: www.psc.org

Platelet Coagulation

- Developed by Fogelson and Peskin
 - Simulation of blood clotting in 2D
 - Immersed boundaries are
 - Cell walls, represented by polygons
 - Artery walls
 - Rules added to simulate adhesion
 - For vector and shared memory machines
 - We did earlier work on this 2D problem in Split-C



Cochlea Simulation

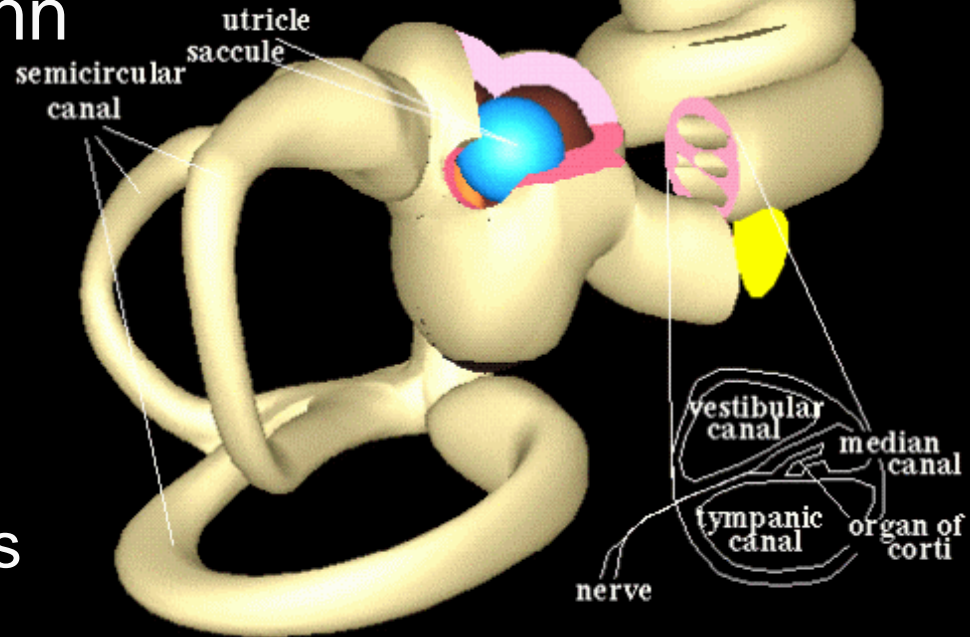
- Simulation by Givelberg and Bunn

- In OpenMP
- 18 hours on HP Superdome

- Embedded 2D structures are

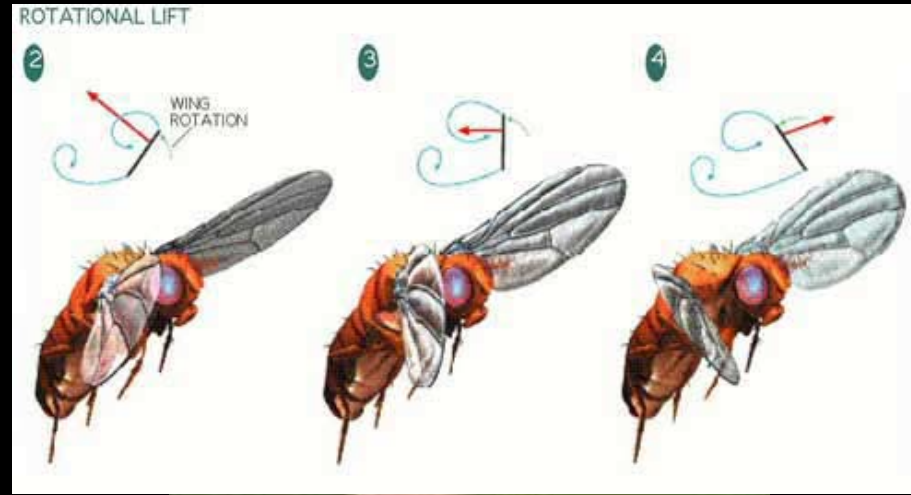
- Elastic membranes and shells
- Simulates fluid-structure interactions due to incoming sound waves
- Potential applications: design of cochlear implants

Structure of Inner Ear



Insect Flight Simulation

- Work by on insect flight
 - Wings are immersed 2D structure
- Under development
 - UW (Wang) and NYU (Miller)
- Applications to
 - Insect robot design



Source: Dickenson, UCB

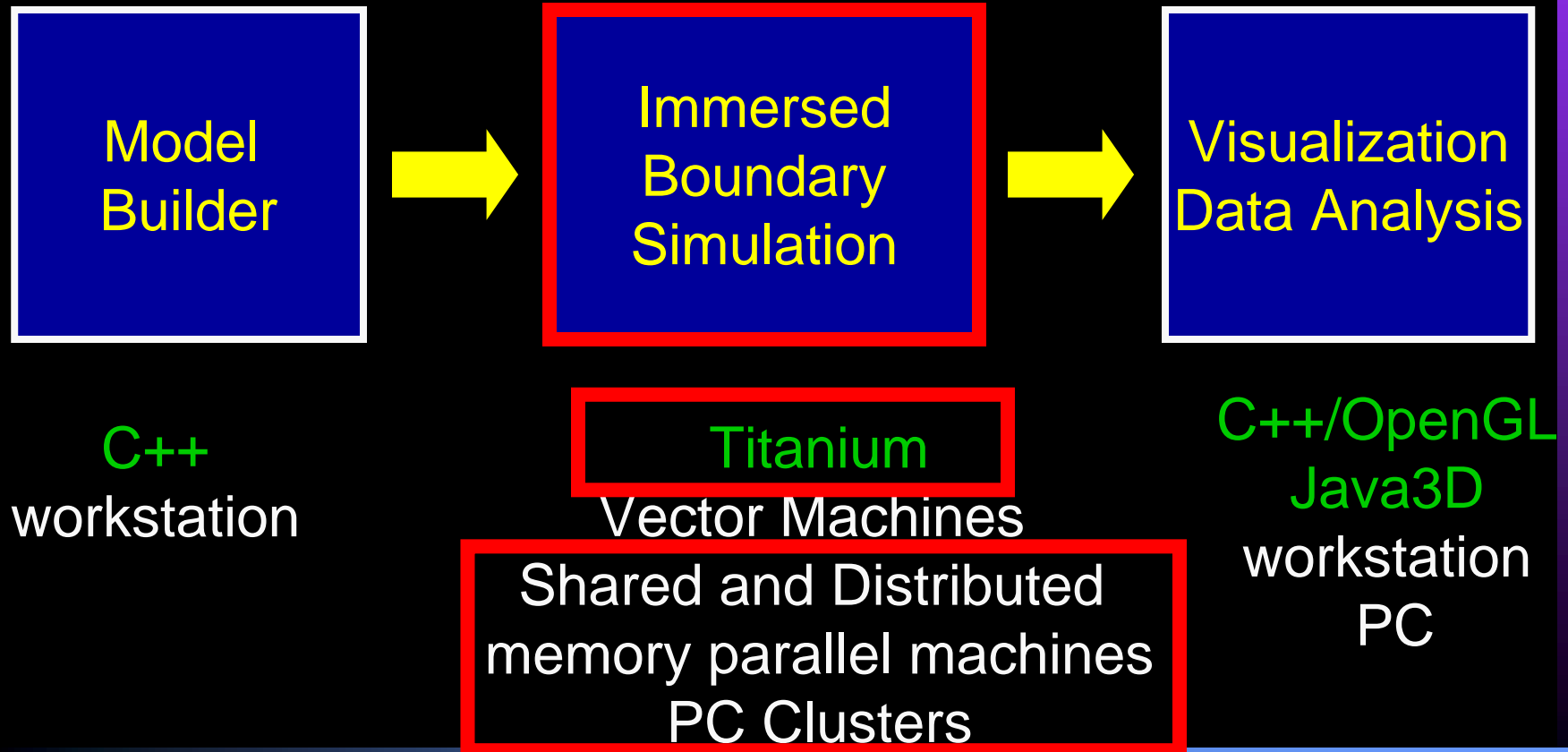
Small Animal Motion

- Simulation of small animal motion by Fauci, Dillon and other
 - Swimming of eels, sperm, and bacteria
 - Crawling motion of amoeba
 - Biofilms, such as plaque, with multiple micro-organisms
- Applications at a smaller scale
 - Molecular motors, fluctuations in DNA
 - Thermal properties may become important
 - Brownian motion extension by Kramer, RPI

Other Applications

- The immersed boundary method has also been used, or is being applied to
 - Flags and parachutes
 - Flagella
 - Embryo growth
 - Valveless pumping (E. Jung)
 - Paper making
 - Whirling instability of an elastic filament (S. Lim)
 - Flow in collapsible tubes (M. Rozar)
 - Flapping of a flexible filament in a flowing soap film (L. Zhu)
 - Deformation of red blood cells in shear flow (Eggleon and Popel)

Immersed Boundary Simulation Framework



Building 3D Models from Images

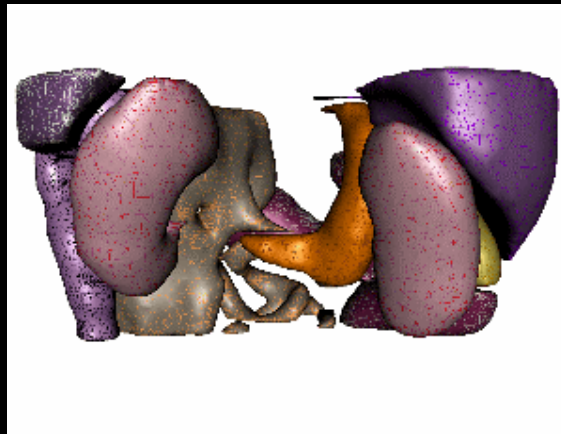


Image data from

- visible human
- MRI
- Laboratory experiments



Automatic construction

- Surface mesh
- Volume mesh
- John Sullivan et al, WPI

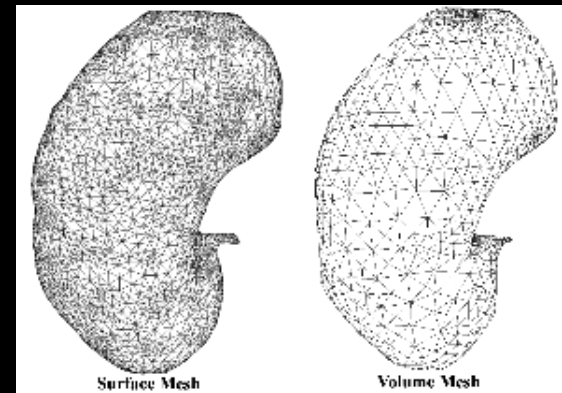
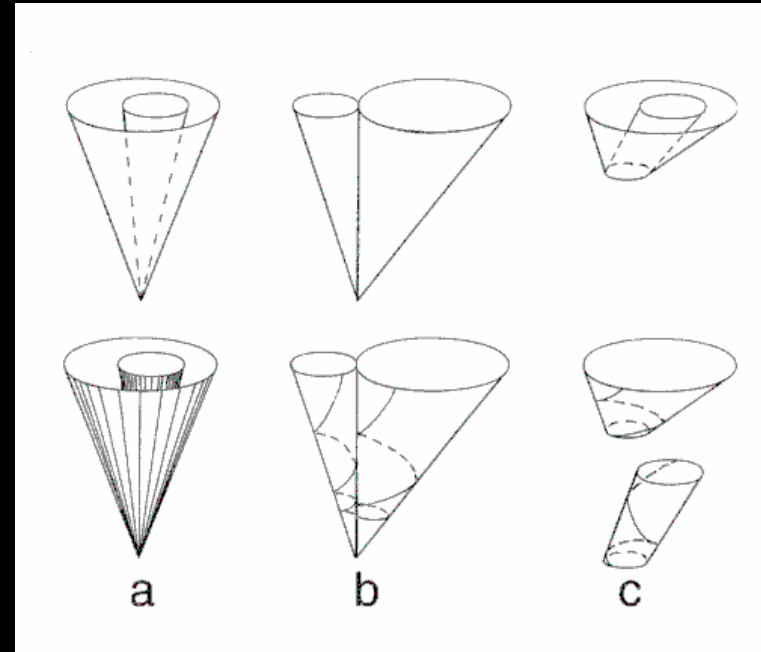


Image source: John Sullivan et al,
WPI

Heart Structure Model

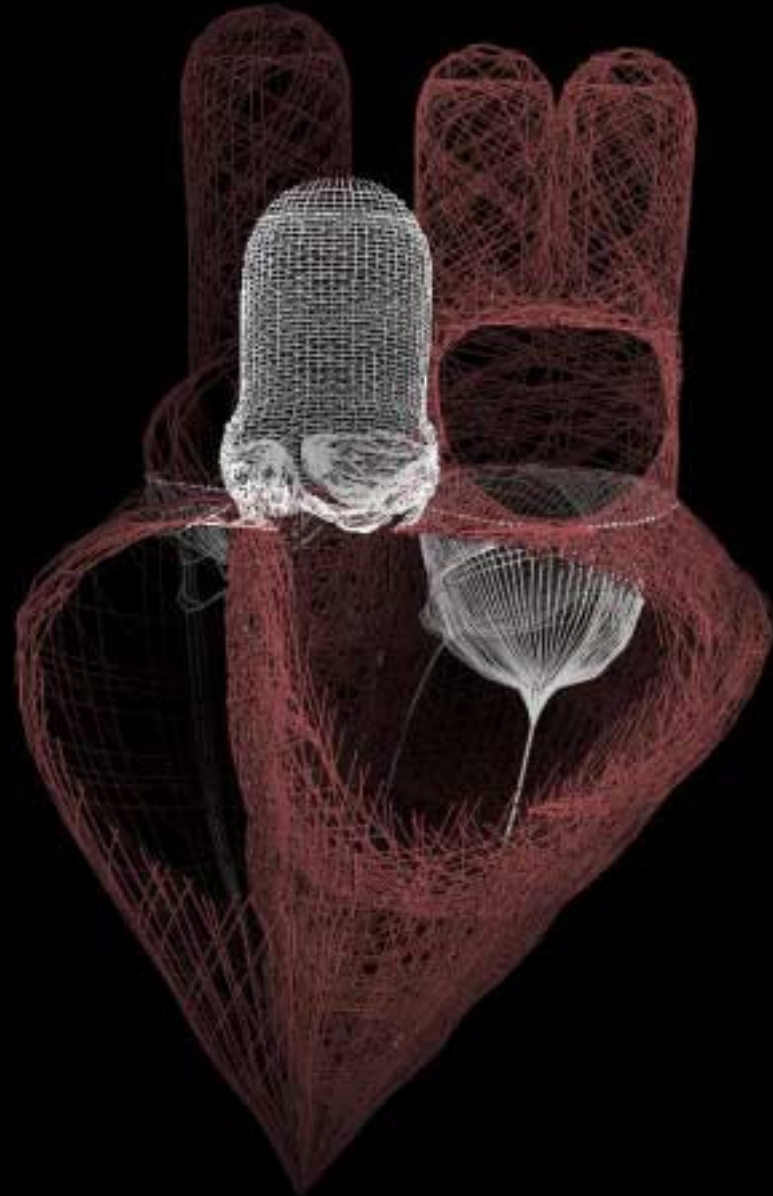
- Current model is based on three types of cones to construct ventricles
 - Outer/Inner layer
 - Right-Inner/Left-Outer
 - Left Cylinder layer



- Advantages: simple model
- Disadvantages: unrealistic and time-consuming to compute

Old Heart Model

- Full structure shows cone shape
- Includes atria, ventricles, valves, and some arteries
- The rest of the circulatory system is modeled by sources and sinks

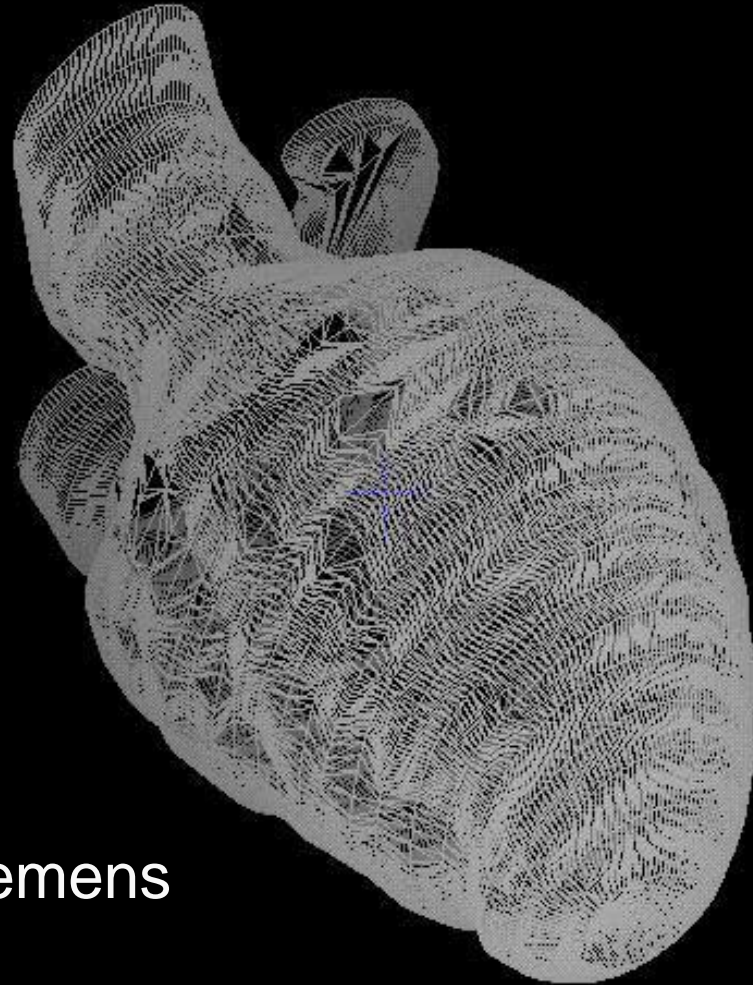


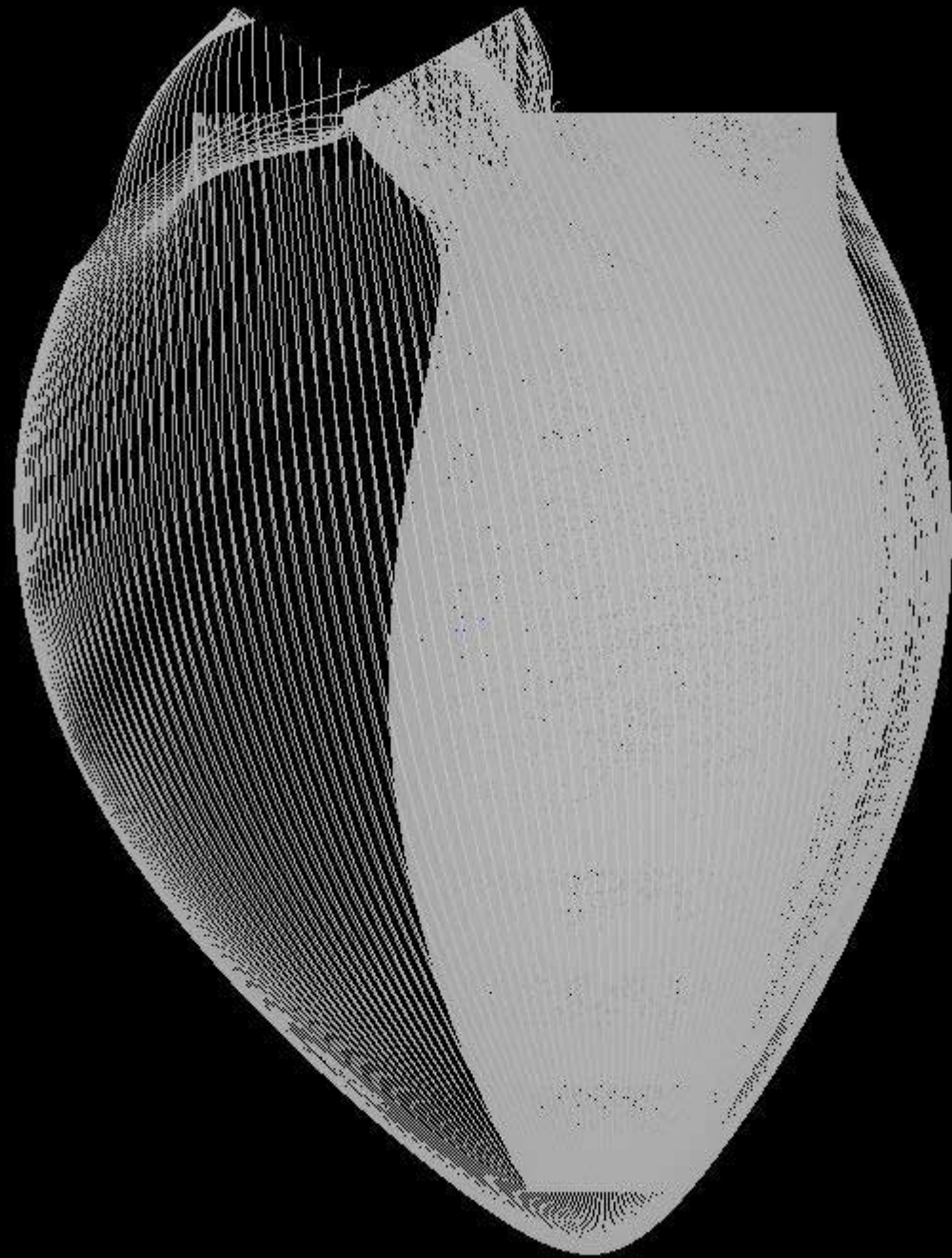
New Heart Model

- New model replaces the geodesics with triangulated surfaces
 - Based on CT scans from a healthy human.
 - Triangulated surface of left ventricle is shown

Work by:

- Peskin & McQueen, NYU
- Paragios & O'Donnell, Siemens
- Setserr, Cleveland Clinic



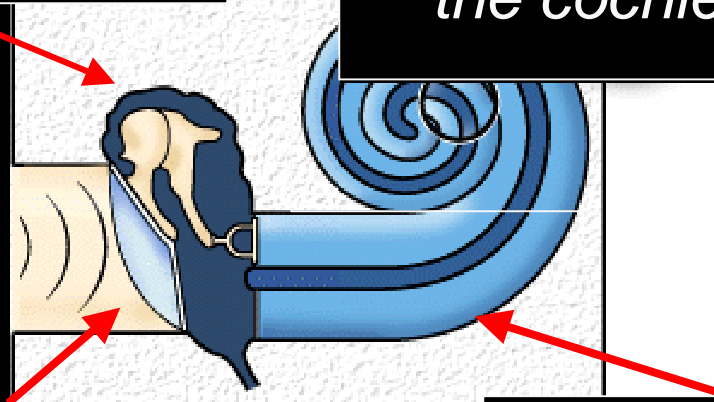


Structure of the Middle Ear

Sound Energy → Cochlear Waves

*The ossicles:
malleus, incus, stapes*

*Transmission of
sound wave energy
by the ossicles from
the ear drum into
the cochlear canal*

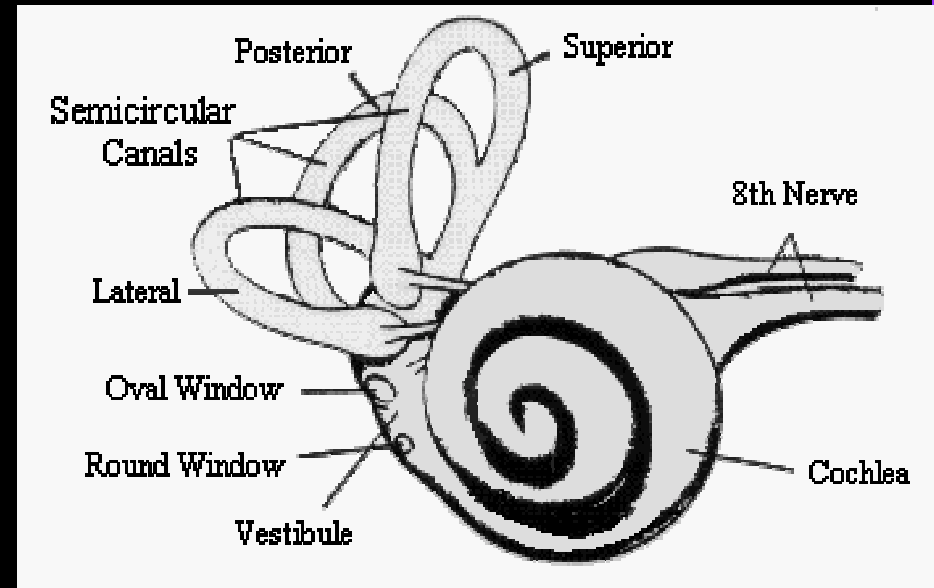


Ear drum

Cochlear canal

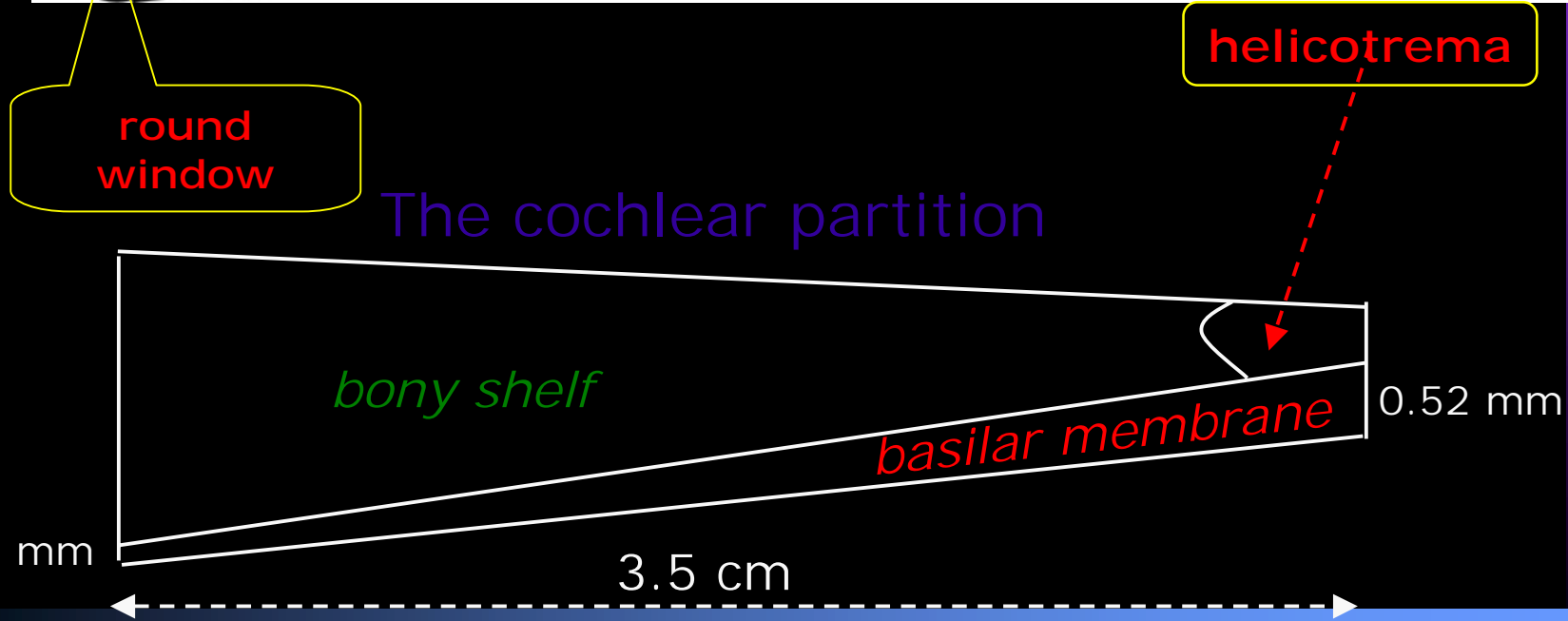
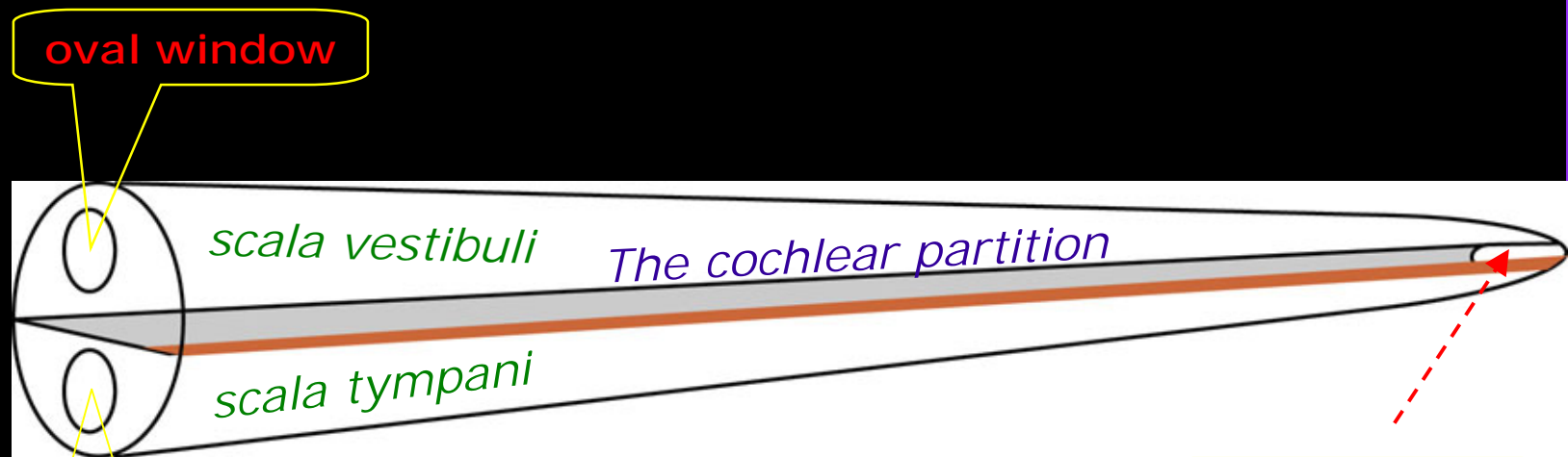
Cochlea and Semi-circular Canals

- The **inner ear** is a fluid-filled cavity containing the cochlea and the semi-circular canals
- Semi-circular canals responsible for balance
- The **fluid** is incompressible and viscous
- Input is from the **stapes** knocking on the **oval window**; the **round window** is covered by a membrane to conserve volume



1 cm

Schematic Description of the Cochlea



Apical Turn of the Cochlea





Immersed Boundary Equations

$$\begin{cases} \rho \left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \right) = -\nabla p + \mu \nabla^2 \mathbf{u} + \mathbf{F} \\ \nabla \cdot \mathbf{u} = 0 \end{cases}$$

$$\mathbf{F}(\mathbf{x}, t) = \int \mathbf{f}(\mathbf{q}, t) \delta(\mathbf{x} - \mathbf{X}(\mathbf{q}, t)) d\mathbf{q}$$

$$\begin{aligned} \frac{\partial \mathbf{X}}{\partial t} &= \mathbf{u}(\mathbf{X}(\mathbf{q}, t), t) \\ &= \int \mathbf{u}(\mathbf{x}, t) \delta(\mathbf{x} - \mathbf{X}(\mathbf{q}, t)) d\mathbf{x} \end{aligned}$$

$\mathbf{u}, p, \mathbf{F}$	fluid velocity, pressure, force
ρ, μ	fluid density, fluid viscosity
\mathbf{f}	the force applied by the immersed matter
\mathbf{x}	position of the fluid particle
\mathbf{X}	position of the immersed matter particle

First Order Immersed Boundary Method

Compute the force \mathbf{f} the immersed material applies to the fluid.

Compute the force applied to the fluid grid:

$$\mathbf{F}^n(\mathbf{x}) = \sum_{\mathbf{q}} \mathbf{f}^n(\mathbf{q}) \delta_h(\mathbf{x} - \mathbf{X}^n(\mathbf{q})) \Delta \mathbf{q}$$

Solve the Navier-Stokes equations:

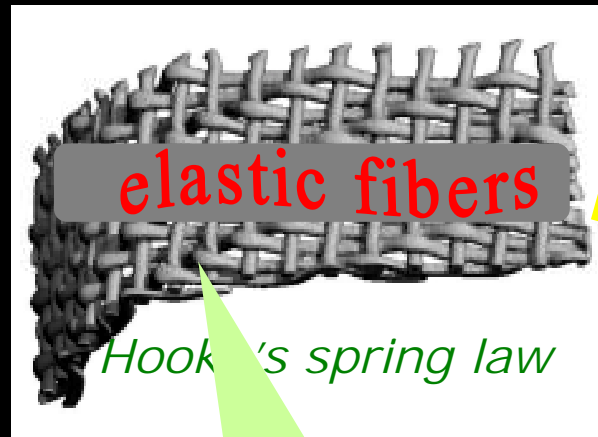
$$\rho \left(\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t} + \sum_{\alpha=1}^3 u_{\alpha}^n D_{\alpha}^{\pm} \mathbf{u}^n \right) = -D^0 p^{n+1} + \mu \sum_{\alpha=1}^3 D_{\alpha}^{+} D_{\alpha}^{-} \mathbf{u}^{n+1} + \mathbf{f}^n$$
$$D^0 \cdot \mathbf{u}^{n+1} = 0$$

Move the material:

$$\mathbf{X}^{n+1}(\mathbf{q}) = \mathbf{X}^n(\mathbf{q}) + \Delta t \sum_{\mathbf{x}} \mathbf{u}^{n+1}(\mathbf{x}) \delta_h(\mathbf{x} - \mathbf{X}(\mathbf{q})) h^3$$

Immersed Boundary Method

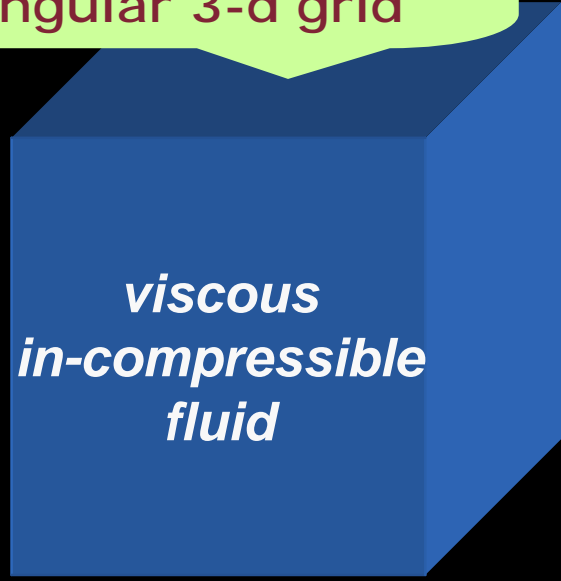
Combines Lagrangean and Eulerian Components



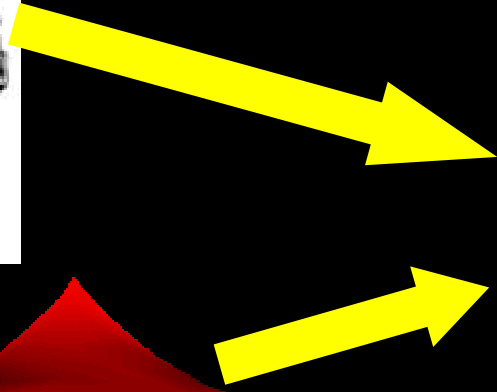
Hooke's spring law
Discretized
on a 1-d
grid



Navier-Stokes equations
discretized on a periodic
rectangular 3-d grid



Fourth order PDE
discretized on a 2-d
grid



Immersed Boundary Method Structure

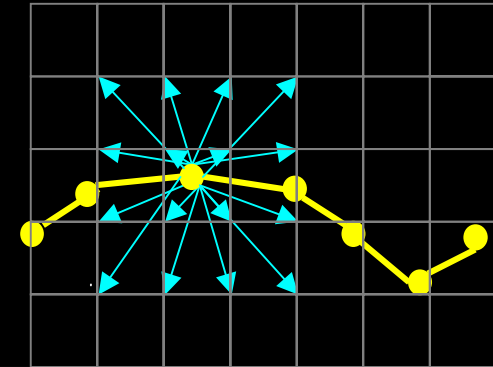
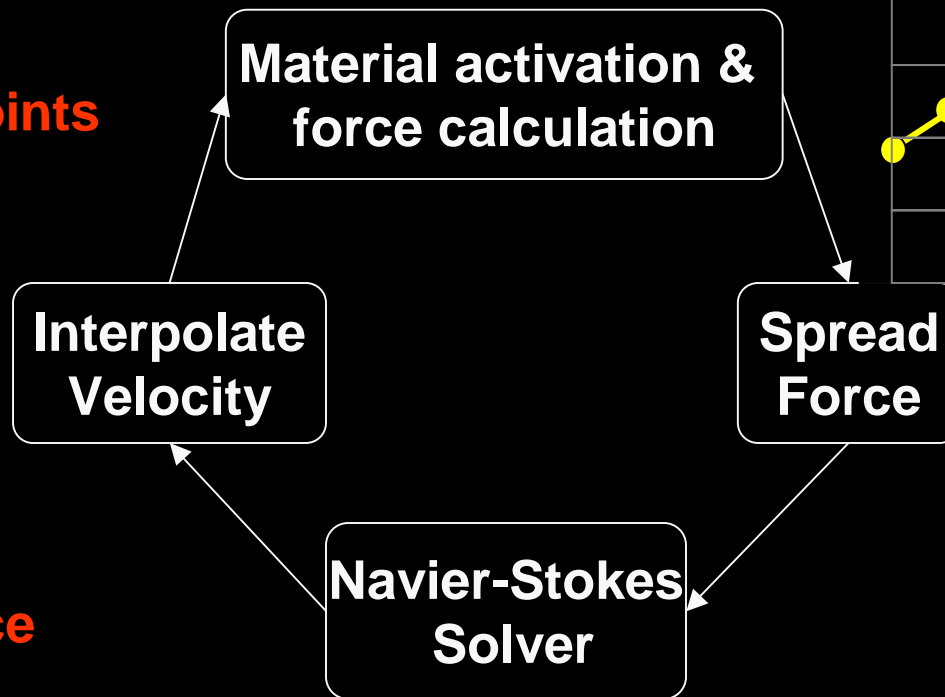
4 steps in each timestep

2D Dirac Delta Function

Material Points

Interaction

Fluid Lattice



Challenges to Parallelization

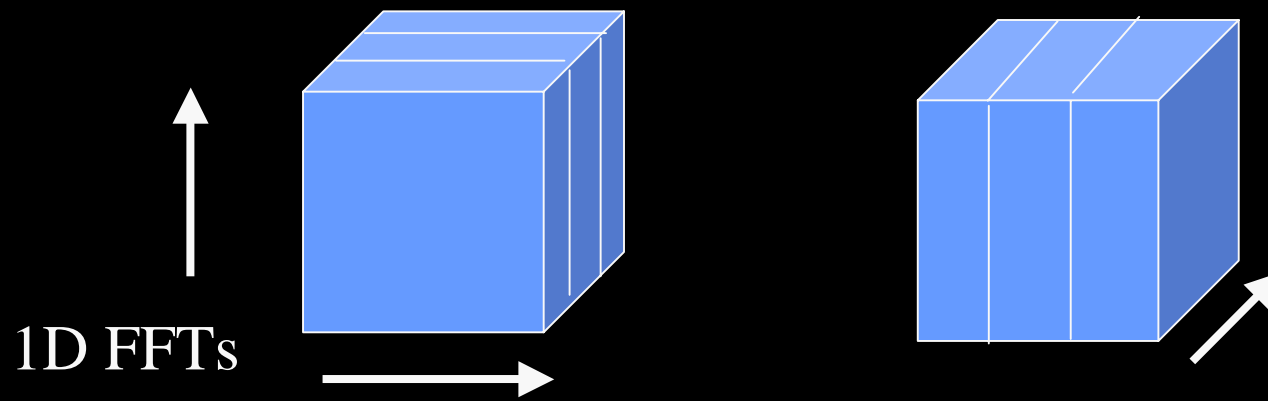
- Irregular material points need to interact with regular fluid lattice.
 - Trade-off between load balancing of material and minimizing communication
 - Efficient “scatter-gather” across processors
- Need a scalable fluid solver
 - Currently based on 3D FFT
 - Requires an all-to-all “transpose”
 - May try to use multigrid in the future
 - Adaptive Mesh Refinement would help

Parallel Algorithm

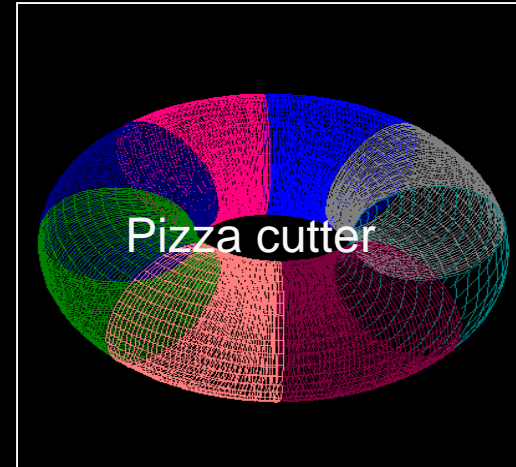
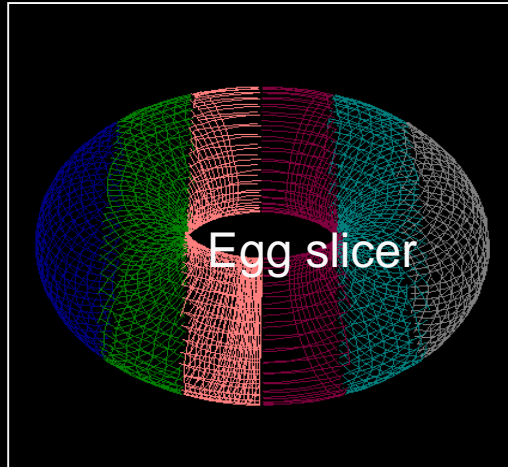
- **Immersed materials** are described by a hierarchy of 1d or 2d arrays
- Grids in current code
 - Reside on a single processor
 - Previous (and possibly future) versions may split them
- The 3D **fluid grid** uses a 1D distribution (slabs)
- **Interactions** between the fluid and material structures requires inter-processor communication.
- Special data structures are maintained for efficient communication.

Fluid Solver

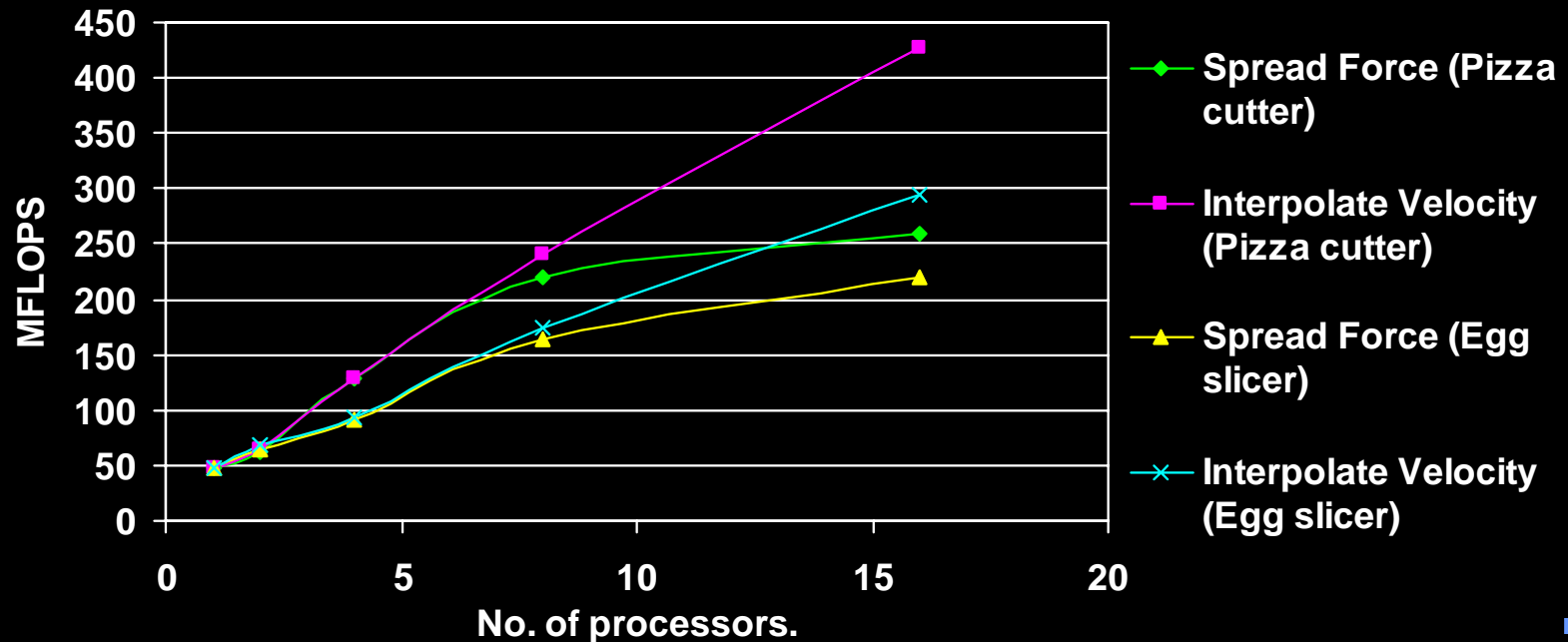
- The incompressible requires an elliptic solver
 - High communication demand
 - Information propagates across domain
- FFT-based solver divides domain into slabs
 - Transposes before last direction of FFT
 - Would like to use finer decomposition



Load Balancing

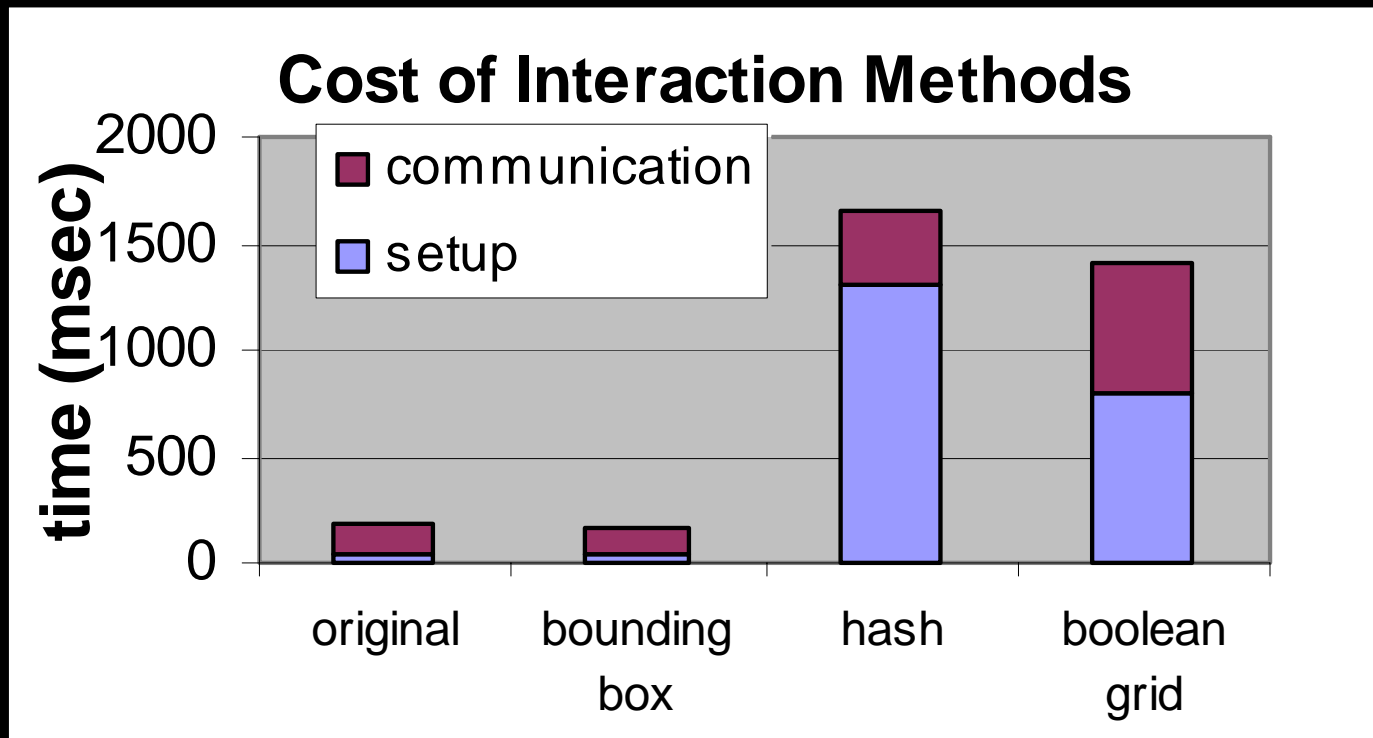


Fluid grid is divided in slabs for 3D FFT

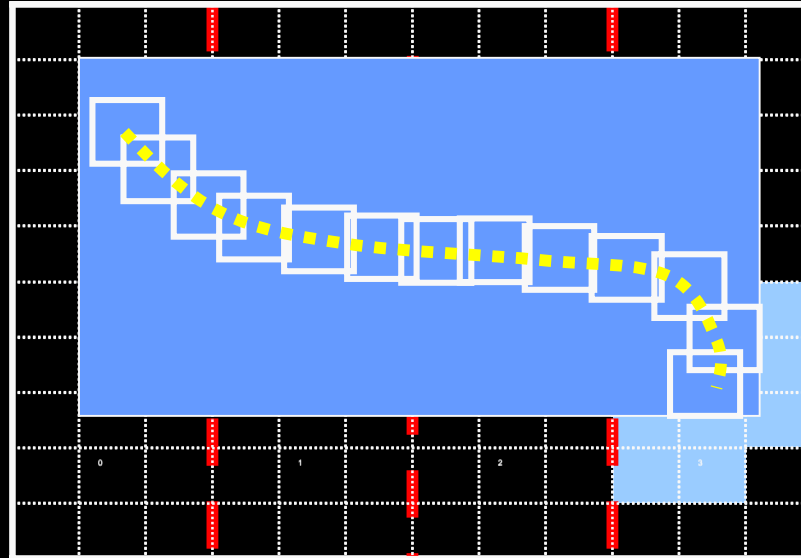


Data Structures for Interaction

- Experimented with several method
- Bounding box is the best (although it sends significantly more data than necessary)

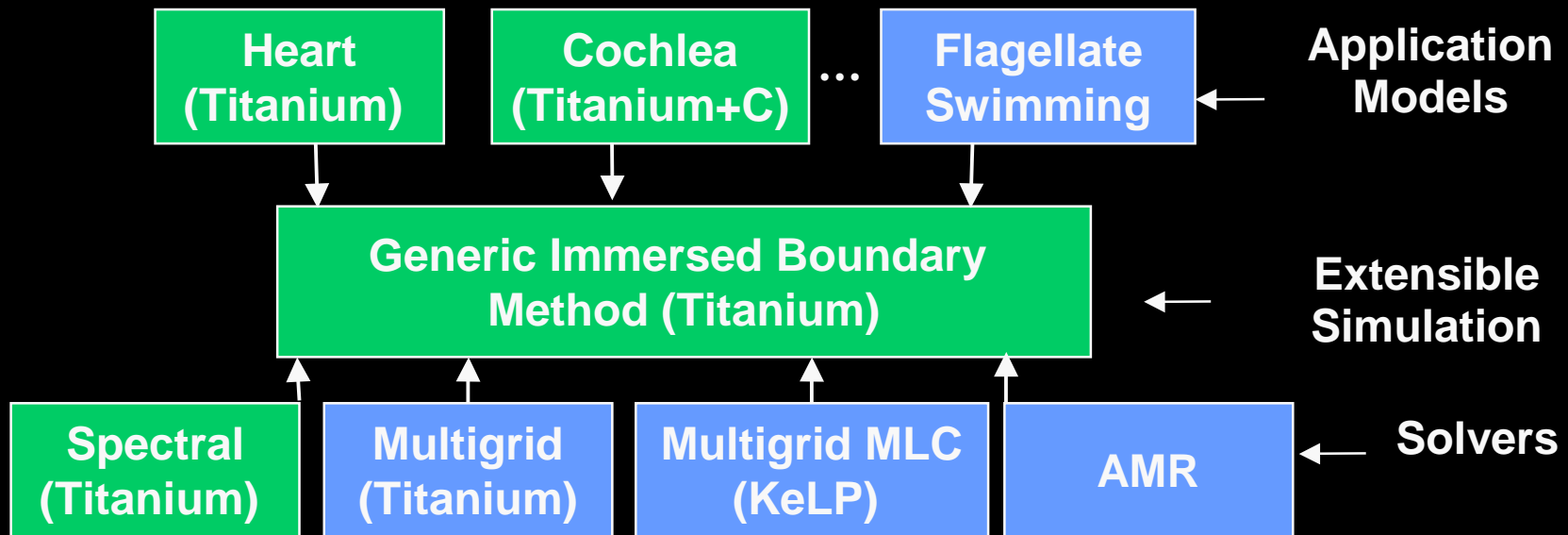


Data Structures for Interaction



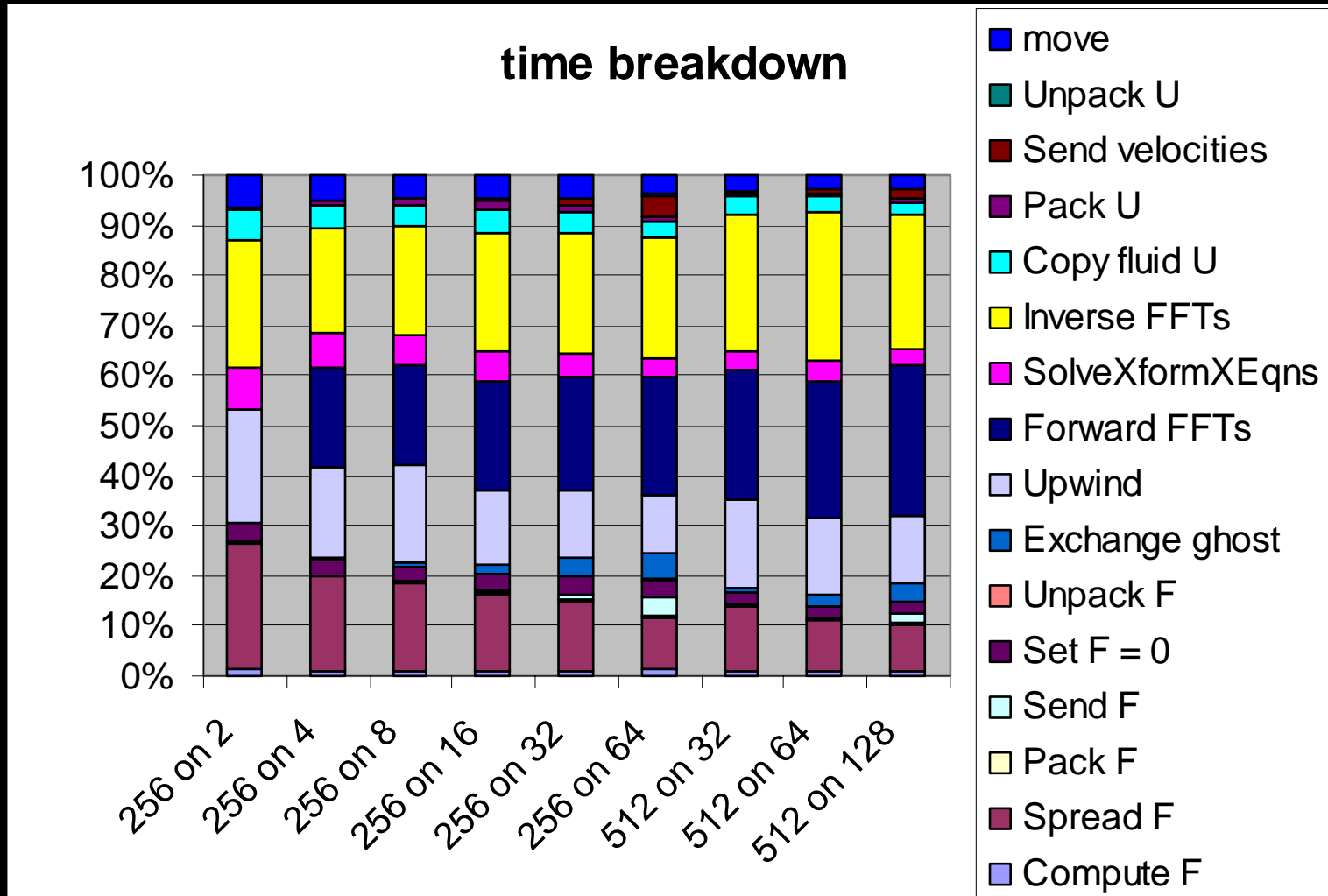
- Bounding box computes only low/high
- Logical grid of 4x4x4 cubes used to balance cost of communication and setup
- Communication aggregation also done

Software Architecture



- Can add new models by extending material points
- Uses Java inheritance for simplicity

Performance Analysis



Tools for High Performance

Challenges to parallel simulation of a digital human are generic

- Parallel machines are too hard to program
 - Users “left behind” with each new major generation
- Efficiency is too low
 - Even after a large programming effort
 - Single digit efficiency numbers are common
- Approach
 - Titanium: A modern (Java-based) language that provides performance transparency
 - BeBOP: Self-tuning scientific kernels
 - GASNet: Portable fast communication

Titanium Overview

Object-oriented language based on Java with:

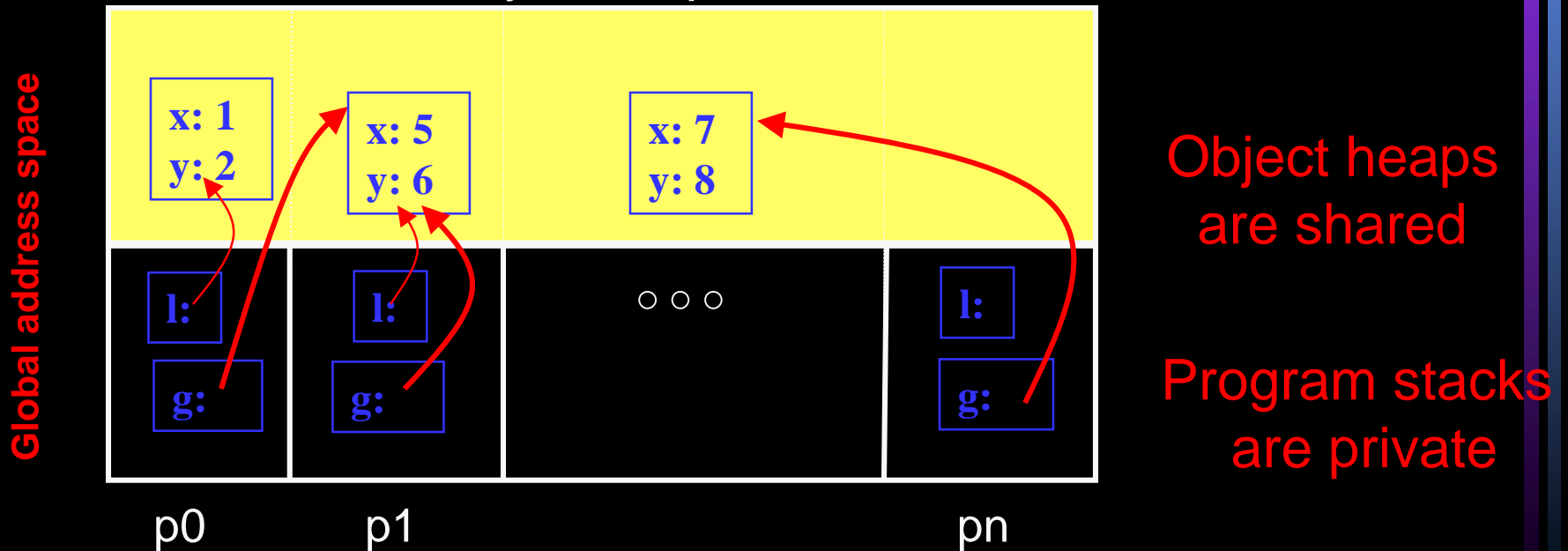
- Scalable parallelism
 - Single Program Multiple Data (SPMD) model of parallelism, 1 thread per processor
- Global address space
 - Processors can read/write memory on other processor
 - Pointer dereference can cause communication
- Intermediate point between message passing and shared memory

Language Support for Performance

- Multidimensional arrays
 - Contiguous storage
 - Support for sub-array operations without copying
- Support for small objects
 - E.g., complex numbers
 - Called “immutable” in Titanium
 - Sometimes called “value” classes
- Semi-automatic memory management
 - Create named “regions” for new and delete
 - Avoids distributed garbage collection
- Optimizations on parallel code
 - Communication and memory hierarchies

Global Address Space Languages

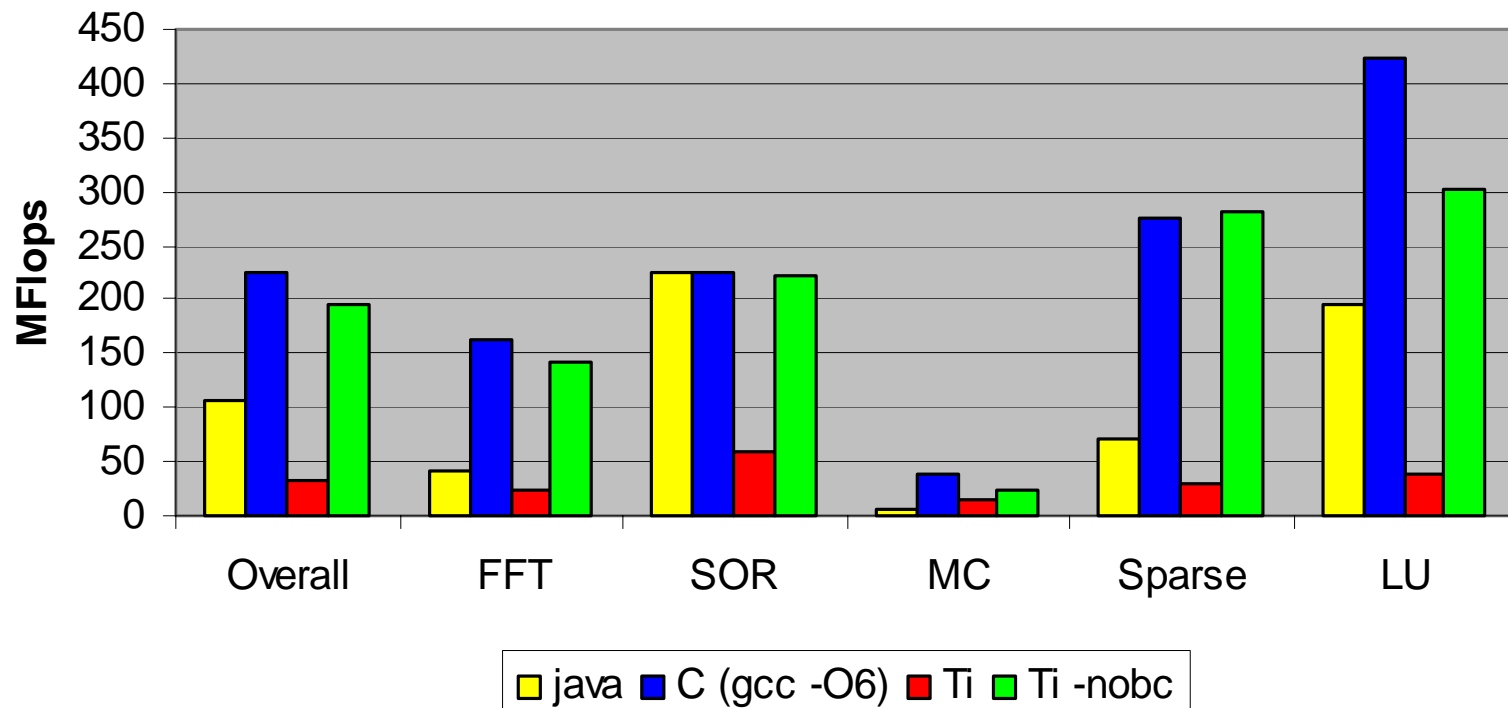
- Titanium is similar to UPC and Co-Array Fortran
- Globally shared address space is partitioned
- References (pointers) are either local or global (meaning possibly remote)
- Distributed arrays and pointer-based structures



- Static parallelism (like MPI)

Performance of Titanium Compiler

Performance on a Pentium IV (1.5GHz)



Titanium Research Problems

- Analysis of explicitly parallel code
- Optimizations for
 - Memory hierarchies
 - Communication (overlap and aggregation)
 - Synchronization
- Dynamic as well as static optimizations
 - For sparse and unstructured data
 - Extensible language (compiler support for scientific data structures)
- Lightweight one-sided communication
 - Joint with UPC group
 - GASNet layer

Semantics: Sequential Consistency

- When compiling sequential programs:

```
x = expr1;  
y = expr2;
```



```
y = expr2;  
x = expr1;
```

Valid if y not in $expr1$ and x not in $expr2$ (roughly)

- When compiling parallel code, not sufficient test.

```
Initially flag = data = 0
```

```
Proc A
```

```
data = 1;
```

```
flag = 1;
```

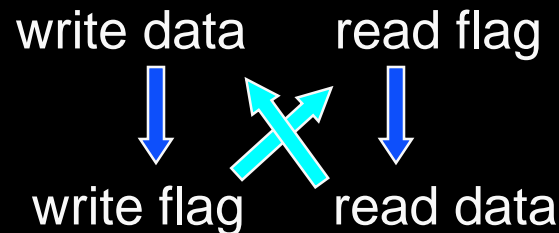
```
Proc B
```

```
while (flag!=1);
```

```
... = ...data...;
```

Cycle Detection: Analysis Problem

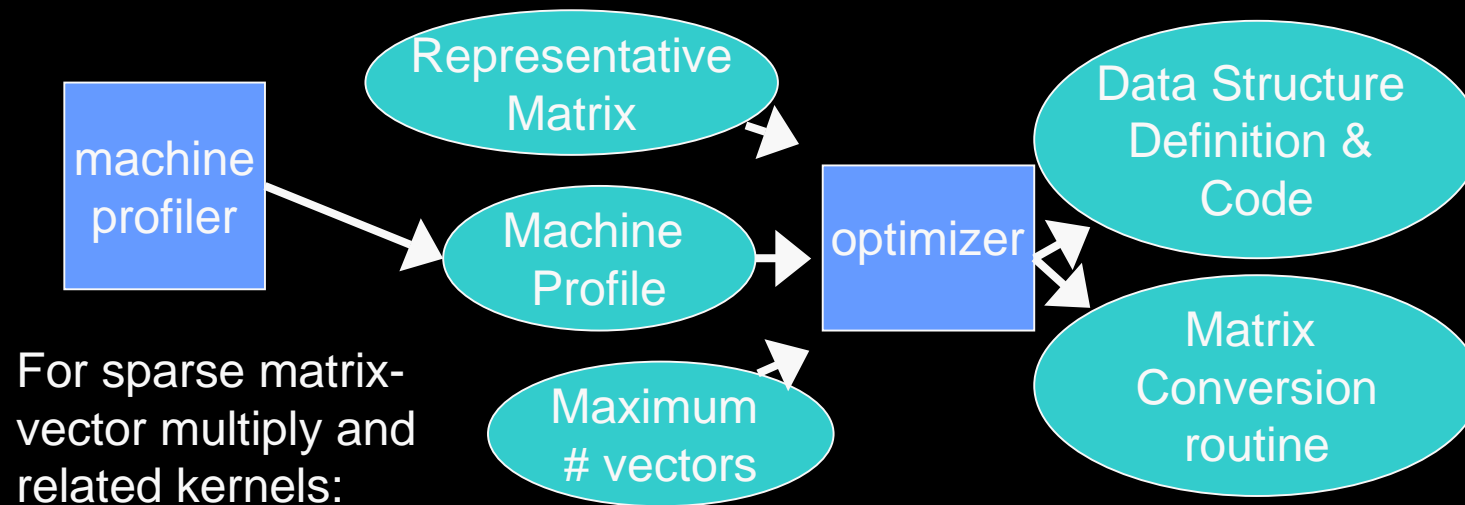
- Processors define a “program order” on accesses from the same thread
 - P is the union of these total orders
- Memory system define an “access order” on accesses to the same variable
 - A is access order (read/write & write/write pairs)



- A violation of sequential consistency is cycle in $P \cup A$.
- Intuition: time cannot flow backwards.

Automatic Performance Tuning

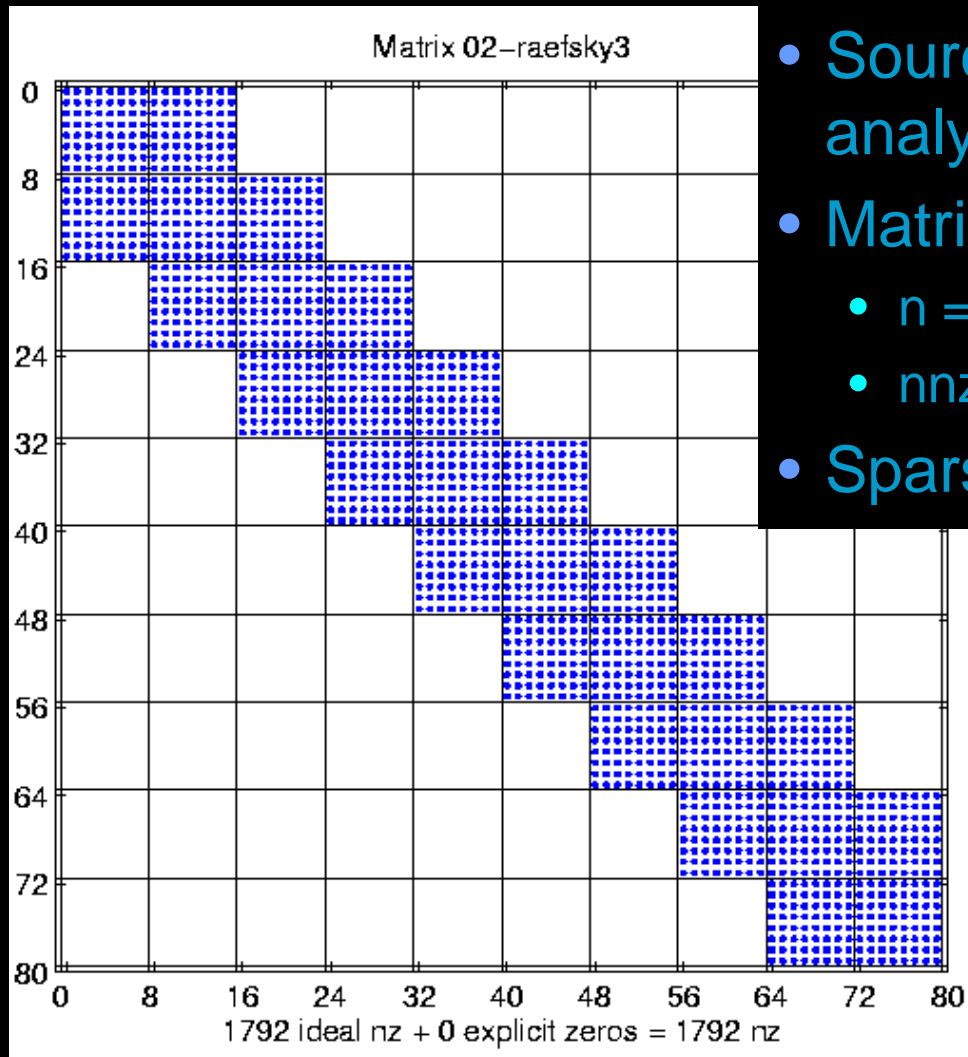
- Problem: low single-processor performance
 - 100s of arithmetic operations per memory operation
 - Complex processors and memory systems are challenging
 - Techniques like tiling help, but parameters are hard to find
- Solution: let computers do automatic tuning
 - FFTW, Atlas (dense linear algebra), Titanium for multigrid
 - BeBOP: sparse matrix kernels, optimizations depend on matrix



Summary of Optimizations

- Optimizations for sparse matrix-vector multiply
 - Register blocking: up to **4x**
 - Variable block splitting: **2.1x**
 - Diagonals: **2x**
 - Reordering to create dense structure + splitting: **2x**
 - Symmetry: **2.8x**
 - Cache blocking: **2.2x**
 - Multiple vectors (SpMM): **7x**
 - And combinations...
- Sparse triangular solve
 - Hybrid sparse/dense data structure: **1.8x**
- Higher-level kernels
 - $AA^T \mathbf{x}$, $A^T A \mathbf{x}$: **4x**
 - $A^2 \mathbf{x}$: **2x**

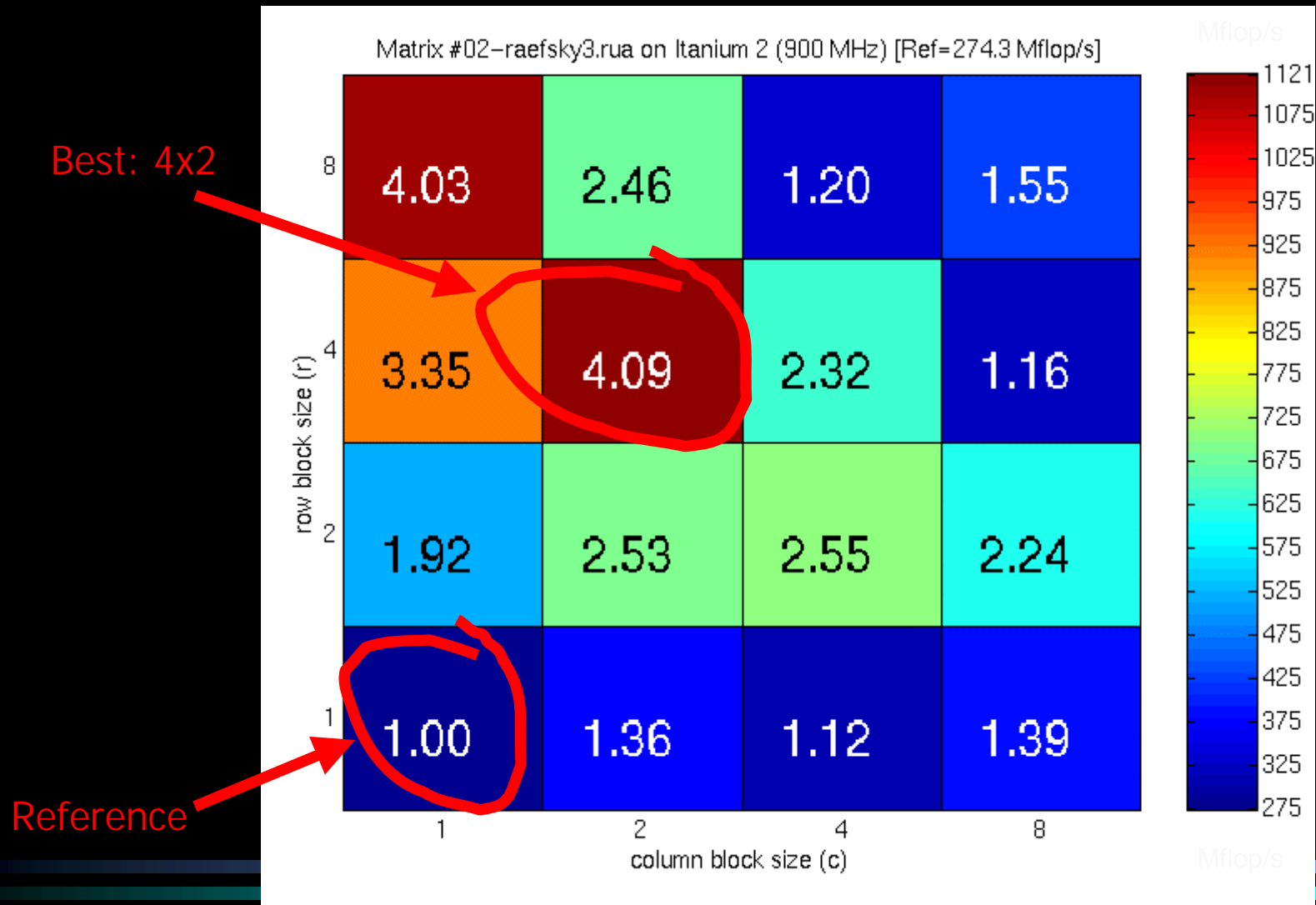
Example: The Difficulty of Tuning



- Source: NASA structural analysis problem
- Matrix:
 - $n = 21216$
 - $nnz = 1.5 \text{ M}$
- Sparse matrix-vector multiply
- Register blocking: store each block contiguously with a single index
- Use 8x8 blocks to math structure, right?

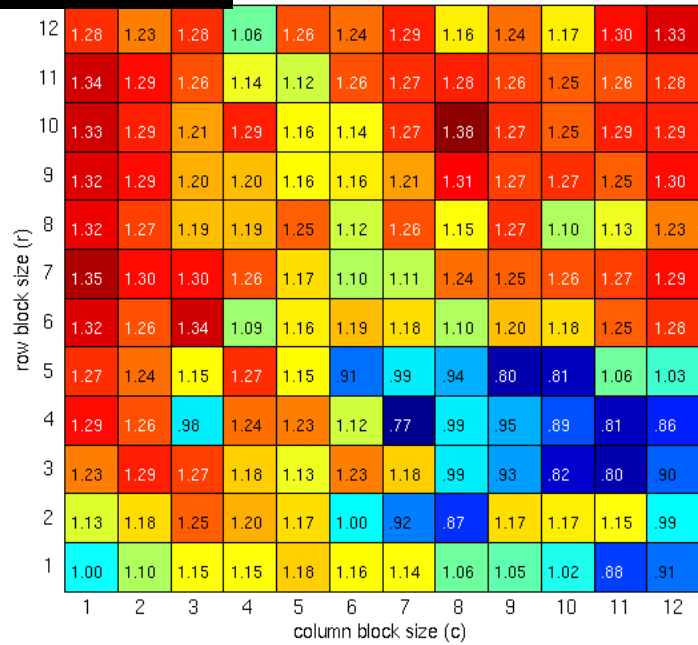
Speedups from Blocking on Itanium 2

The “natural” block size is far from optimal: search for best.

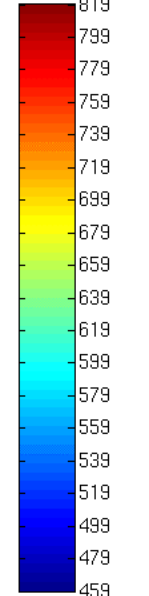


Power4 - 16%

Profile [ref=594.9 Mflop/s; 1.3 GHz Power4, IBM xlc v6]



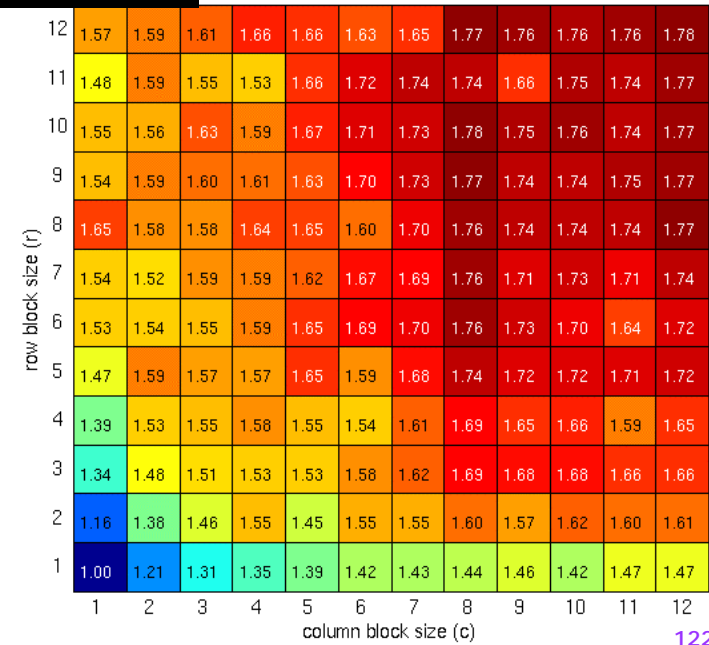
820 Mflop/s



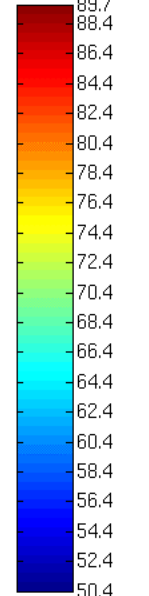
459 Mflop/s

Ultra 3 - 5%

Profile [ref=50.3 Mflop/s; 900 MHz Sun Ultra 3, Sun C v6.0]



90 Mflop/s

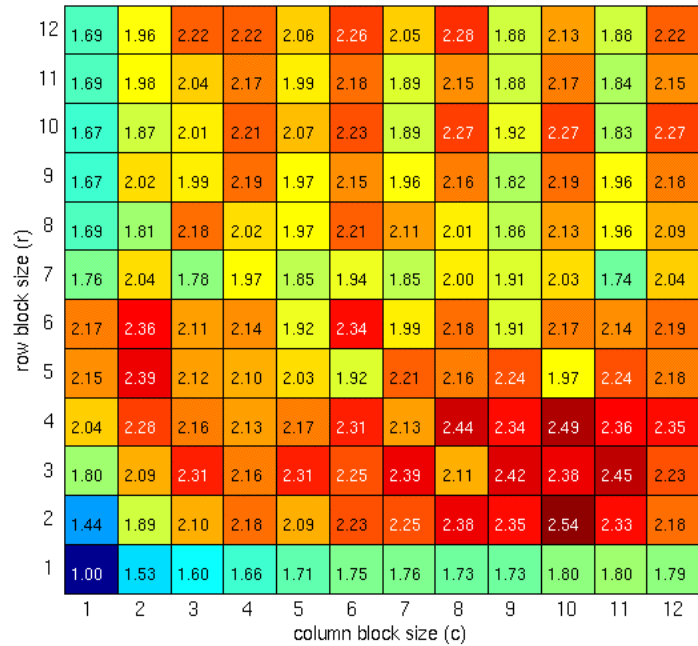


50 Mflop/s

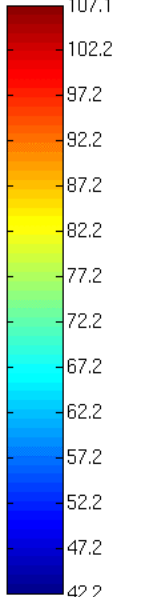
122 Mflop/s

Pentium III - 21%

Profile [ref=42.1 Mflop/s; 500 MHz Pentium III, Intel C v7.0]



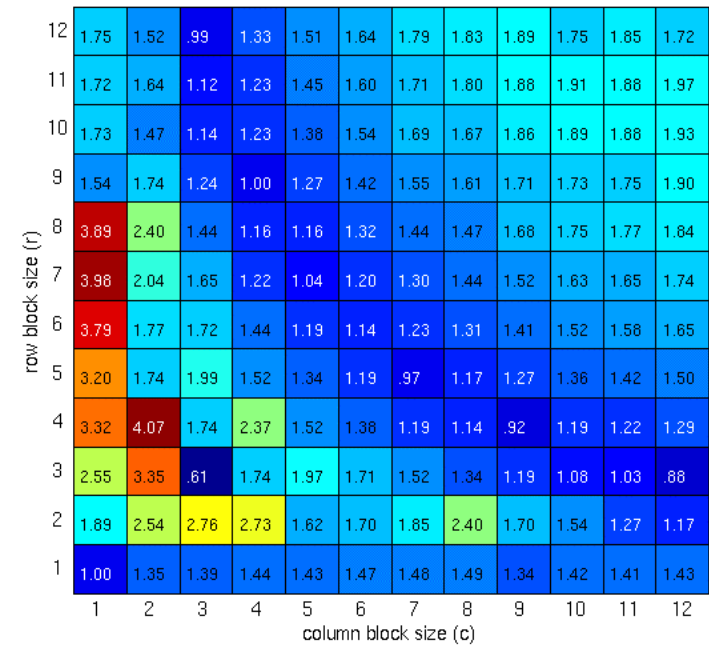
108 Mflop/s



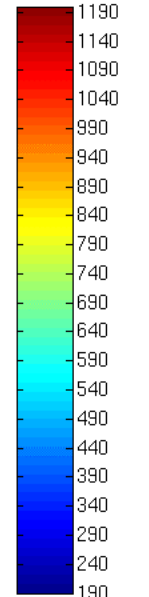
42 Mflop/s

Itanium 2 - 33%

Profile [ref=294.5 Mflop/s; 900 MHz Itanium 2, Intel C v7.0]

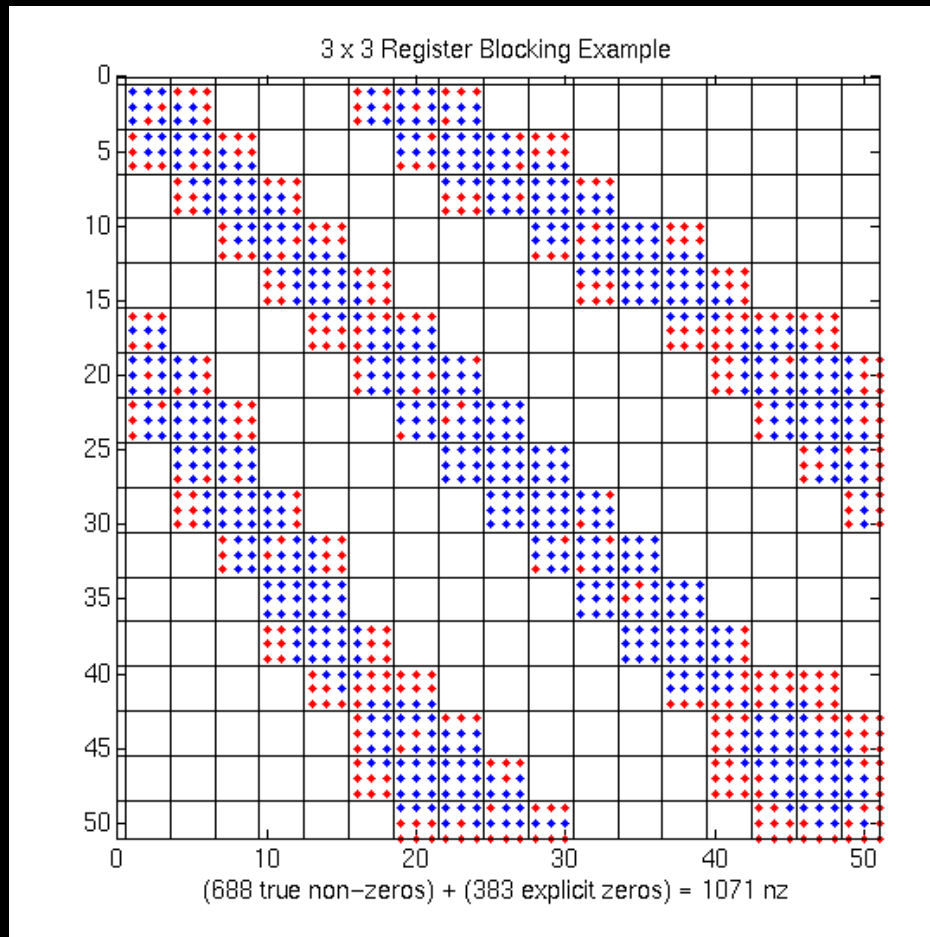


1.2 Gflop/s



190 Mflop/s

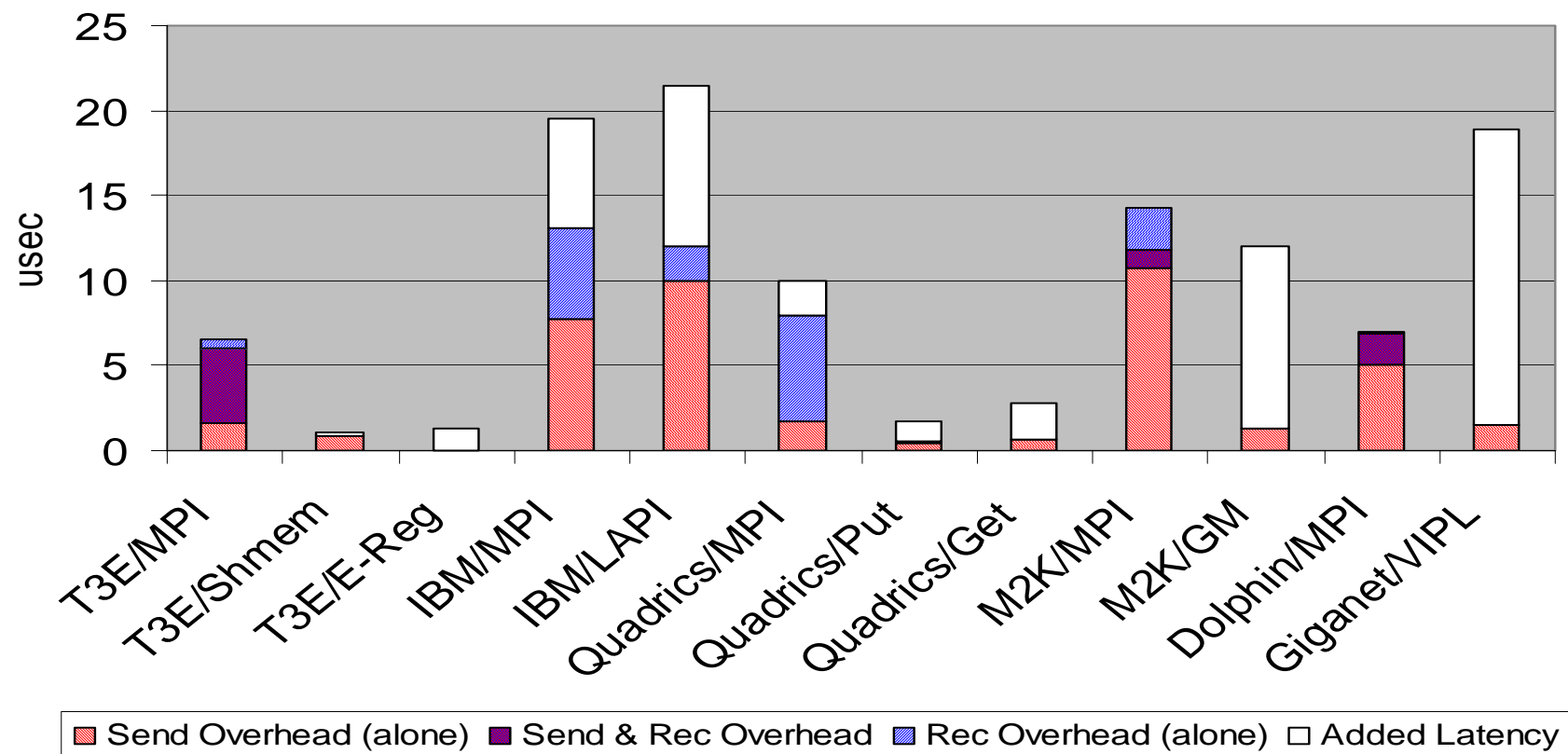
Extra Work Can Improve Efficiency!



- More complicated non-zero structure in general
- Example: 3x3 blocking
 - Logical grid of 3x3 cells
 - Fill-in explicit zeros
 - Unroll 3x3 block multiplies
 - “Fill ratio” = 1.5
- On Pentium III: **1.5x speedup!**

Network Performance Tuning

- Two-sided message passing (MPI) is not the fastest form of communication
- Low latency/overhead allows for easier implementations

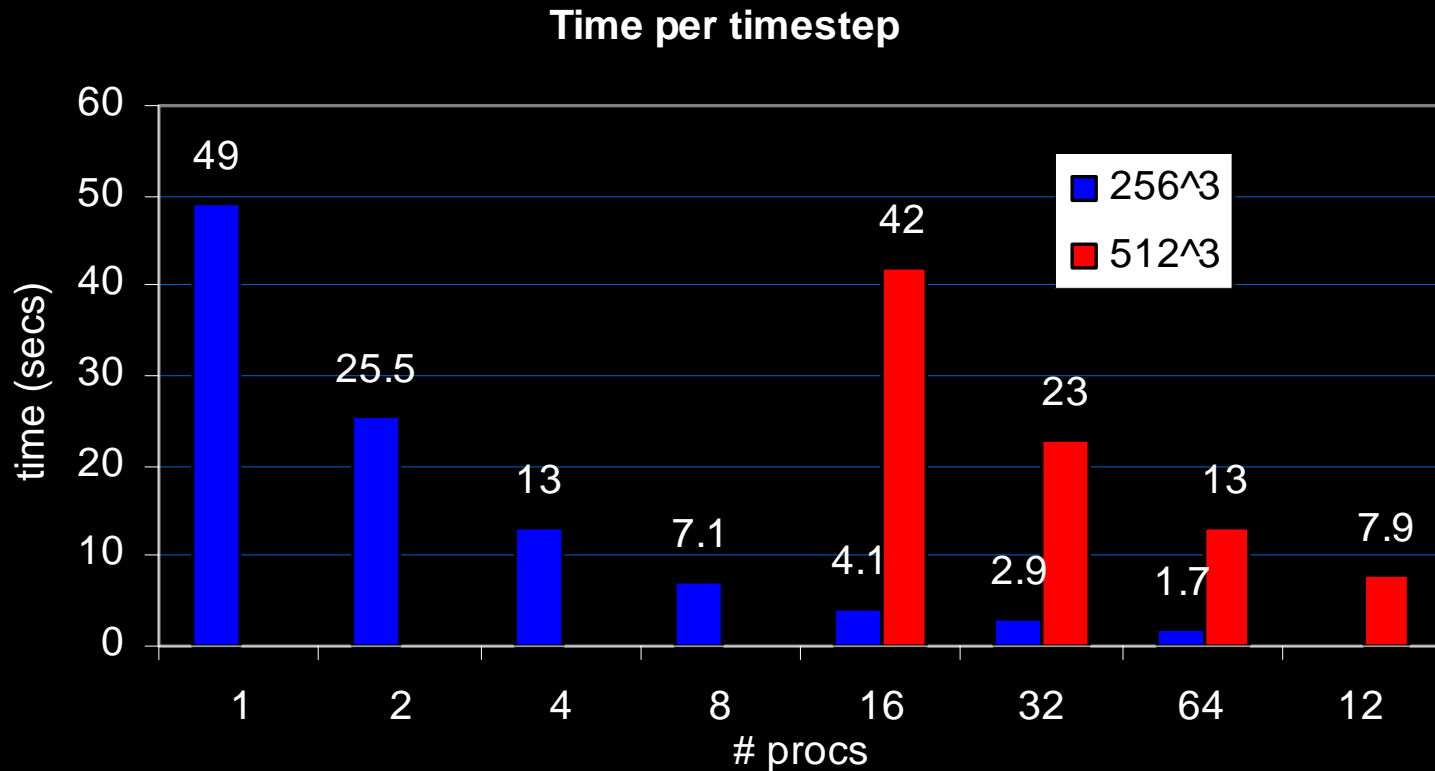


Putting it all together

Performance of the Immersed Boundary
code using:

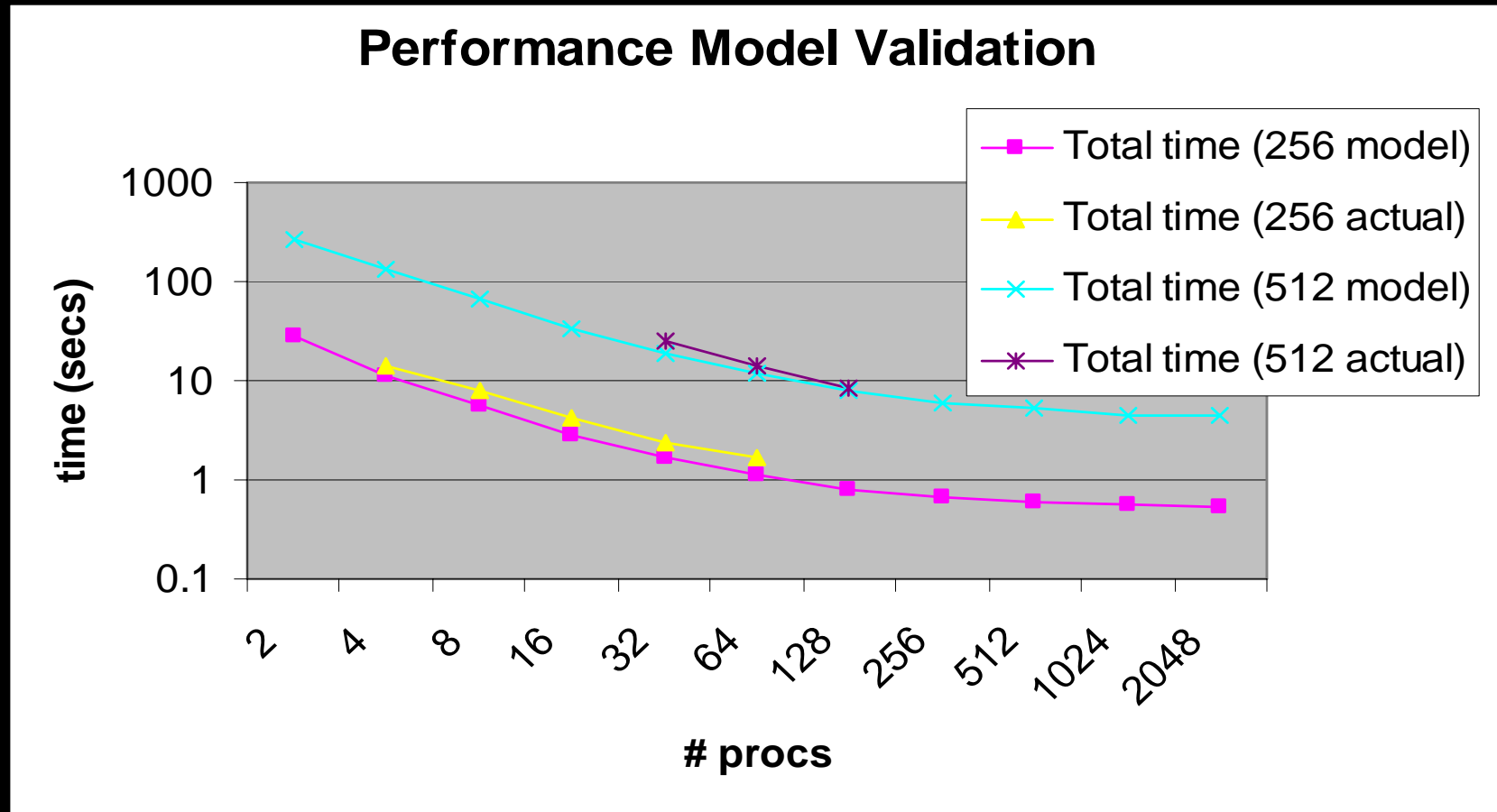
- Titanium
- GASNet
- Automatically tuned FFTs (FFTWs)
(No sparse matrices yet)

Scaling Behavior (Synthetic Problem)



- Measured on the IBM SP at NERSC
- Also run on Itanium/Myrinet clusters and elsewhere

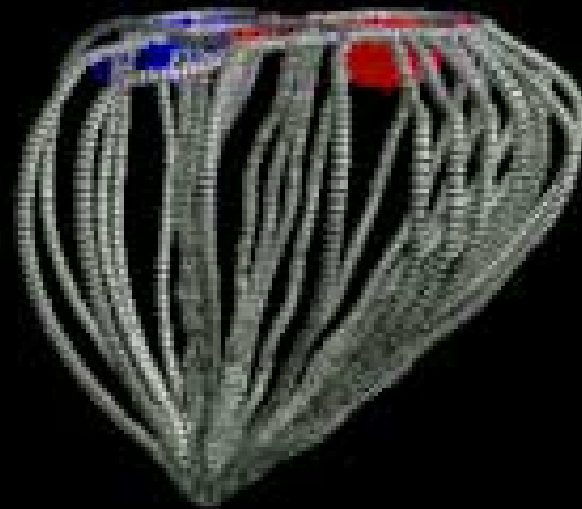
A Performance Model



- 512^3 in < 1 second per timestep not possible
- 10x increase in bisection bandwidth would fix this

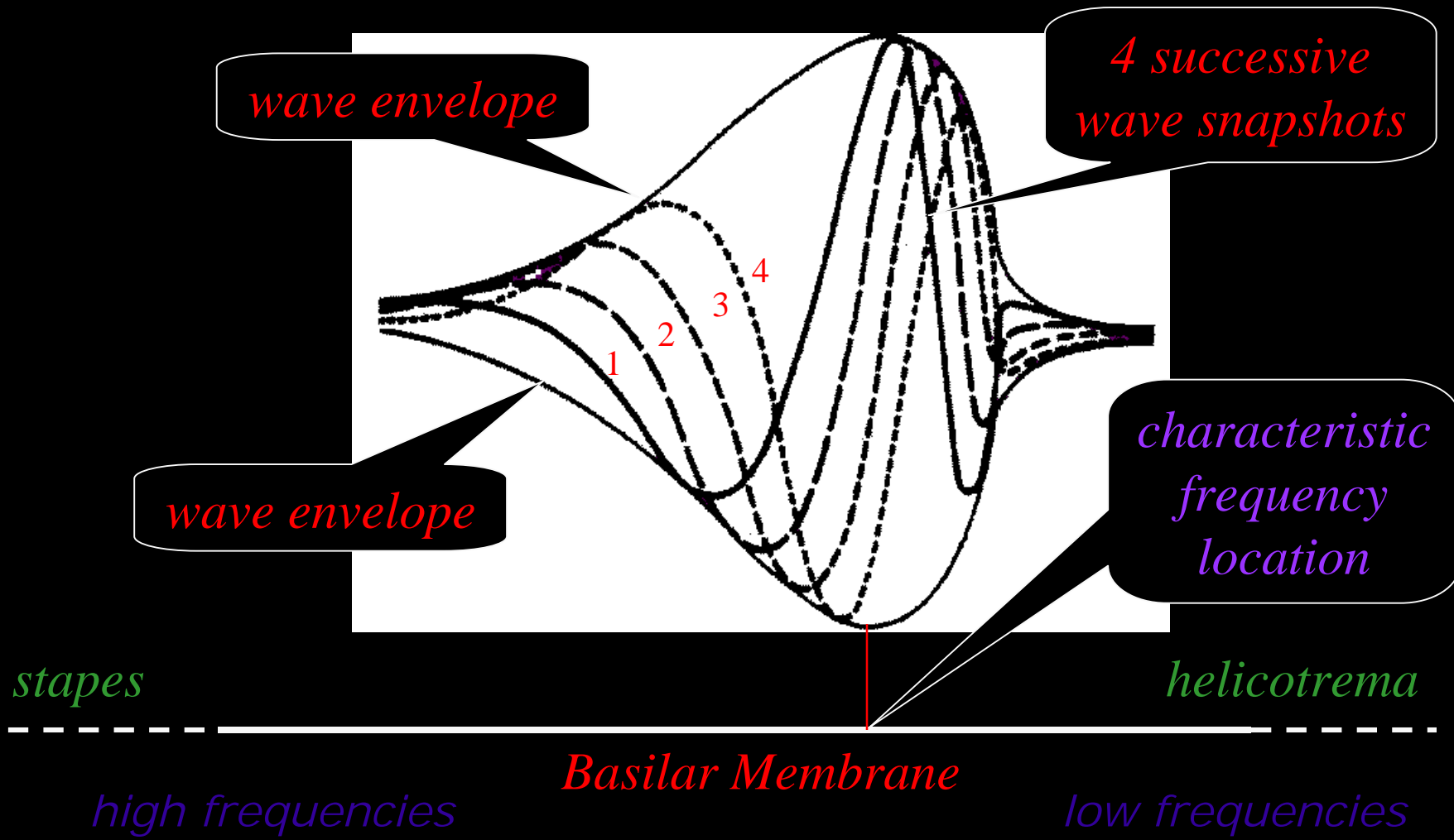
Heart Simulation

Animation of lower portion of the heart

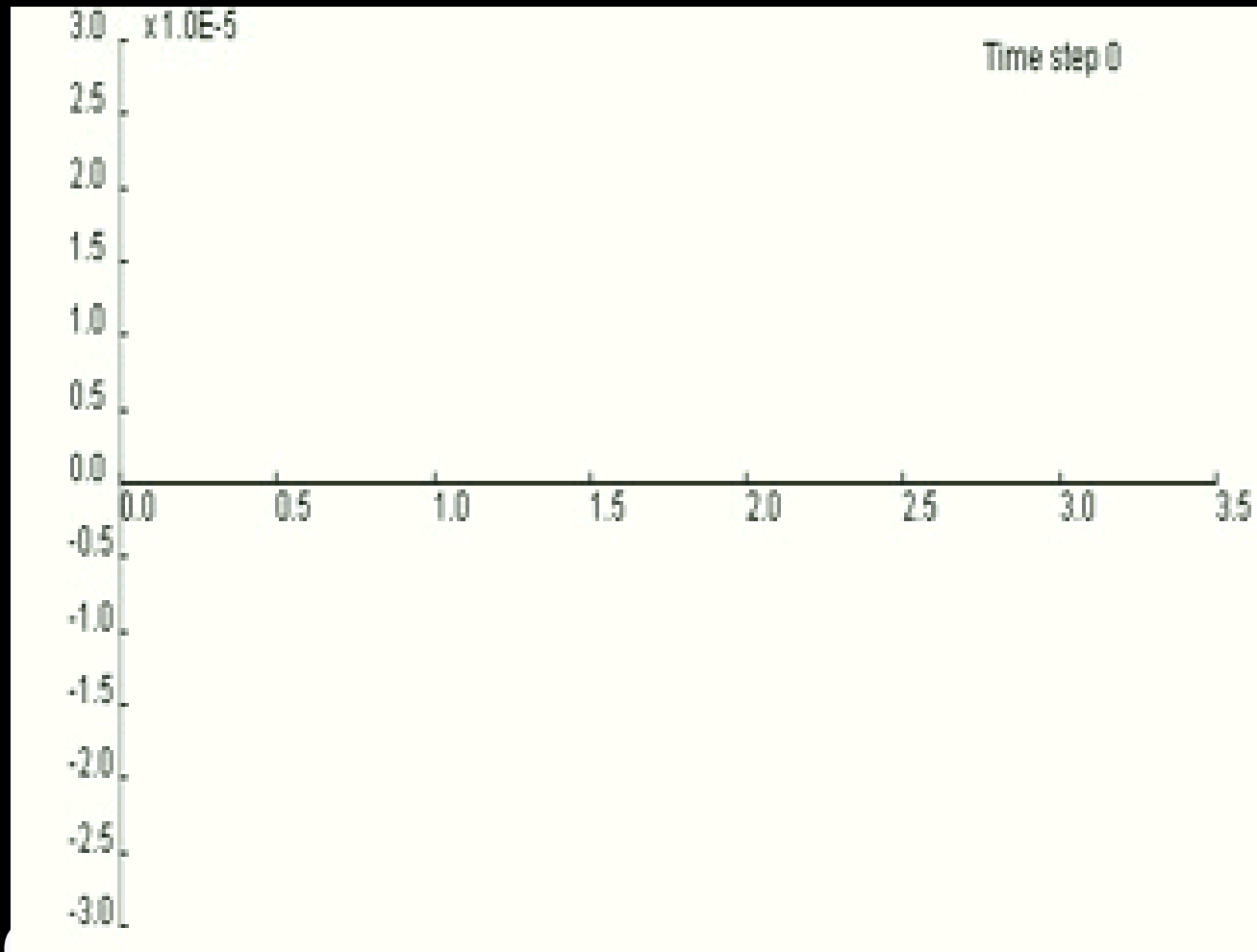


Source: www.psc.org

Traveling Wave in the Cochlea



Sound Wave Propagation in Cochlea



- Centering of the Basilar Membrane
- Response to 10 KHz frequency input

Future Research

- Variable timestepping
- Improved scalability
 - Finer decomposition of materials and fluid
 - Multigrid or other solvers
- Second-order accurate method
- Adaptive Mesh Refinement
- Multi-physics models
 - Incorporating diffusion, electrical models, etc.
 - Needed for cardiac muscle contraction
 - Needed for Outer Hair Cell in cochlea

Collaborators

Titanium Faculty:

- Susan Graham
- Paul Hilfinger
- Alex Aiken
- Phillip Colella, LBNL

Bebop faculty

- Jim Demmel
- Eun-Jin Im, Kookmin

NYU IB Method:

- **Charlie Peskin**
- **Dave McQueen**

Students, Postdocs, Staff:

- Christian Bell
- Wei Chen
- Greg Balls, SDSC
- Dan Bonachea
- **Ed Givelberg***

- Peter McQuorquodale, LBNL
- Tong Wen, LBNL
- Mike Welcome, LBNL
- Jason Duell, LBNL
- Paul Hargrove, LBNL
- Christian Bell
- Wei Chen
- **Sabrina Merchant**
- Kaushik Datta
- Dan Bonachea
- Rich Vuduc
- Amir Kamil
- Omair Kamil
- Ben Liblit
- Meling Ngo
- Geoff Pike, ISI

- Jimmy Su
- **Siu Man Yau**
- Shaoib Kamil
- Benjamin Lee
- Rajesh Nishtala
- Costin Iancu, LBNL
- David Gay, Intel
- **Armando Solar-Lezama**

<http://titanium.cs.berkeley.edu/>

<http://upc.lbl.gov>

<http://bebop.cs.berkeley.edu>

* Primary researchers on the IB simulation