

# TurbulenceDB: A Data-Intensive Architecture for the Analysis of Multi-scale Fluid Simulations

Randal Burns

Institute for Data Intensive  
Engineering and Science (IDIES)  
Johns Hopkins University



*Los Alamos Computer Science Symposium, 13 October 2009*

# The JHU Public Turbulence Database

Eric Perlman<sup>2</sup>, Minping Wan<sup>1</sup>, Yi Li<sup>1</sup>, Yunke Yang<sup>1</sup>,  
Huidan Yu<sup>1</sup>, Jason Graham<sup>1</sup>, Kalin Kanov<sup>1</sup>,  
Randal Burns<sup>2</sup>, Shiyi Chen<sup>1</sup>, Gregory Eyink<sup>4</sup>,  
Charles Meneveau<sup>1</sup>, Alex Szalay<sup>3</sup>

(1) Mechanical Engineering, (2) Computer Science, (3) Physics and Astronomy, (4) Applied Mathematics & Statistics.

Significant help and support:

Jan Vandenberg<sup>3</sup>, Alainna White<sup>3</sup>, Tms Budvari<sup>3</sup>, Ed Givelberg<sup>3</sup>, Xiaodan Wang<sup>1</sup>

Funding: National Science Foundation (Hecura, CDI, ITR, and MRI),  
Microsoft, Keck Foundation, Gordon and Betty Moore Foundation



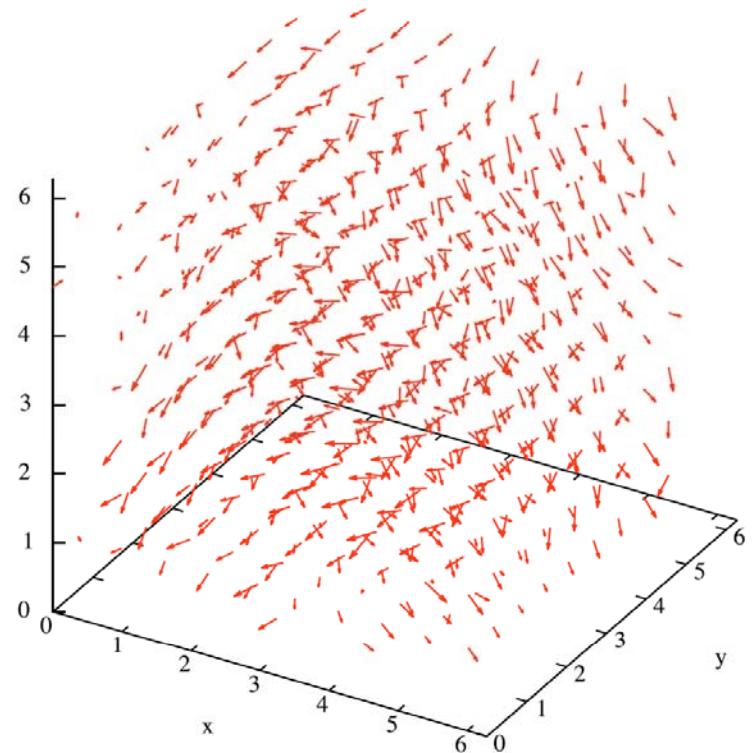
# Overview

- Description of the TurbulenceDB
  - As an example of a JHU/IDIES data intensive architecture
  - We support several others
    - Sloan Digital Sky Survey
    - PanSTARRs
    - Life Under Your Feet (sensor network for soil ecology)
    - Chesapeake Bay Environmental Observatory (environmental data fusion)
- Landscape of data intensive computing (at Universities)
  - Power, density, Amdahl-balanced systems
  - Workload characterization
- Evolution of data intensive architectures
  - Stepping off the power curve
  - From faculty closets to clusters of low-power blades
- I/O challenges in TurbulenceDB
  - As time allows



# Background: Turbulence Simulations

- DNS simulations generate 10s to 100s of TBs
- Traditional ways to interact with data:
  - Analyze dynamics on the fly during simulation
  - Store and analyze selected snapshots on desktops machines
- If time-evolution needed or unforeseen questions arise
  - Redo simulation
  - Keep large data sets to reload onto HPC facilities
  - Non-local users: ship hard disks, but they still need HPC resources



# Accessibility

*“very large simulations remain out of reach of most”*

The problem will not automatically get better--even if wires get faster, size of “top-ranked simulations” growing even faster: *i.e. without changing current approach, top-ranked simulations will be accessible only to a shrinking subset of the scientific community.*



# The TurbulenceDB Approach

Build databases of the complete space-time history of high-resolution multi-scale simulations for:

- Ad-hoc inspection and casual use
- Data mining and feature extraction (landmark database)
- Public access
- Retrospective studies, repeatability, and archival

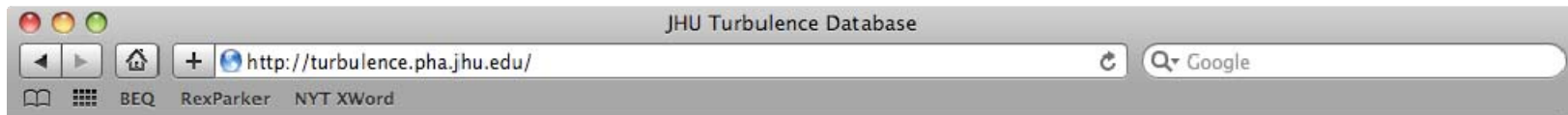
Databases preserve computational effort

- Separate simulation (solving system) from experiment
- Repeat experiments without repeating computation
- Make high-resolution data available outside HPC

Enable new classes of applications

- That iterate back and forth through time
- That examine large space-time spans





# The JHU Turbulence Database Cluster

*turbulence.pha.jhu.edu*

[Dataset descriptions](#)

[Instructions to access via Web services interface](#)  
(using Fortran, C or Matlab running on client computer)

[Access via web-browser](#)

[Analysis tools documentation](#)  
(interpolation algorithms, differentiation, etc...)

[Publications](#)

[People and credits](#)

[Citing the database in your work](#)

[Related group webpages](#)

[Links to other turbulence and fluid dynamics datasets](#)

[Home](#)

## Welcome to the JHU Turbulence Database Cluster (TDC.v1) site

This website is a portal that enables access to multi-Terabyte turbulence databases. The data reside on several nodes and disks on our database cluster computer and are stored in small 3D subcubes. Positions are indexed using a Z-curve for efficient access.

Access to the data is facilitated by a Web services interface that permits numerical experiments to be run across the Internet. We offer C and Fortran interfaces layered above [Web services](#) so that scientists can use familiar programming tools on their client platforms (MATLAB under development). Calls to fetch subsets of the data can be made directly from within a program being executed on the client's platform. [Manual queries](#) for data at individual points and times via web-browser are also supported. Evaluation of velocity and pressure at arbitrary points and time is supported using interpolations executed on the database nodes. Spatial differentiation using various order approximations (up to 8th order) are also supported (for details, see [documentation page](#)). Other functions such as spatial filtering are being developed.

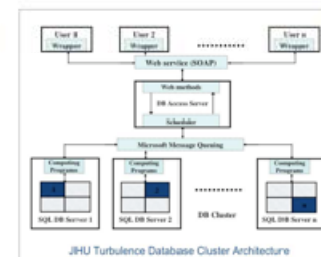
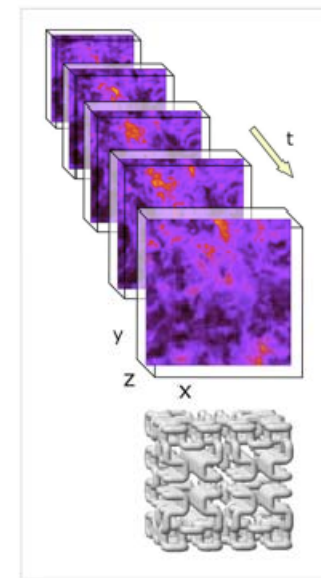
So far the database contains a  $1024^4$  space-time history of a direct numerical simulation of isotropic turbulent flow, in incompressible fluid in 3D. The simulation was performed using 1024 grid points in each direction using a pseudo-spectral method, and forcing at large scales. The database allows access to 1024 time steps covering about one integral turn-over time-scale of the turbulence. The dataset comprises 27 Terabytes of data. Basic characteristics of the data sets can be found in the [dataset description page](#). Technical details about the database techniques used for this project are described in the [publications](#).

The Turbulence Database Cluster project is funded by the US [National Science Foundation](#).

Questions and comments? [turbulence@jhu.edu](mailto:turbulence@jhu.edu)

**26693135374 points queried**

Please excuse our dust as we continue to develop this site. The Turbulence Database is on-line but may periodically be unavailable as we continue to add functionalities.

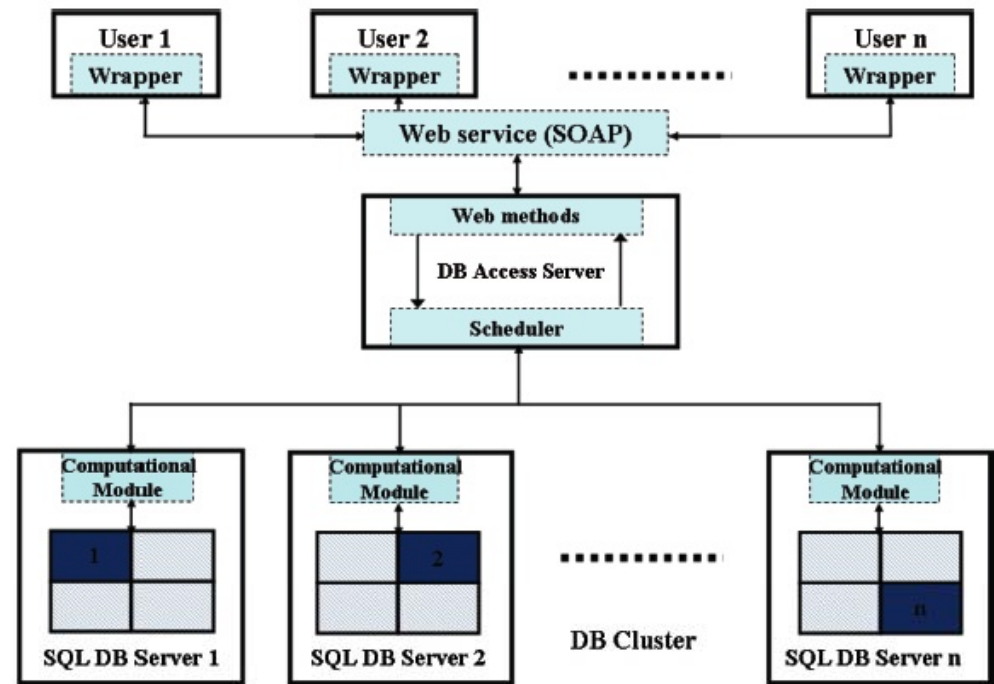




# High-Performance Web Services

Build data warehouses according to Gray's laws

- Bring the computation to the data
  - Using active database features, such as user-defined functions
  - Avoid transferring large amounts of data across networks
- Scale out, not scale up
  - Rely on inexpensive commodity hardware
- Use lightweight enterprise-standard middleware
  - WSDL and SOAP
  - Integrates with Fortran, MATLAB, and R





# Data Set #1:

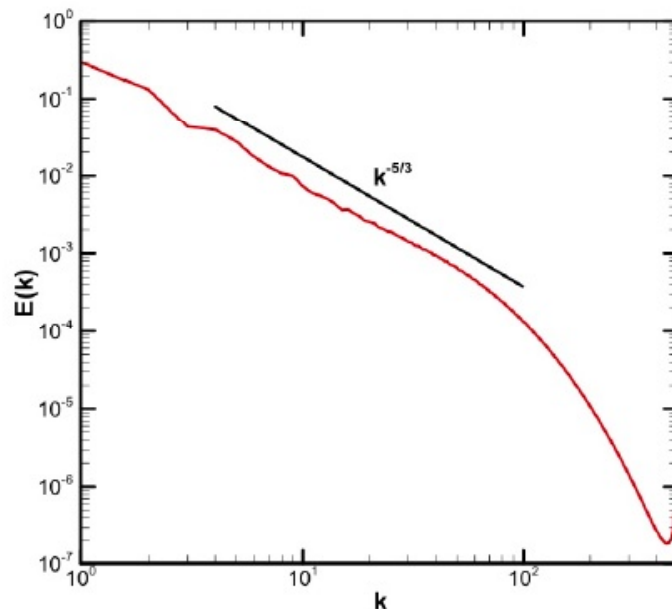
## DNS of forced isotropic turbulence

(standard pseudo-spectral)

$1024^4$  space-time history

16  $\rightarrow$  27 TBytes

$Re_\lambda \sim 430$



### Simulation parameters:

Domain:  $2\pi \times 2\pi \times 2\pi$  (i.e. range of  $x_1, x_2$  and  $x_3$  is  $[0, 2\pi]$ )

Grid:  $1024^3$

Viscosity ( $\nu$ ) = 0.000185

Simulation time-step  $\Delta t = 0.0002$

Data are stored separated by  $\delta t = 0.002$  (i.e. every 10 DNS time-steps is stored)

Time stored: between  $t=0$  and 2.048 (1024 time samples separated by  $\delta t$ )

### Statistical characteristics of turbulence, time averaged over $t=0$ and 2.048:

Total kinetic energy,  $E_{tot} = \left\langle \sum_k \frac{1}{2} \hat{\mathbf{u}} \cdot \hat{\mathbf{u}}^* \right\rangle_{time}$  :  $E_{tot} = 0.695$

Dissipation,  $\varepsilon = \left\langle \sum_k (\nu k^2 \hat{\mathbf{u}} \cdot \hat{\mathbf{u}}^*) \right\rangle_{time}$  :  $\varepsilon = 0.0928$

Rms velocity,  $u' = \sqrt{(2/3)E_k}$  :  $u' = 0.681$

Taylor Micro. Scale  $\lambda = \sqrt{15\nu u'^2 / \varepsilon}$  :  $\lambda = 0.118$

Taylor-scale Reynolds #,  $Re_\lambda = u\lambda / \nu$  :  $Re_\lambda = 433$

Kolmogorov time scale  $\tau_\eta = \sqrt{\nu / \varepsilon}$  :  $\tau_K = 0.0446$

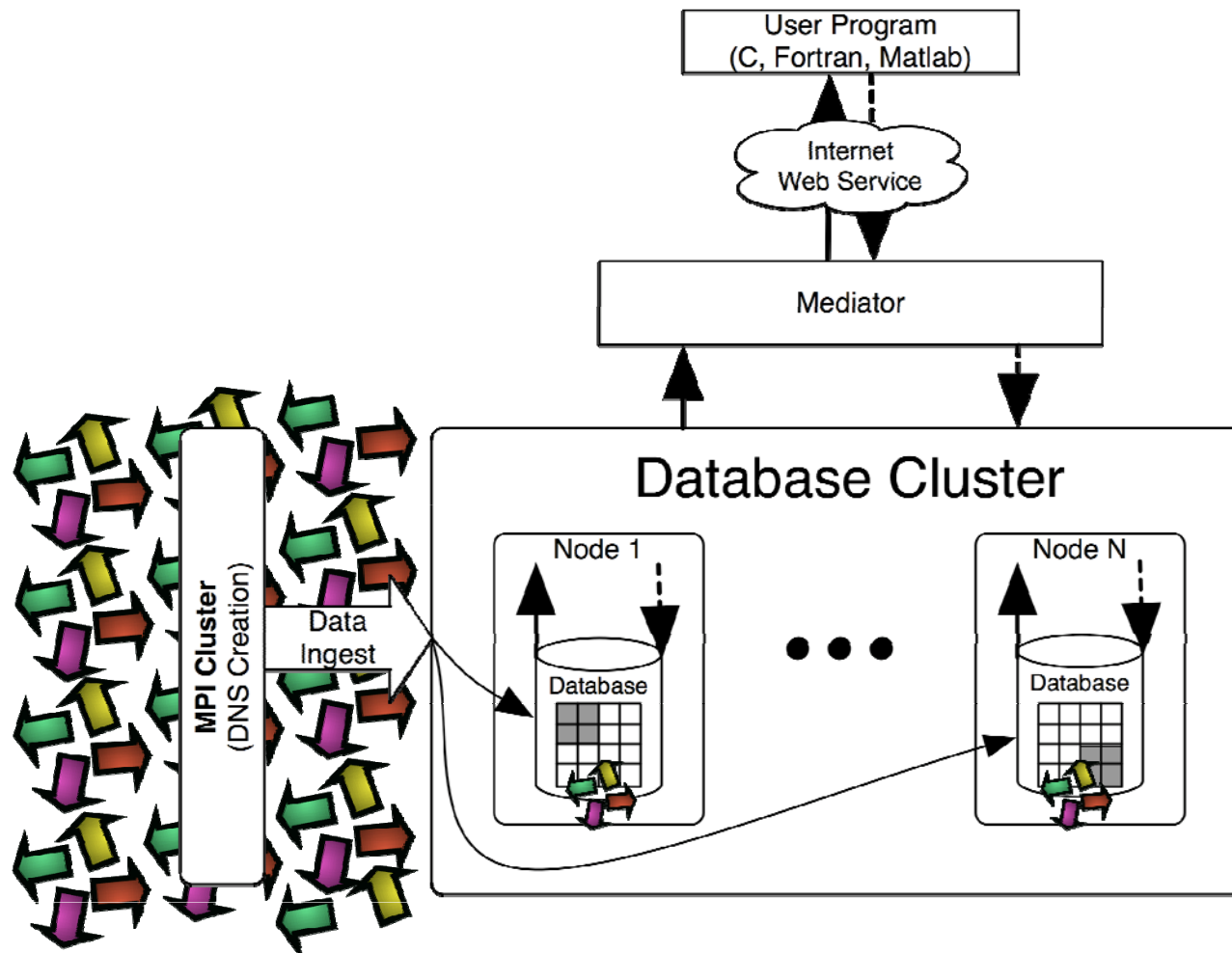
Kolmogorov length scale  $\eta = \nu^{3/4} \varepsilon^{-1/4}$  :  $\eta = 0.00287$

Integral scale:  $L = \frac{\pi}{2u'^2} \int \frac{E(k)}{k} dk$  :  $L = 1.376$

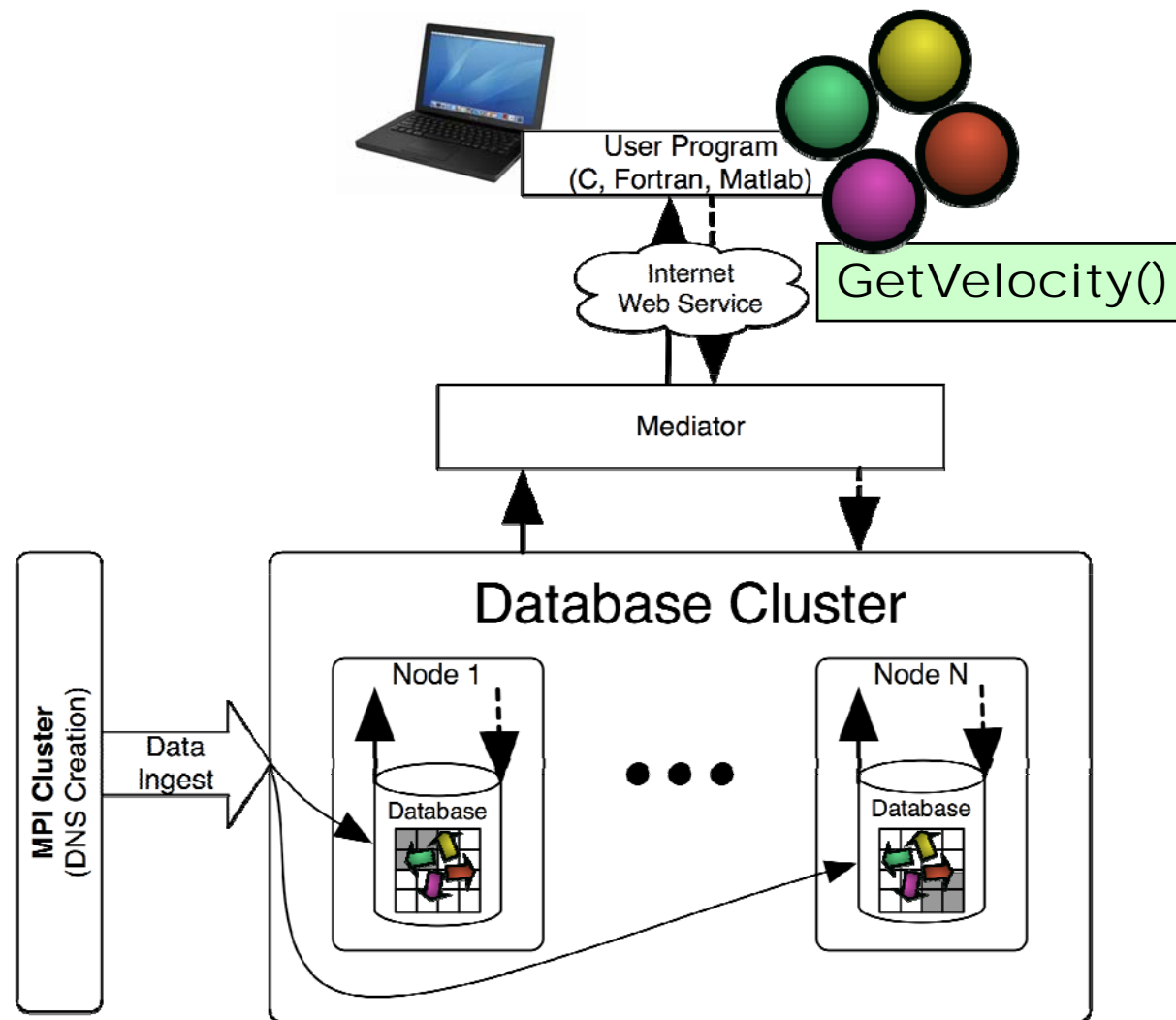
Large eddy turnover time:  $T_L = L / u'$  :  $T_L = 2.02$



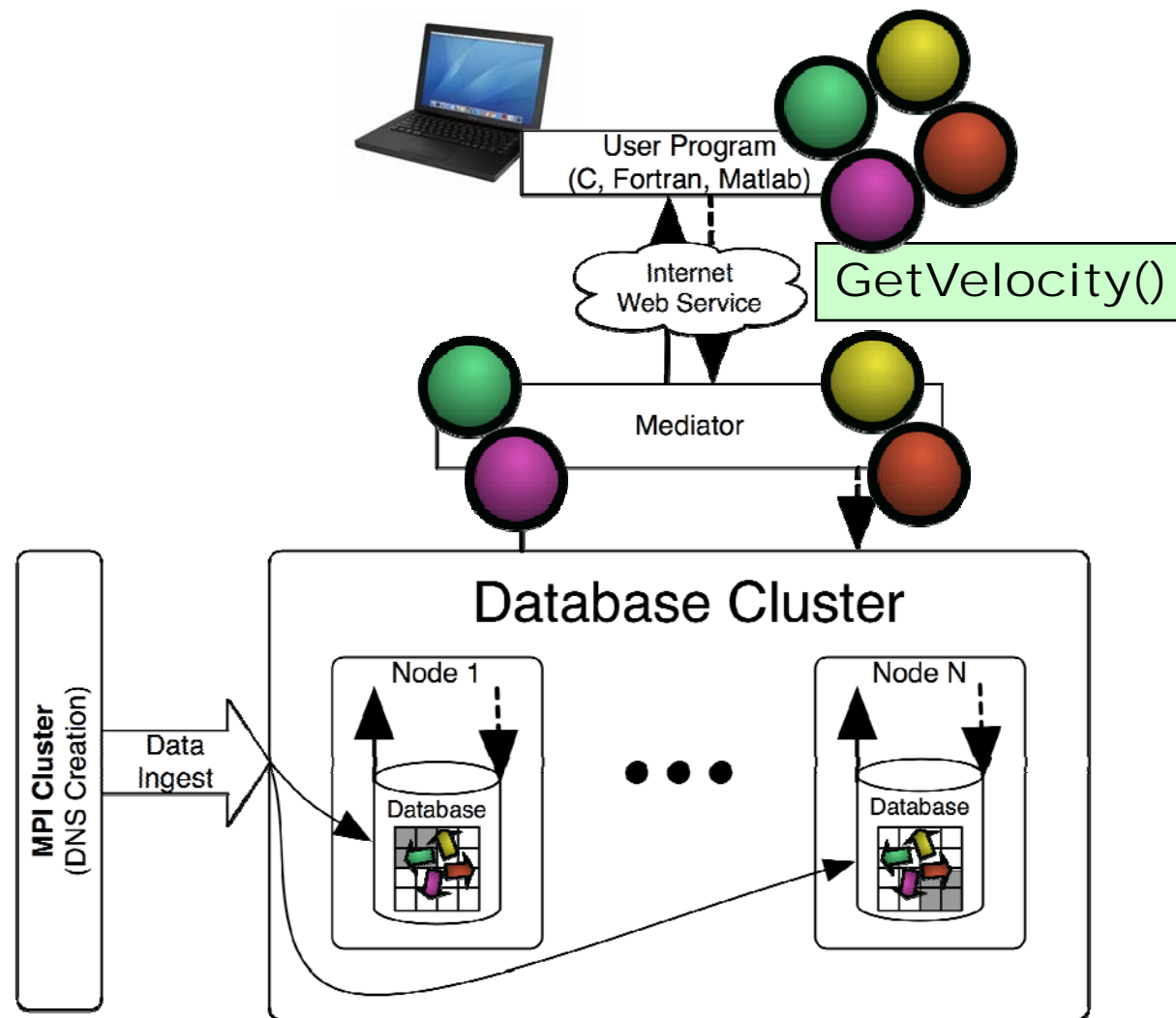
# Data Generation and Ingest



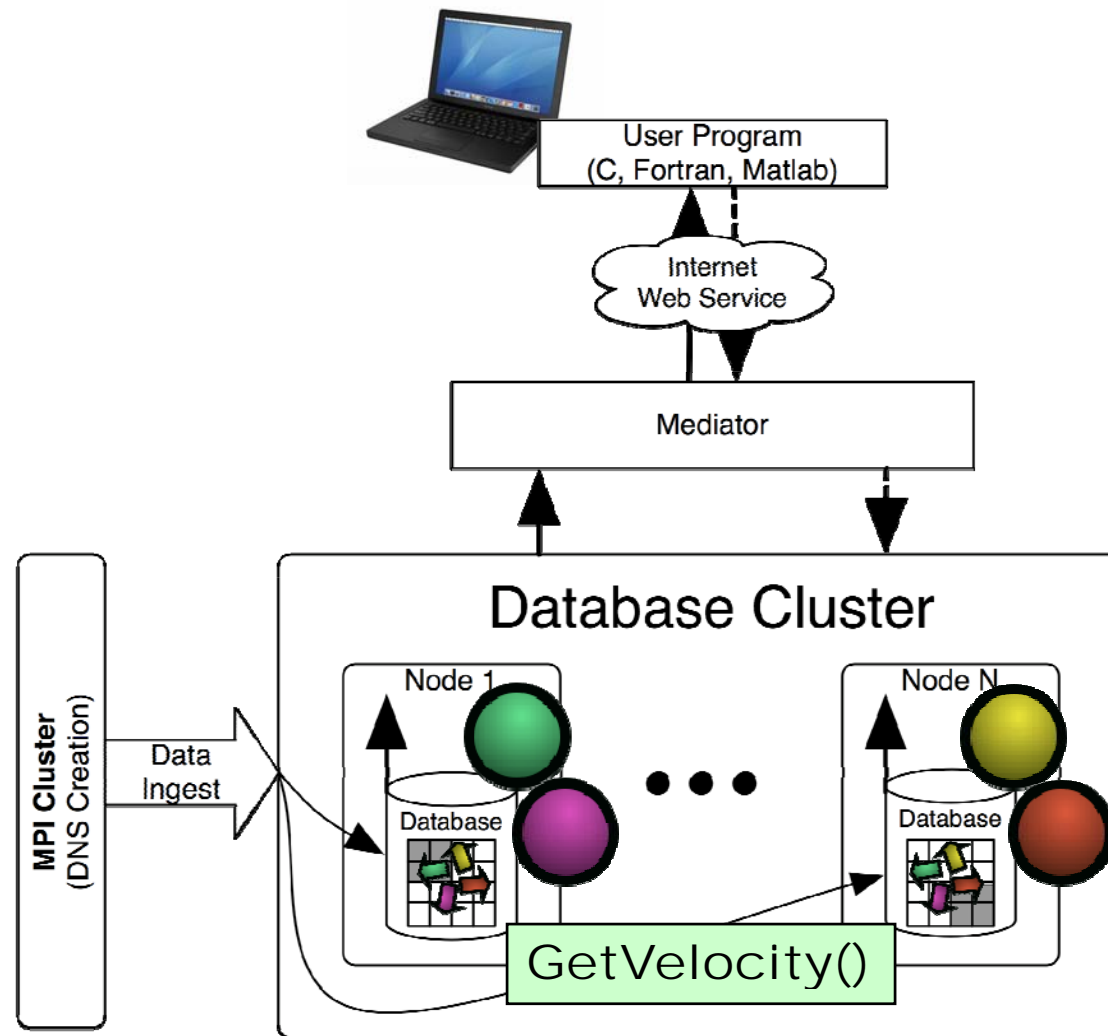
# GetVelocity() Web service



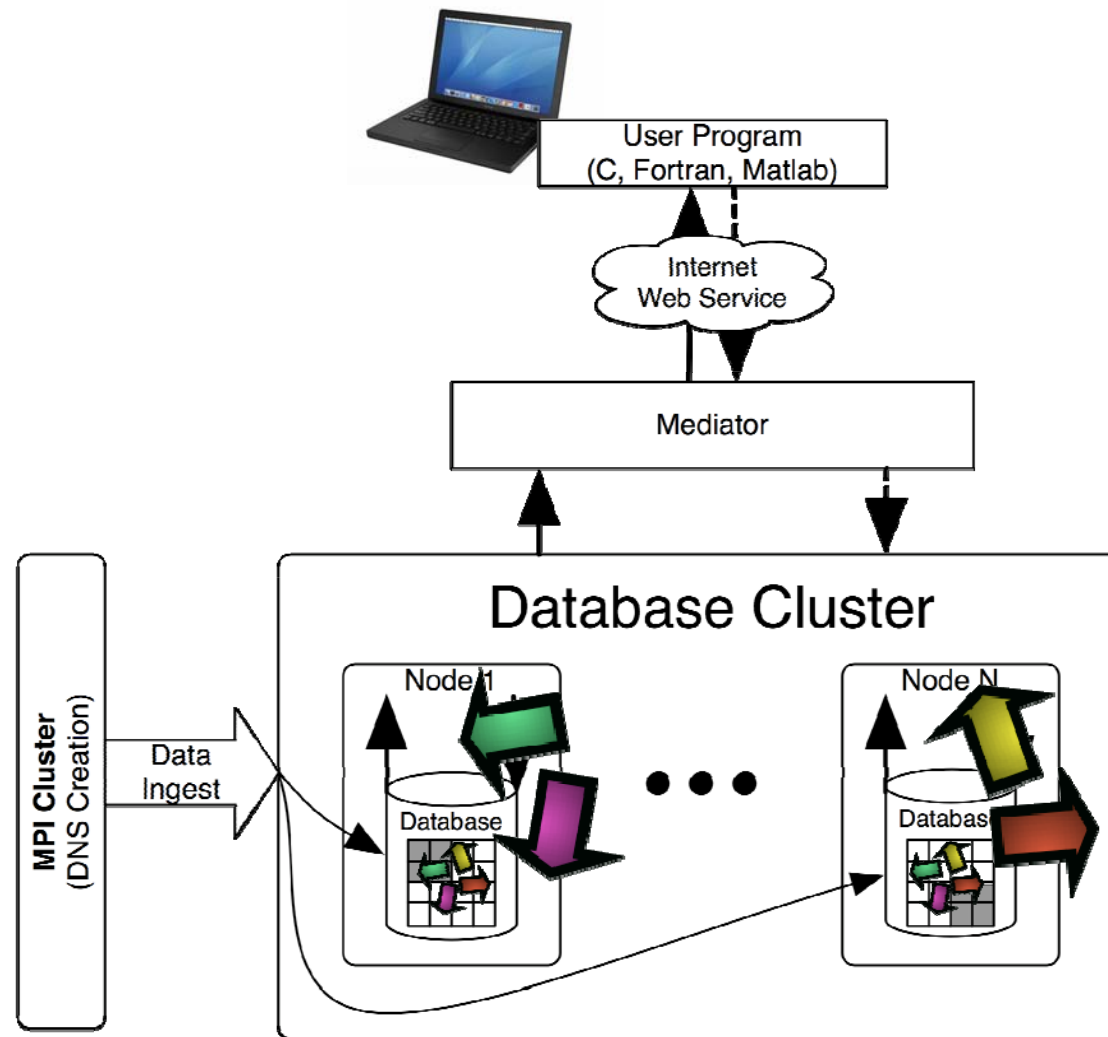
# Mediator divides workload spatially



# Request dispatched to databases

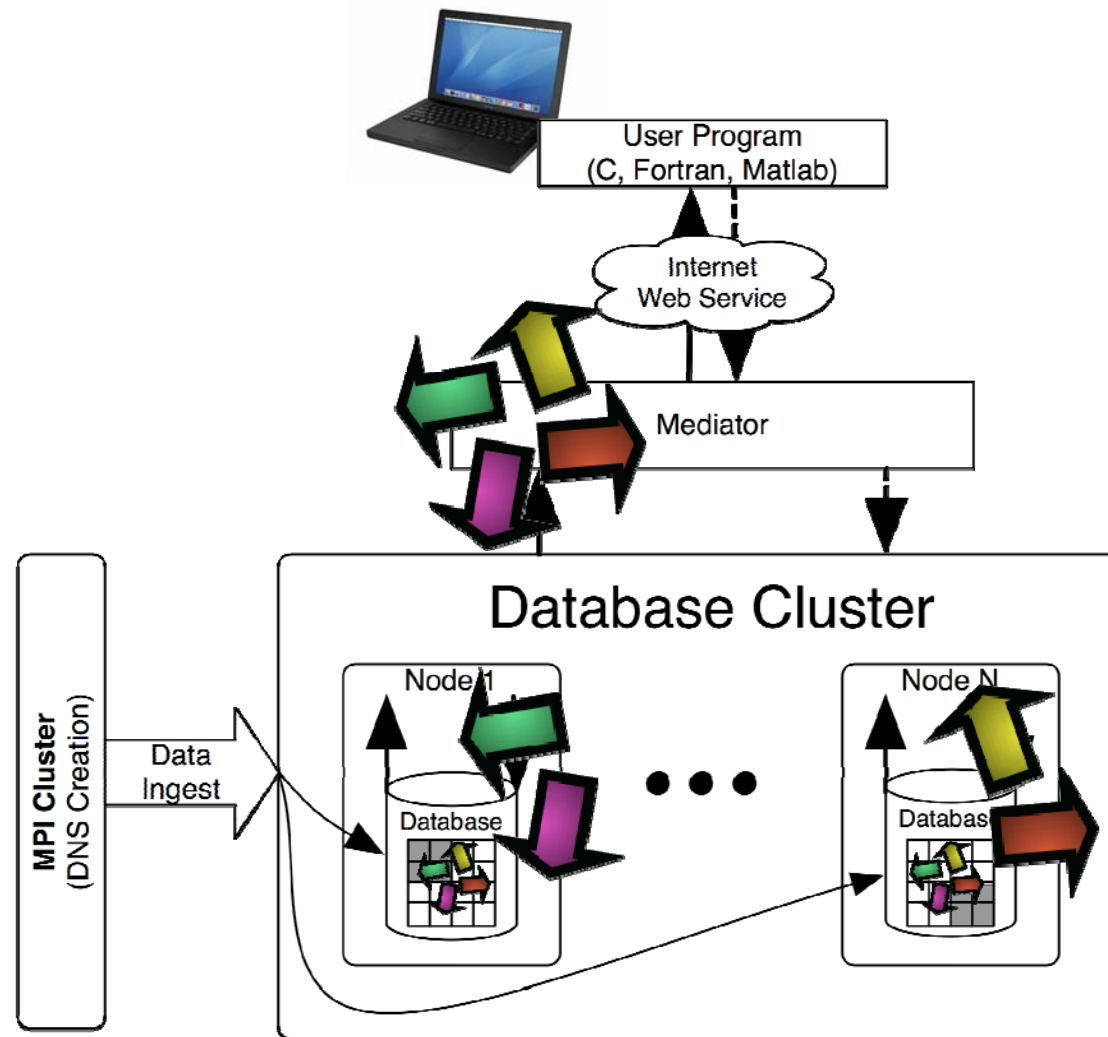


# Velocities are returned

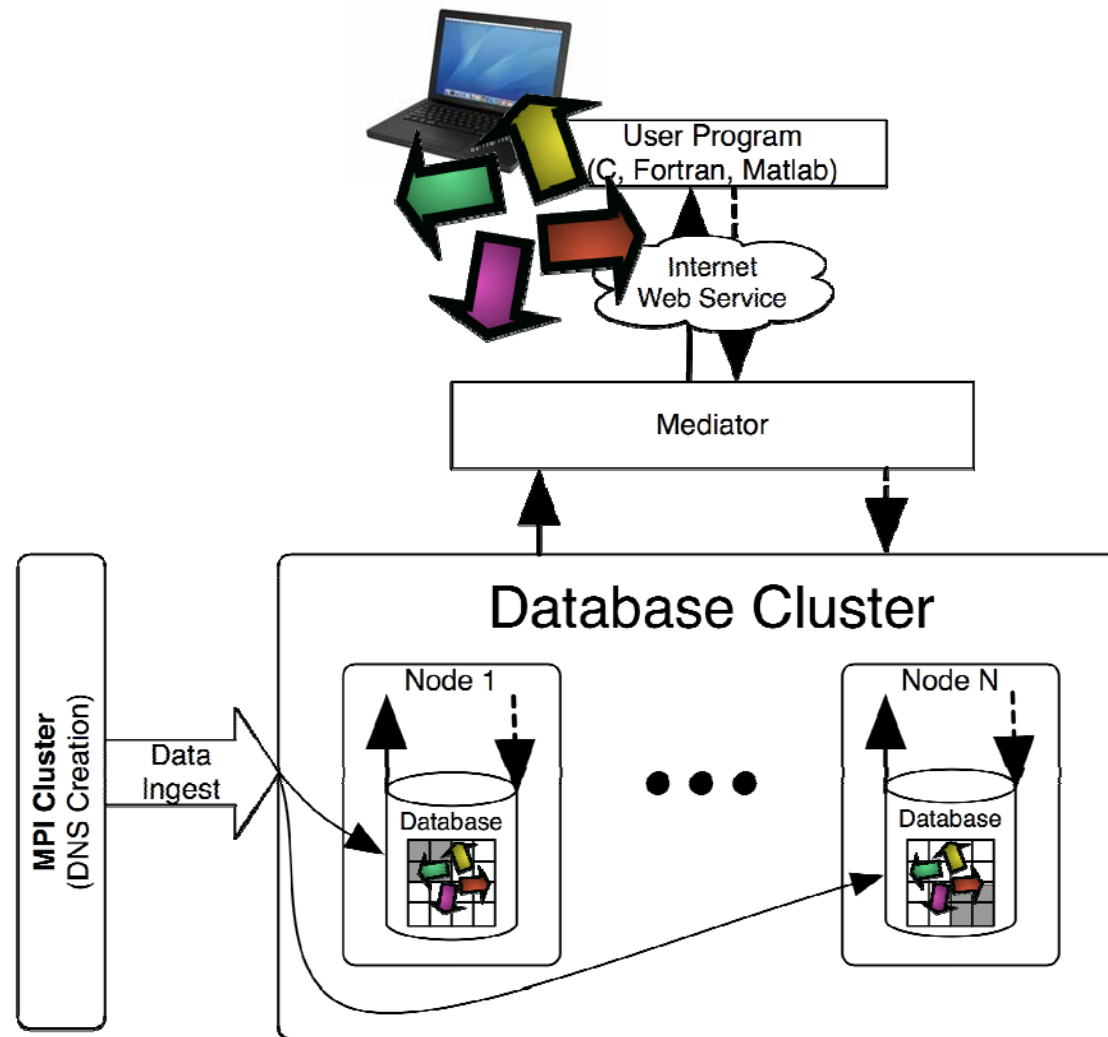




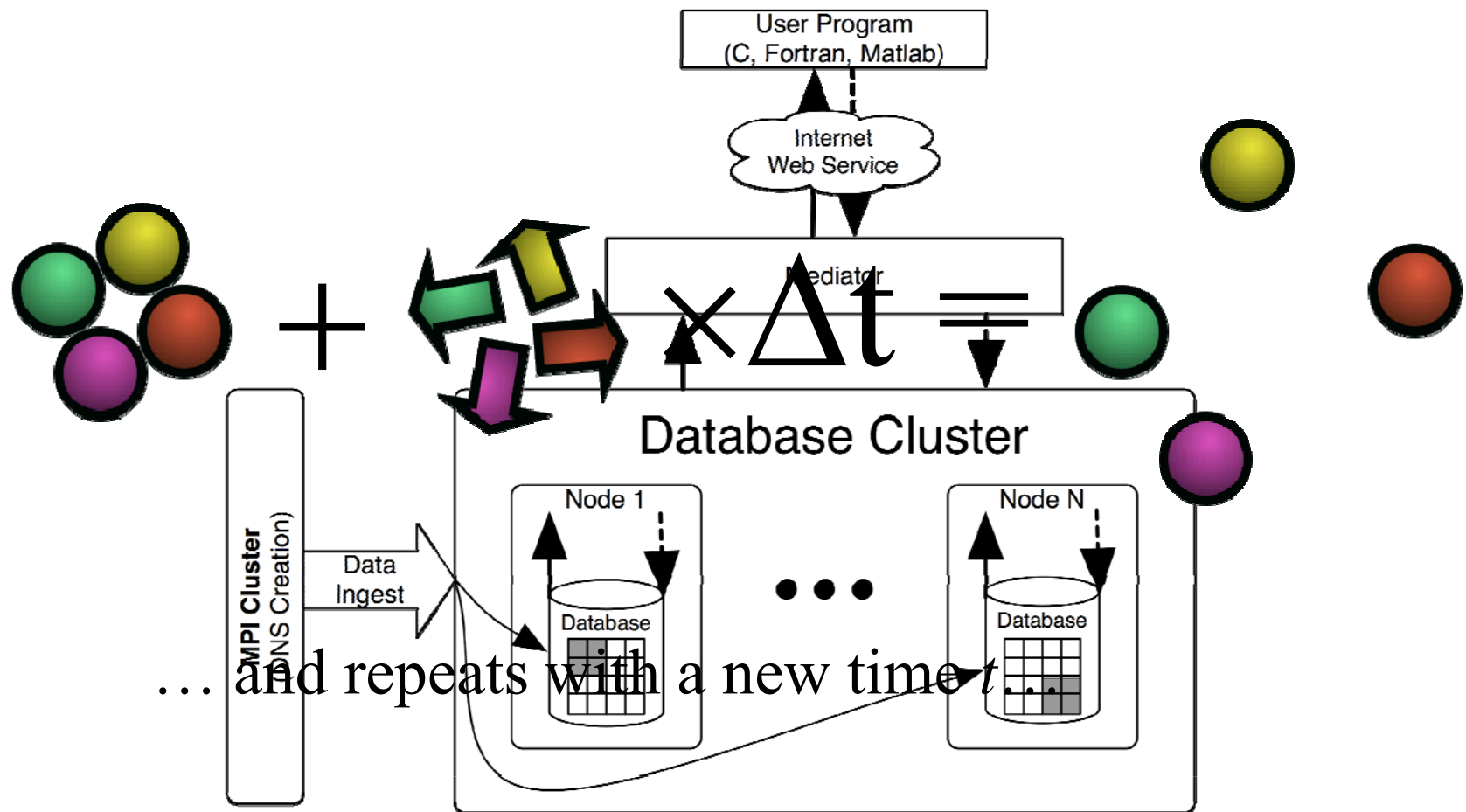
# Collated by Mediator



# ...and returned to the User



# Client determines particle tracks



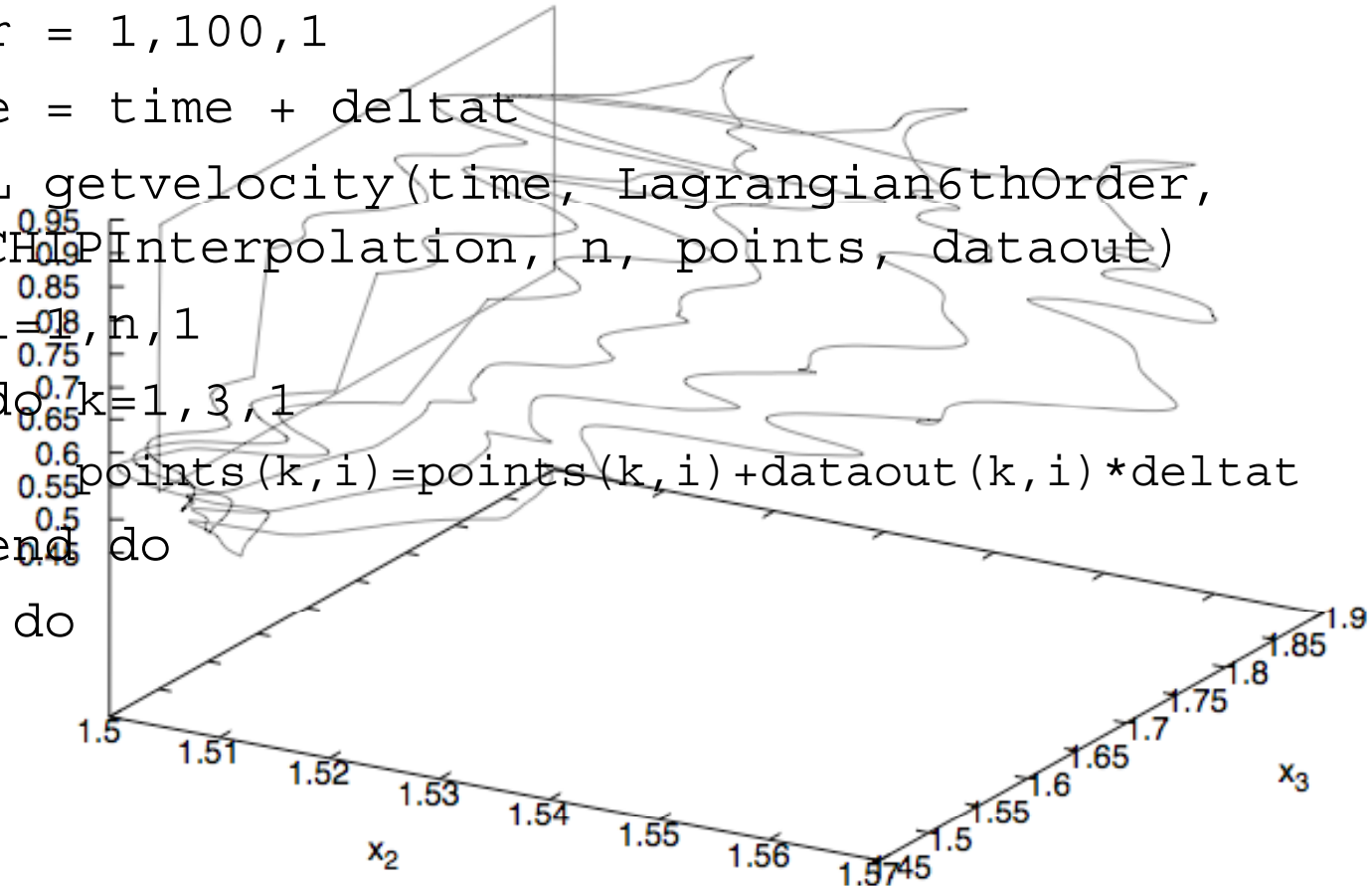
# Defining Interfaces

- Low-level interfaces are inefficient
  - E.g. get velocity at point
  - Provide few opportunities for optimization, batch operations, request reordering, bulk data transfer
- High-level interfaces are restrictive
  - E.g. track 1M particles through 1K timesteps
  - Allow for little customization or transparency/interactivity
  - Requires a new Web service for each new experiment
- Middle ground: request batches of data points with server-side space/time interpolation, gradients, etc.
  - Perform common compute intensive tasks at server
  - I/O and scheduling optimizations possible
  - But, client code customizes experiment (e.g., particle mass)

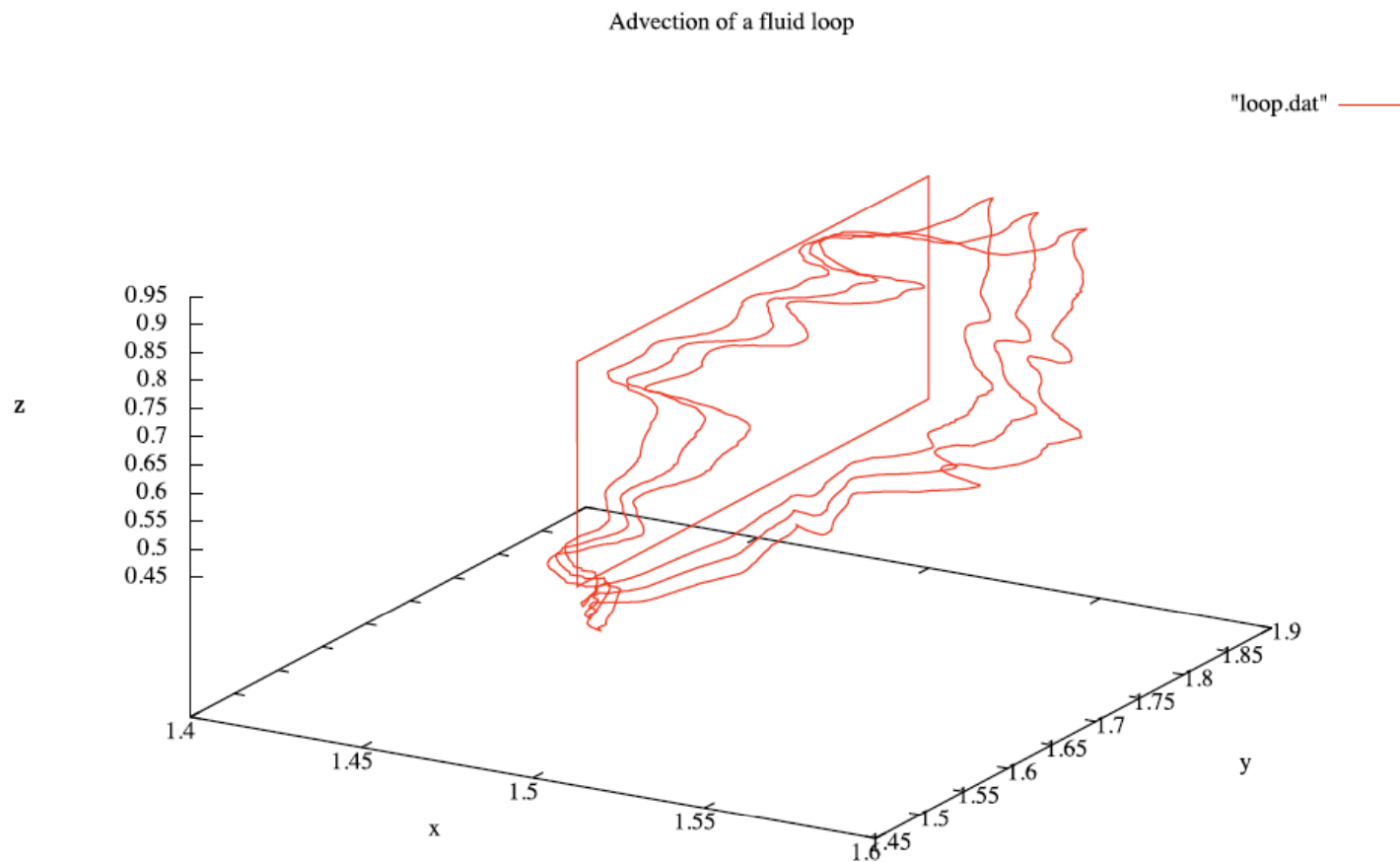


# Demo of Particle Tracking

```
do iter = 1,100,1
  time = time + deltat
  CALL getvelocity(time, Lagrangian6thOrder,
    PCHIPInterpolation, n, points, dataout)
  do i=1,n,1
    do k=1,3,1
      points(k,i)=points(k,i)+dataout(k,i)*deltat
    end do
  end do
end do
end do
```



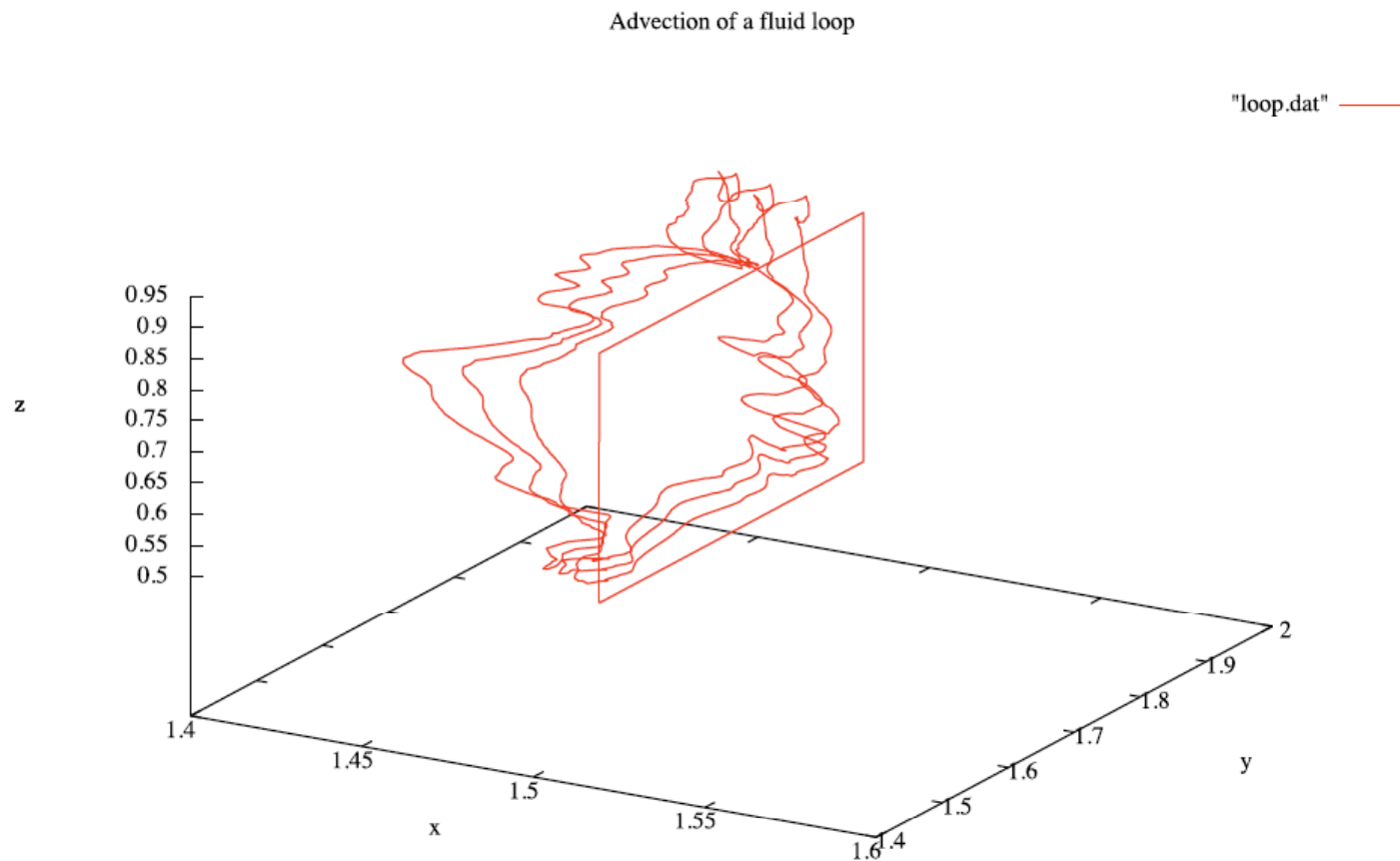
# The evolution of a shape





# ... and the pre-history

not possible during DNS simulation

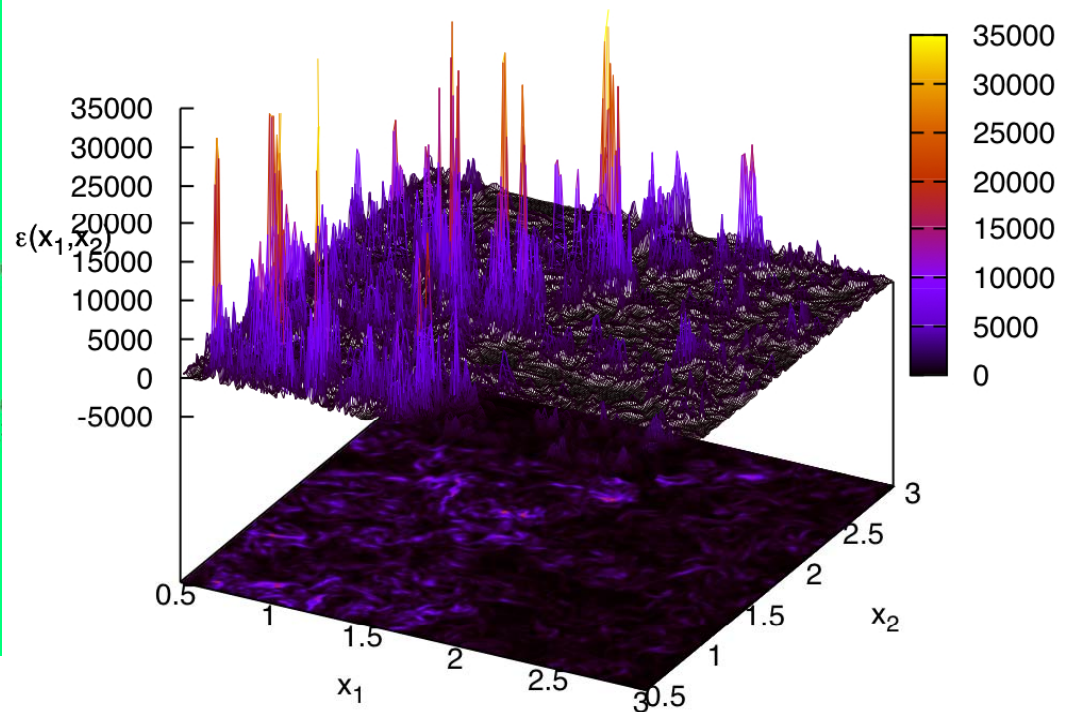


## Sample code (gfortran 90) running on this Mac (unix)

```
UNIX Terminal — tcsh — ttys000

if (icase.eq.4) then
  open (50,file='dissip.dat',status='unknown')
!
  write(*,*) 'A test of the JHU Turbulence Database'
  write(*,*) 'extract sub-plane of velocities and dissipation for contour plot'
!
  time = 0.01
  n=250
!
  do i=1,n
    xx(i)=0.5+2.5*i/n
    yy(i)=0.5+2.5*i/n
  end do
!
  do i=1,n
    do j=1,n
      ii=(i-1)*n+j
      points(1,ii)=xx(i)
      points(2,ii)=yy(j)
      points(3,ii)=1.7
    end do
  end do
!
  CALL getvelocitygradient(authkey, dataset1, time, Lagrangian)
!
  do i=1,n
    do j=1,n
      ii=(i-1)*n+j
      vf=2.*(dataout9(2,ii)+dataout9(4,ii))*2.+2.*(dataout9(6,ii)-dataout9(8,ii))*2.+dataout9(10,ii)
      write(50,*) points(1,ii),points(2,ii),vf
    end do
    write(50,*) " "
  end do
!
endif
```

Get velocity gradients on a plane  
and evaluate dissipation



# The University Data Intensive Landscape

- Scientific (and other) data double every year
- Trend driven by
  - Inexpensive sensors
  - Increased storage density
- More data-intensive scalable architectures needed
- Most scientific data analysis done on small to midsize BeoWulf clusters, from faculty startup
- Universities hitting the “power wall”
- **Not scalable, not maintainable...**
- *How to build a scalable, data-intensive architecture?*



# Amdahl's Laws

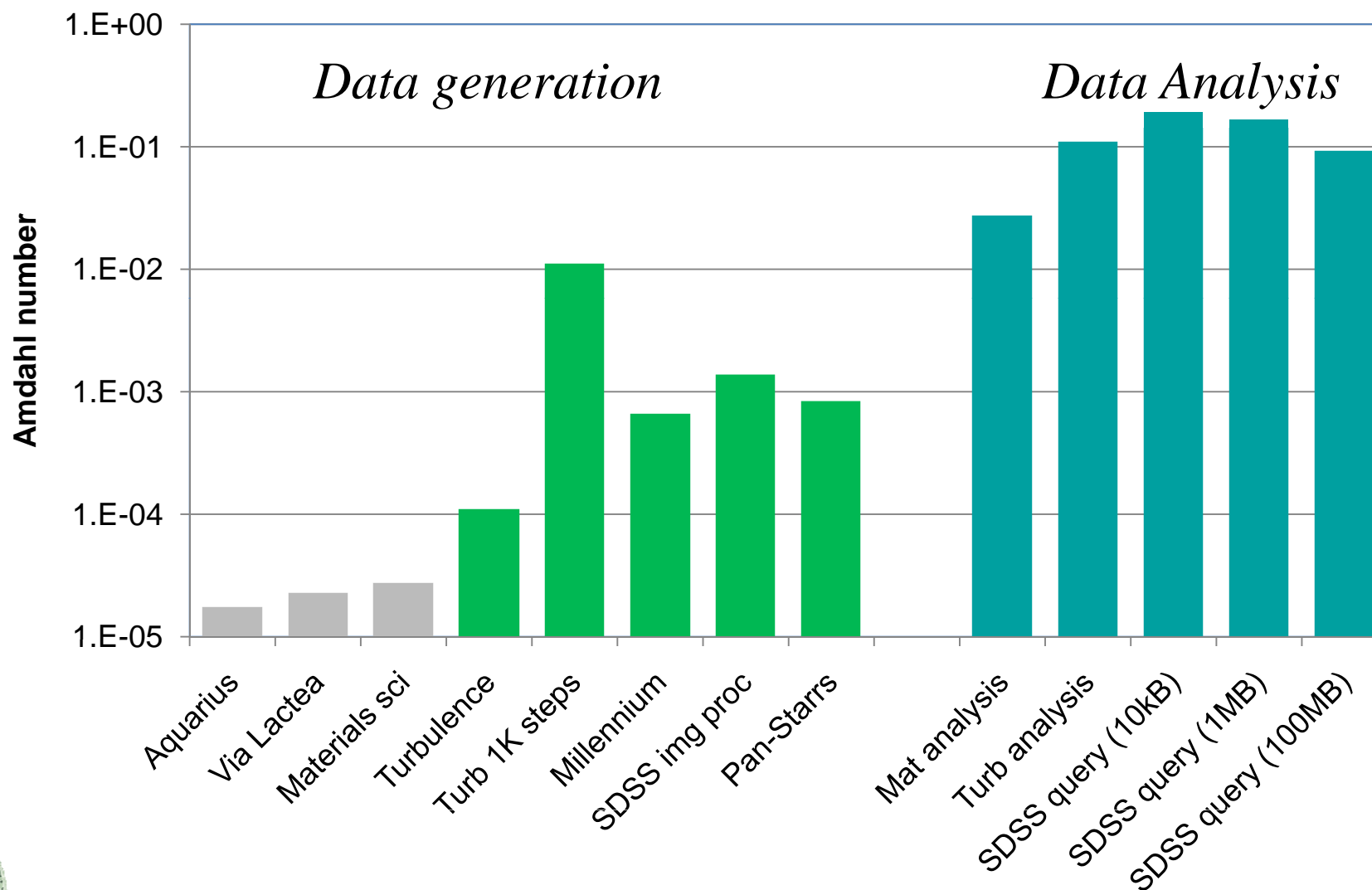
Gene Amdahl (1965): Laws for a balanced system

- i. Parallelism: max speedup is  $S/(S+P)$
- ii. **One bit of IO/sec per instruction/sec (BW)**
- iii. One byte of memory per one instruction/sec (MEM)

Modern multi-core systems move farther  
away from Amdahl's Laws  
(Bell, Gray and Szalay 2006)



# Amdahl Numbers for Data Sets



# Typical Amdahl Numbers

- National infrastructure focused on CPU cycles
- Even HPC projects choking on I/O
- Sociology:
  - Data collection in larger collaborations
  - Analysis decoupled, from data archived by smaller groups

System	CPU count	GIPS [GHz]	RAM [GB]	diskIO [MB/s]	Amdahl	
					RAM	IO
BeoWulf	100	300	200	3000	0.67	0.08
Desktop	2	6	4	150	0.67	0.2
Cloud VM	1	3	4	30	1.33	0.08
SC1	212992	150000	18600	16900	0.12	0.001
SC2	2090	5000	8260	4700	1.65	0.008
GrayWulf	416	1107	1152	70000	1.04	0.506



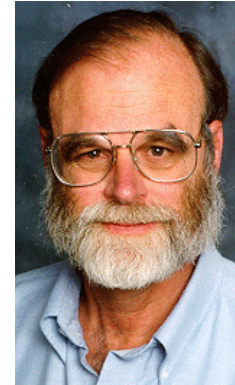


# Architecture v1 (Commodity Closet)

- Commodity cluster (4 nodes)
  - 2x Quad-core Intel Xeon 2.33GHz, 8G RAM
  - 12, 750 GB SATA drives per node
- Amdahl I/O number = 0.70
  - Processors:  $2 \text{ GHz} * 4 \text{ cores} = 2^{33} \text{ cycles per sec}$
  - I/O:  $12 \text{ spindles} * 60 \text{ MB/s} \sim 2^{33} \text{ bits/sec}$
- Simple configuration for data-intensive computing
  - Result is an Amdahl balanced system
  - Storage density dictates that if I/O can keep up, then we have sufficient capacity for Turbulence data



# Architecture v2

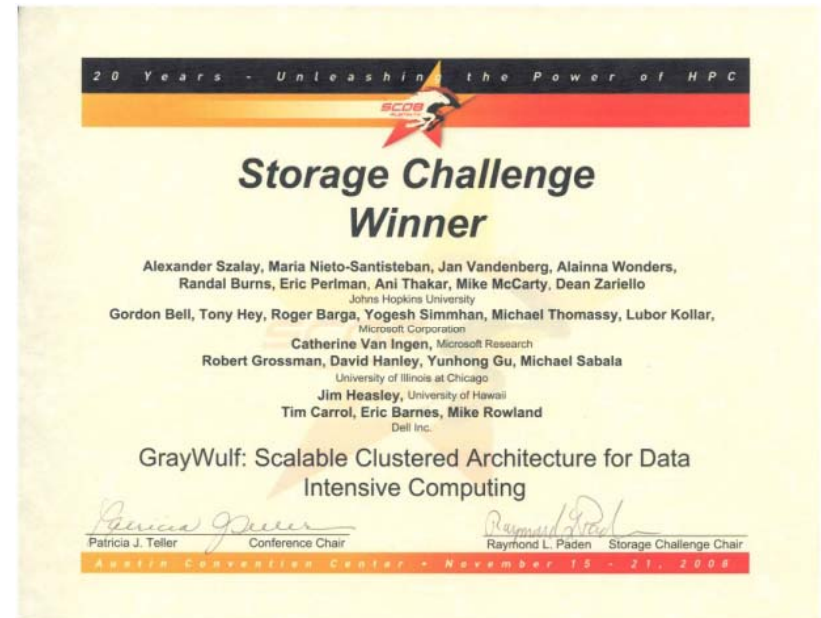


- Implement Jim Gray's vision of data-intensive, scale-out computing
  - High Amdahl number ( $>0.5$ )
- Distributed SQLServer cluster/cloud
  - 1.1PB disk, 500 CPUs
  - Connected with 20 Gbit/sec Infiniband
  - Linked to 1500 core compute cluster
  - 10 GB lambda uplink to UIC
- Dedicated to eScience, provide publicly-accessible Web services
- Funded by Moore Foundation, Microsoft and Pan-STARRS



# GrayWulf Performance

- Demonstrated large scale computations involving ~200TB of DB data (won SC08 Storage Challenge)
  - DB speeds close to “speed of light” (72%)
- Scale-out over SQL Server cluster
  - 70GB/s for 46 nodes from <\$700K
- Very cost efficient: \$10K/GBps
- Amdahl number: 0.56
- **But: hitting the “power wall”!!!!**



# Cyberbricks/Amdahl Blades

- **Scale down** the CPUs to the disks!
  - Solid State Disks (SSDs)
  - 1 low power CPU per SSD
- Current SSD parameters
  - OCZ Vertex 120GB, 250MB/s read, 10,000 IOPS, \$300
  - Power consumption 0.2W idle, 1-2W under load
- Low power motherboards
  - Intel dual Atom N330 + ION chipset 28W at 1.6GHz
- Combination is perfect Amdahl blade
  - $200\text{MB/s} = 1.6\text{Gbits/s} \Leftrightarrow 1.6\text{GHz of Atom}$



# Building a Low Power Cluster

*Szalay, Bell, Huang, Terzis, White (HotPower09 paper):*

*Evaluation of many different motherboard + SSD combinations*

Table 2: Basic properties of a single node for various systems considered

	CPU	mem	seqIO	randIO	disk	power	cost	relative	Amdahl numbers		
	[GHz]	[GB]	[GB/s]	[kIOPS]	[TB]	[W]	[\$]	power	seq	mem	rand
GrayWulf	21.3	24	1.500	6.0	22.5	1,150	19,253	1.000	0.56	1.13	0.014
ASUS	1.6	2	0.124	4.6	0.25	19	820	0.017	0.62	1.25	0.144
Intel	3.2	2	0.500	10.4	0.50	28	1,177	0.024	1.25	0.63	0.156
Zotac	3.2	4	0.500	10.4	0.50	30	1,189	0.026	1.25	1.25	0.163
AxiomTek	1.6	2	0.120	4.0	0.25	15	995	0.013	0.60	1.25	0.125
Alix 3C2	0.5	0.5	0.025		0.008	4	325	0.003	0.40	1.00	



# Scaling: Sweet Spot Found

Scaled to a fixed sequential read rate

system	CPU[GHz]	seqIO[GB/s]	kIOPS	disk[TB]	power[W]	cost [\$]	rel. power	nodes
GrayWulf	21.3	1.5	6	22.5	1150	19250	1.000	1
ASUS	19.4	1.5	56	3.0	230	9920	0.200	12
Intel	9.6	1.5	30	1.5	84	3530	0.073	3
Zotac	9.6	1.5	31	1.5	90	3570	0.078	3
AxiomTek	20.0	1.5	50	3.1	188	12430	0.163	12.5
Alix 3C2	30.0	1.5		0.5	240	19510	0.209	60

*Cost includes 3 years of operation plus HW*

- Scaledown and power savings overcome SSD cost
- SSDs radically alter capacity/performance ratios



# Status

- Compared many low power motherboards, SSDs
- Building 50 node cluster for under \$50K
- Zotac Atom/Ion motherboards received from NVIDIA
  - N330 dual core CPU + 4GB memory
  - 16 GPUs with integrated memory controller
  - 3 SATA ports
- Adding two OCZ Vertex drive we measure
  - 500MB/sec sequential read
  - 400MB/sec sequential write
  - 20,000 IOPS
  - 28W power consumption
- 2009 NSF HECURA Award (PI Szalay)





# Architecture Summary

- Science community starving for storage and I/O
  - Data-intensive computations as close to data as possible
- Need objective metrics for I/O systems
  - Amdahl number appears to be good match to applications
- Future in low-power, fault-tolerant architectures
  - Need to get off the curve leading to power wall
  - We propose scale-out “Amdahl Data Clouds”
  - On our way to a medium size testbed
- Real reference applications for objective metrics
  - Use large data sets for scalability studies (100TB+)  
e.g. SDSS, Pan-STARRS, Sensors, Turbulence



# TurbulenceDB: Usage Statistics

Questions and comments? [turbulence@jhu.edu](mailto:turbulence@jhu.edu)

**27093020791** points queried

Please excuse our dust as we continue to develop this site. The Turbulence Database is on-line but may periodically be unavailable as we continue to add functionalities.

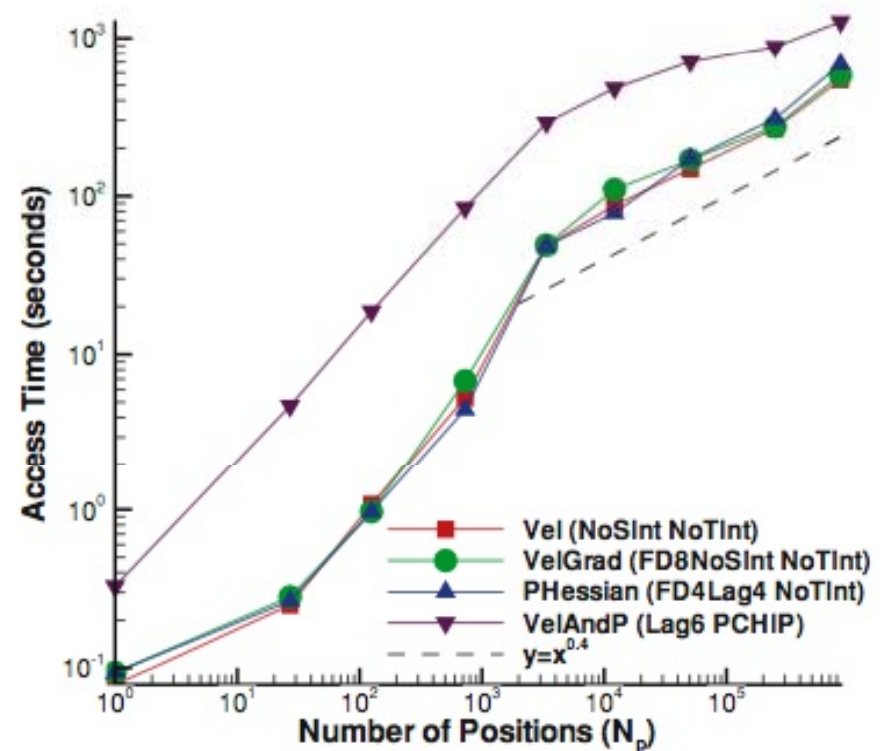
- Built our on-line user community
  - 10-12 Heavy users
  - 160 separate IP addresses.
  - Researchers without HPC facilities
  - International users
  - Educational applications



# Popularity: The Downfall of Data-Intensive Science?

- Heavy usage restricts community accessibility
  - A single user issuing a “data-intensive” session can occupy the entire system for seconds to hours!
- I/O resources need to be allocated and optimized in data-intensive clusters

Access speeds by request size  
 $N_p$  points distributed randomly in  $(0, 2\pi)^3$



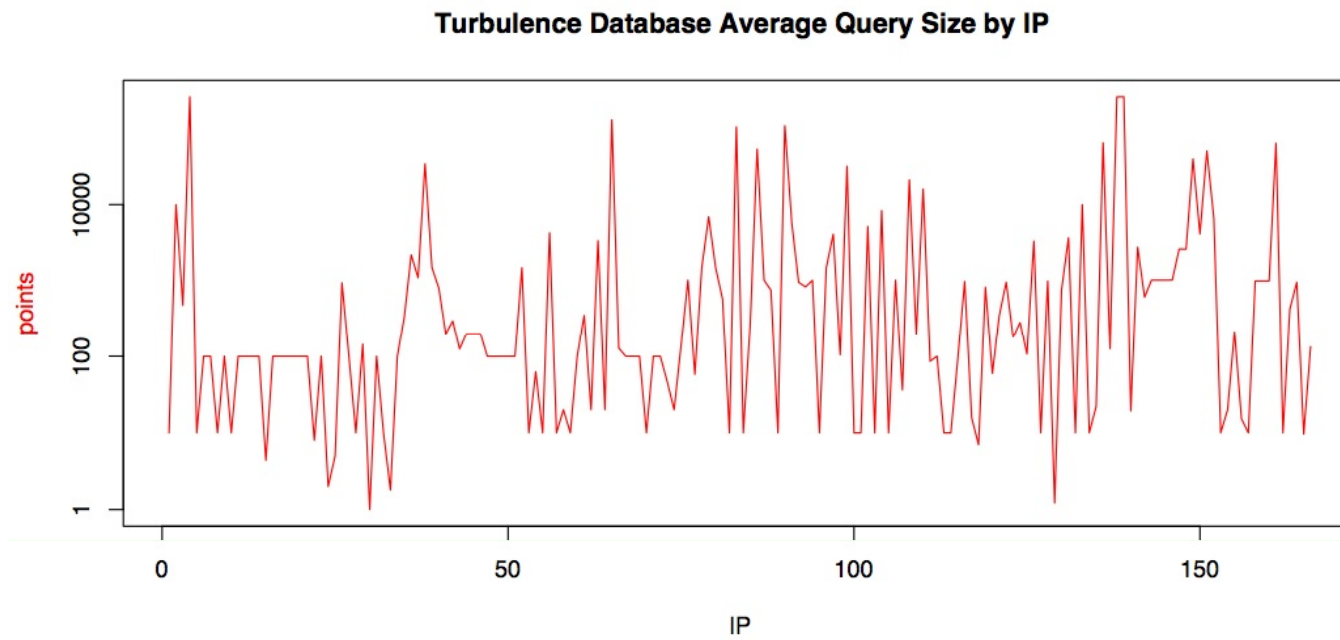
# Directions for I/O Scaling

- Data-Driven Batch Scheduling
  - I/O sharing for queries with overlapping data requirements
- Managing/Allocating I/O as a first class citizen
  - Integration with HPC scheduling
  - Balanced utilization of I/O, memory, and compute through reconfiguration, elasticity, and co-scheduling for parallel jobs
  - Another 2009 NSF HECURA grant (PI Burns)
- Replicating services and partitioning users
  - Into long-running, data-intensive sessions (for batch scheduling)
  - And casual/exploratory use (for demand scheduling)
  - We've done this for the Sloan Digital Sky Survey for 2+ years



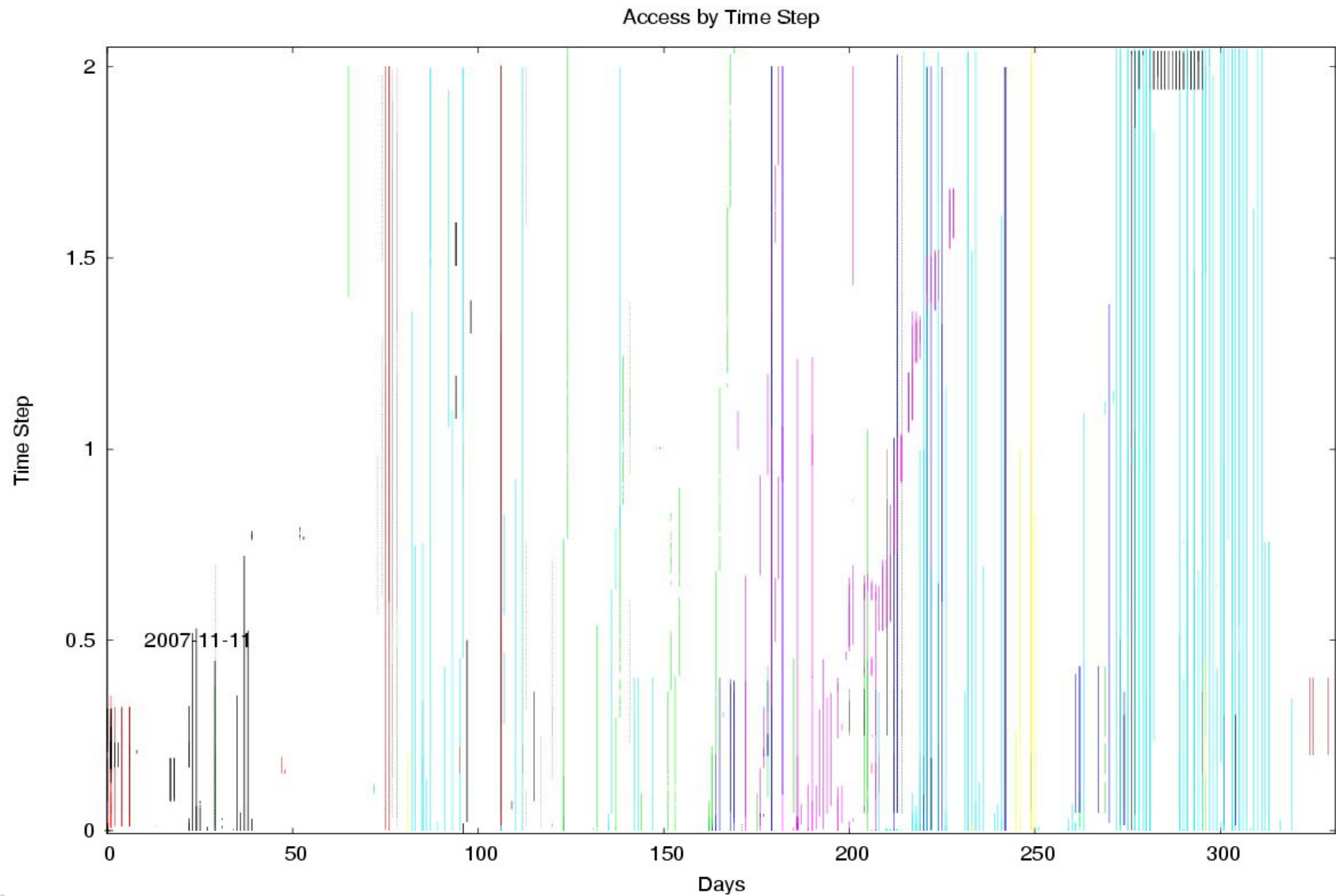
# More About TurbulenceDB Workload

- Many casual users, few intense users
- Even largest jobs request  $10^8$  points (out of  $10^{12}$  total)
- But, large jobs have spatial and temporal commonality



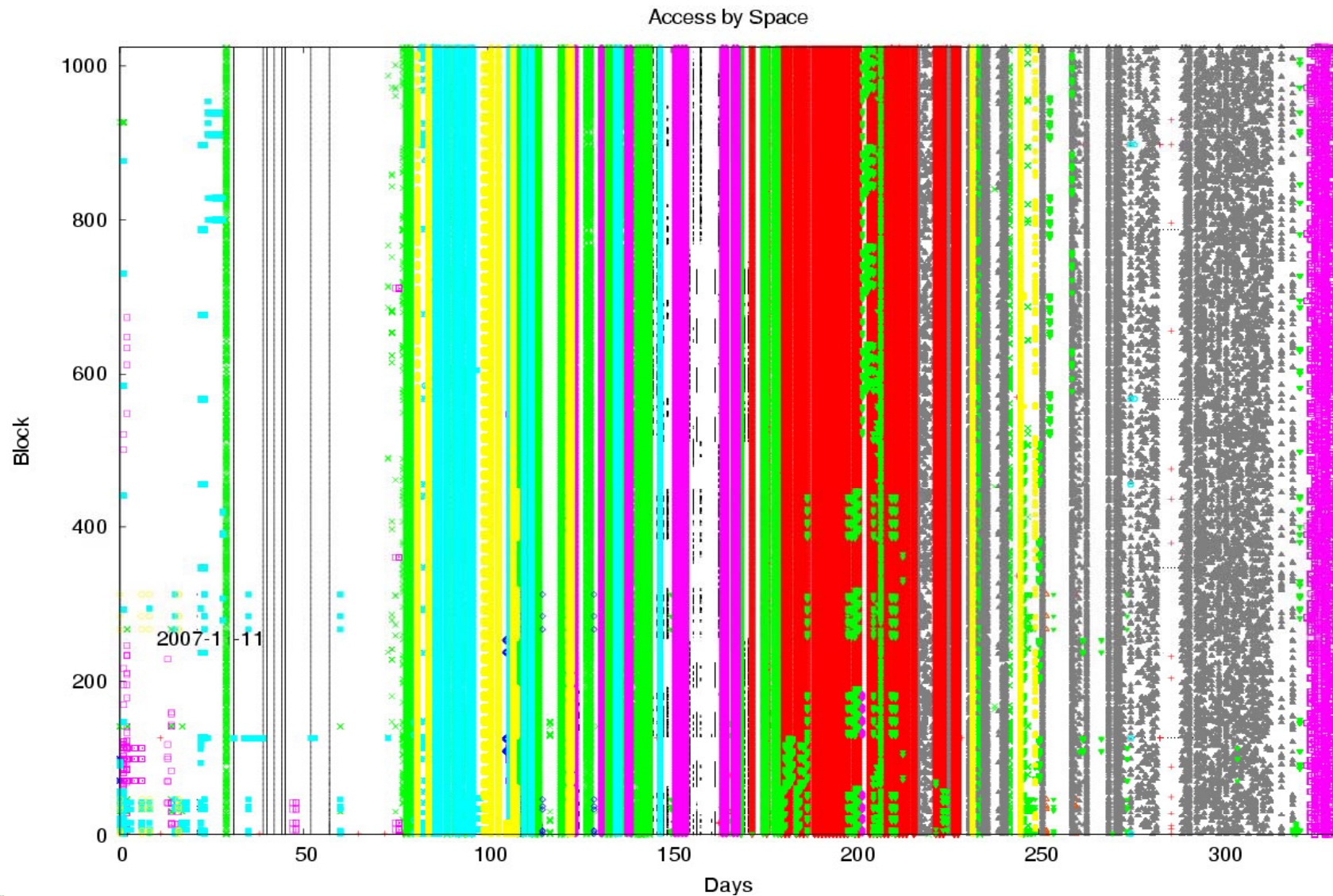
# Time Step Accessed by Job

(colors denote unique users)





# Spatial Region Accessed by Job



# Data-Driven Batch Scheduling

- Identify jobs with overlapping I/O requirement and co-schedule their execution on each timestep
  - Perform I/O once to each timestep for all outstanding jobs
  - Synchronize jobs that iterate through time
- Create batch and session interfaces
  - Sessions declare their time/space spans
  - Declarative: compute a function against a selected time/space region, which allows for out-of-order execution
- Previous results show >2x throughput improvement on declarative Astrophysics queries
  - Wang et al. CIDR 2009





# Future Directions for TurbulenceDB

- Low-power Amdahl blades
- I/O Enhancements
- Integration of DISC and HPC
  - For re-simulation, refinement, or compute intensive analysis
  - For rapid parallel ingest
- Multi-resolution storage
  - For fast coarse-grained ad-hoc queries
  - Support for visualization systems
- Improved metadata
  - Landmarks database
  - Support for education applications

