APS-DFD meeting, November 2007

Analysis of Turbulence Datasets using a Database Cluster: Requirements, Design, and Sample Applications

ITR group @ JHU

Charles Meneveau[1], Yi Li[1*], Eric Perlman[2], Minping Wan[1], Yunke Yang[1],

Randal Burns[2], Shiyi Chen[1], Gregory Eyink[3], Alex Szalay[4]

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

(* Now at Sheffield Univ. UK)

Funding: National Science Foundation
ITR and MRI

JOHNS HOPKINS
UNIVERSITY

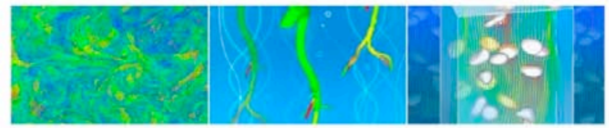
Report on Cyber Fluid Dynamics

Cyber-Fluid Dynamics

Report (Draft) on

NSF Workshop on Cyber-Fluid Dynamics:
New Frontiers in Research and Education

NSF Headquarters, Arlington VA
July 19-20, 2007
http://www.nsf-cyberfluids.gatech.edu



Organizers:

P.K. Yeung, Georgia Institute of Technology
R.D. Moser, University of Texas at Austin

NSF Program in Fluid Dynamics

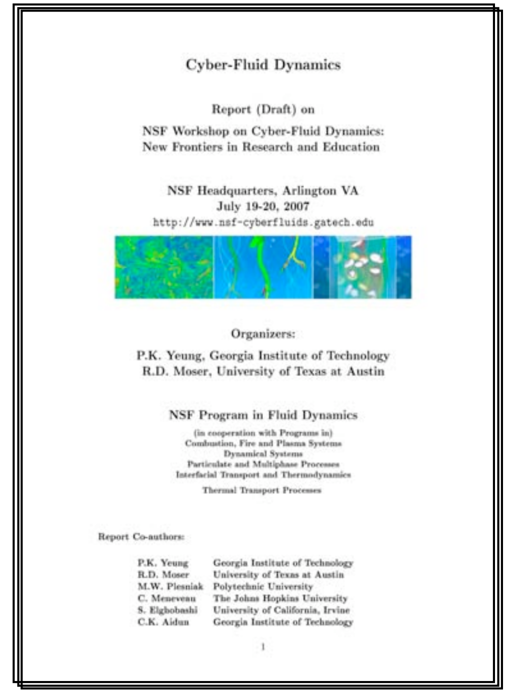
(in cooperation with Programs in)
Combustion, Fire and Plasma Systems
Dynamical Systems
Particulate and Multiphase Processes
Interfacial Transport and Thermodynamics
Thermal Transport Processes

Report Co-authors:

P.K. Yeung	Georgia Institute of Technology
R.D. Moser	University of Texas at Austin
M.W. Plesniak	Polytechnic University
C. Meneveau	The Johns Hopkins University
S. Elghobashi	University of California, Irvine
C.K. Aidun	Georgia Institute of Technology

JOHNS HOPKINS
UNIVERSITY

Report on Cyber Fluid Dynamics


Cyber-Fluid Dynamics

Report (Draft) on

NSF Workshop on Cyber-Fluid Dynamics:
New Frontiers in Research and Education

NSF Headquarters, Arlington VA
July 19-20, 2007
http://www.nsf-cyberfluids.gatech.edu

Organizers:
P.K. Yeung, Georgia Institute of Technology
R.D. Moser, University of Texas at Austin

NSF Program in Fluid Dynamics
(in cooperation with Programs in)
Combustion, Fire and Plasma Systems
Dynamical Systems
Particulate and Multiphase Processes
Interfacial Transport and Thermodynamics
Thermal Transport Processes

Report Co-authors:

P.K. Yeung	Georgia Institute of Technology
R.D. Moser	University of Texas at Austin
M.W. Plesniak	Polytechnic University
C. Meneveau	The Johns Hopkins University
S. Elghobashi	University of California, Irvine
C.K. Aidun	Georgia Institute of Technology

1

Among recommendations:

....

a culture of open communication and community-wide standards so that as many researchers as possible, including students exposed to the national supercomputing landscape, will benefit without duplication of effort. We call for leading data authors and data users in several areas to formulate community agreements on data formats and download or transfer protocols, especially for very large datasets of either computational or experimental origin. Efforts at building virtual organizations incorporating these elements, and more, are highly encouraged. We also recommend that the community work more closely, within itself and with NSF program

....

JOHNS HOPKINS
UNIVERSITY

Motivation:

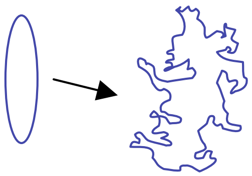
- DNS of turbulence generating huge multi-Tbyte datasets
- Analyzed on the fly during simulation by practitioner
- Must redo runs for each new question that was not foreseen during simulation,
- Or: store time-history
- If time-history is stored, one must retrieve data from large storage to large CPU for analysis, e.g. with local accounts on HPC machine.

Must know how to use and run programs there, privileges for access to large chunks of memory, casual initial analysis not possible without investing significant effort to set up such access…

Motivation:

- This prevailing approach has hampered wide accessibility of large CFD datasets.

  e.g. mathematician wishing to test whether material loops in turbulence develop fractal wrinkling will have a hard time exploring this question on (e.g.) a *$1{,}024^3$ DNS simulation* without asking the simulator to do it for him/her.



- The fact is:

  "very large simulations remain out of reach of most"

  The problem will not automatically get better - even if "ftp" gets 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.**

Motivation:

- Develop **database techniques** for analysis of DNS
- What does that mean?

Look to other fields where database technology is more developed (e.g. astronomy)

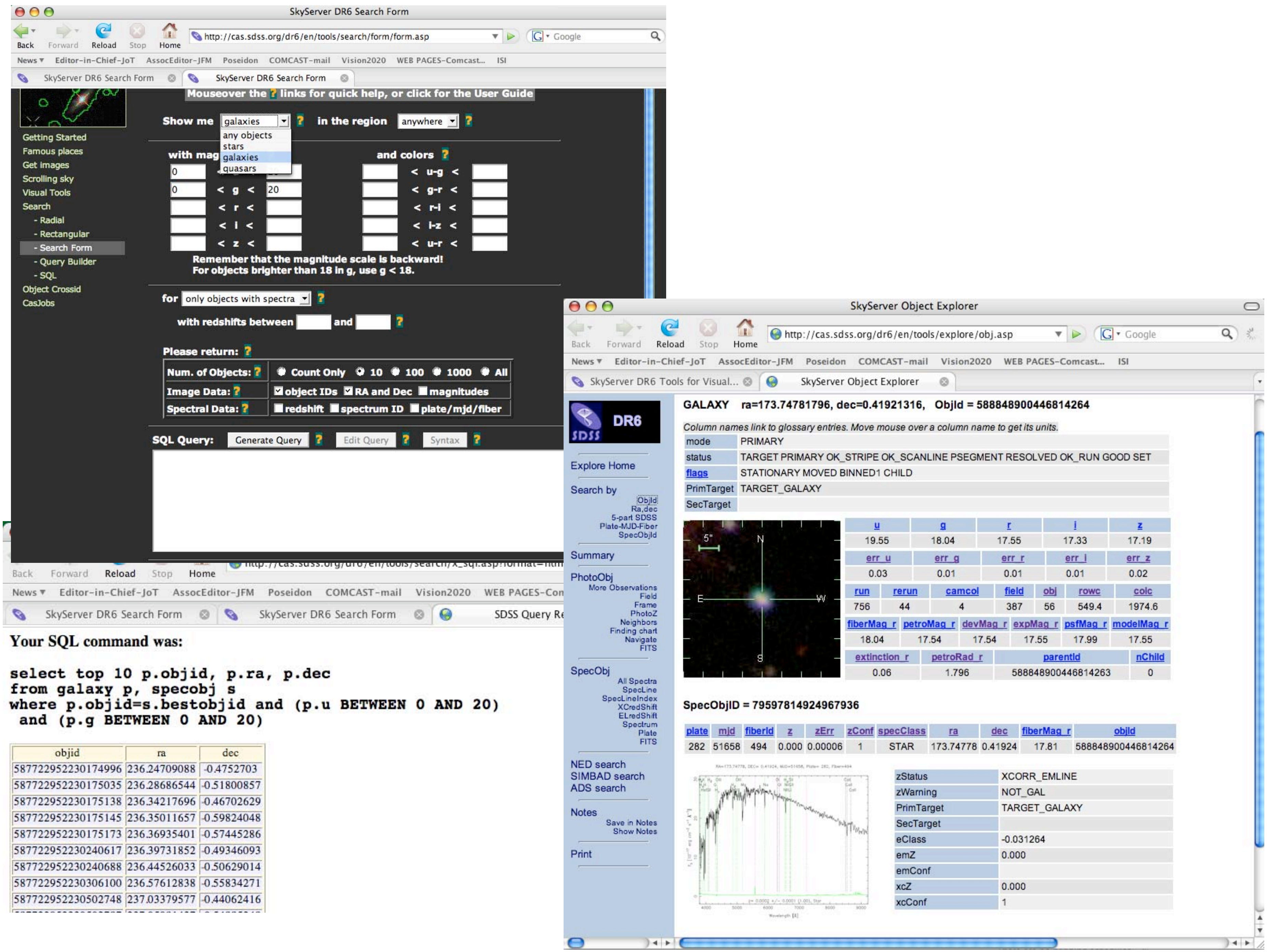
Example of web-accessible astronomy database (co-developed at JHU):

Sloan Digital Sky Survey (SDSS):

- When completed: images covering more than a quarter of the sky, and a 3-D map of about a million galaxies and quasars.
- SDSS (“the Cosmic Genome Project”) is the most cited astronomical resource in the world,
- Led to more than 1,300 refereed publications so far.
- Its 2.5 Terabyte archive has been accessed more than 200 million times over the last five years.
- Some breakthroughs this astronomy resource has enabled:
  - first precision measurements of the fundamental cosmological parameters
  - discovery of the most distant quasars.

Example of web-accessible astronomy database (co-developed at JHU):

fGetNearestObjIdAllEq	Returns the objId of nearest photo object within @r arcmins of ra, dec
fGetNearestObjIdEq	Returns the objId of nearest photoPrimary within @r arcmins of ra, dec
fGetNearestObjIdEqMode	Returns the objId of nearest @mode PhotoObjAll within @r arcmins of ra, dec
fGetNearestObjIdEqType	Returns the objID of the nearest photPrimary of type @t within @r arcmin
fGetNearestObjXYZ	Returns nearest primary object within @r arcminutes of an xyz point (@nx,@ny, @nz).
fGetObjectsEq	A helper function for SDSS cutout that returns all objects within a certain radius of an (ra,dec)
fGetObjectsMaskEq	A helper function for SDSS cutout that returns all objects within a certain radius of an (ra,dec)
fGetObjFromRect	Returns table of objects inside a rectangle defined by two ra,dec pairs. Note the order of the parameters: @ra1, @ra2, @dec1, @dec2
fGetObjFromRectEq	Returns table of objects inside a rectangle defined by two ra,dec pairs. Note the order of the parameters: @ra1, @dec1, @ra2, @dec2
fGetPhotoApMag	Returns the r-band aperture magnitude and error for the given aperture.
fGetUrlExpEq	Returns the URL for an ra,dec, measured in degrees.
fGetUrlExpId	Returns the URL for an Photo objID.
fGetUrlFitsAtlas	Get the URL to the FITS file of all atlas images in a field
fGetUrlFitsBin	Get the URL to the FITS file of a binned frame given FieldID and


SkyServer DR6 Search Form
http://cas.sdss.org/dr6/en/tools/search/form/form.asp
Mouseover the ? links for quick help, or click for the User Guide
Show me galaxies in the region anywhere
any objects
stars
galaxies
quasars
with mag
and colors
Remember that the magnitude scale is backward!
For objects brighter than 18 in g, use g < 18.
for only objects with spectra
with redshifts between and
Please return:
Num. of Objects: Count Only 10 100 1000 All
Image Data: object IDs RA and Dec magnitudes
Spectral Data: redshift spectrum ID plate/mjd/fiber
SQL Query: Generate Query Edit Query Syntax
Getting Started
Famous places
Get images
Scrolling sky
Visual Tools
Search
- Radial
- Rectangular
- Search Form
- Query Builder
- SQL
Object Crossid
CasJobs
Your SQL command was:
select top 10 p.objid, p.ra, p.dec
from galaxy p, specobj s
where p.objid=s.bestobjid and (p.u BETWEEN 0 AND 20)
and (p.g BETWEEN 0 AND 20)
objid ra dec
587722952230174996 236.24709088 -0.4752703
587722952230175035 236.28686544 -0.51800857
587722952230175138 236.34217696 -0.46702629
587722952230175145 236.35011657 -0.59824048
587722952230175173 236.36935401 -0.57445286
587722952230240617 236.39731852 -0.49346093
587722952230240688 236.44526033 -0.50629014
587722952230306100 236.57612838 -0.55834271
587722952230502748 237.03379577 -0.44062416
SkyServer Object Explorer
http://cas.sdss.org/dr6/en/tools/explore/obj.asp
GALAXY ra=173.74781796, dec=0.41921316, ObjId = 588848900446814264
Column names link to glossary entries. Move mouse over a column name to get its units.
mode PRIMARY
status TARGET PRIMARY OK_STRIPE OK_SCANLINE PSEGMENT RESOLVED OK_RUN GOOD SET
flags STATIONARY MOVED BINNED1 CHILD
PrimTarget TARGET_GALAXY
SecTarget
u g r i z
19.55 18.04 17.55 17.33 17.19
err_u err_g err_r err_i err_z
0.03 0.01 0.01 0.01 0.02
run rerun camcol field obj rowc colc
756 44 4 387 56 549.4 1974.6
fiberMag_r petroMag_r devMag_r expMag_r psfMag_r modelMag_r
18.04 17.54 17.54 17.55 17.99 17.55
extinction_r petroRad_r parentId nChild
0.06 1.796 588848900446814263 0
SpecObjID = 79597814924967936
plate mjd fiberId z zErr zConf specClass ra dec fiberMag_r objId
282 51658 494 0.000 0.00006 1 STAR 173.74778 0.41924 17.81 588848900446814264
zStatus XCORR_EMLINE
zWarning NOT_GAL
PrimTarget TARGET_GALAXY
SecTarget
eClass -0.031264
emZ 0.000
emConf
xcZ 0.000
xcConf 1
DR6 SDSS
Explore Home
Search by
Summary
PhotoObj
SpecObj
NED search
SIMBAD search
ADS search
Notes
Print

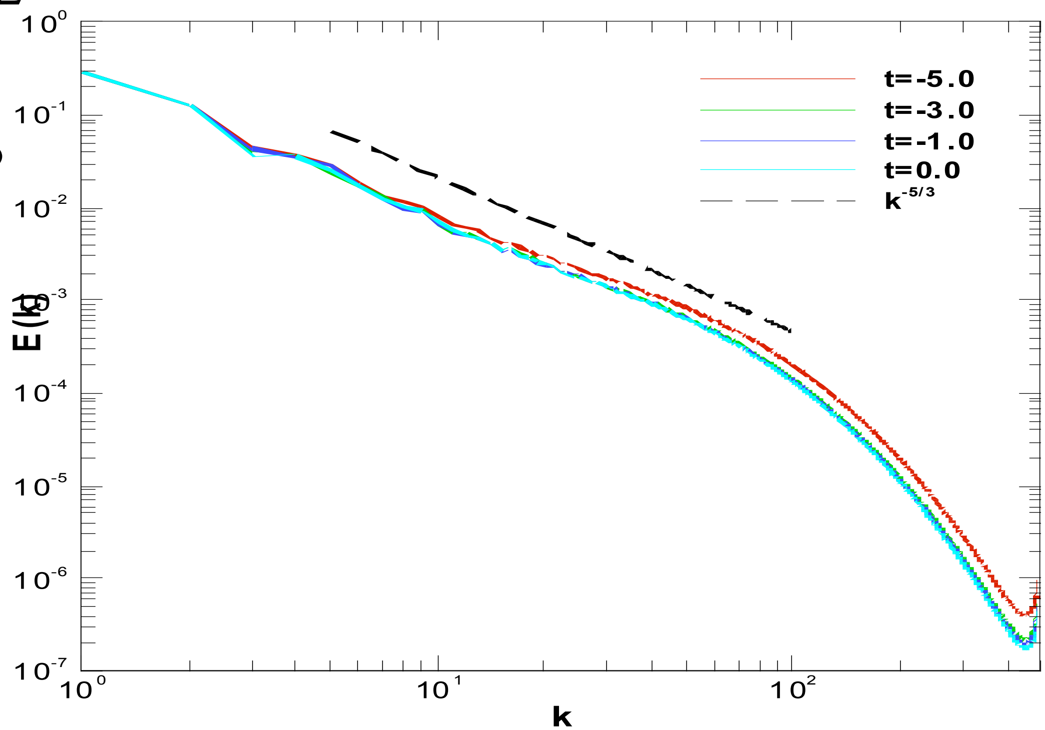
Can same technology be used to store and access a full DNS database?

- Turbulence has concentrated coherent structures (like galaxies ?)
- Can data be stored in a way that facilitates particular queries?
- Can/should these be classified in some way?
- e.g. SQL search for “most intense vortices in region around high mean dissipation” ??
- Too cumbersome. We’ll show that slightly different approach is needed
- First, describe the dataset:

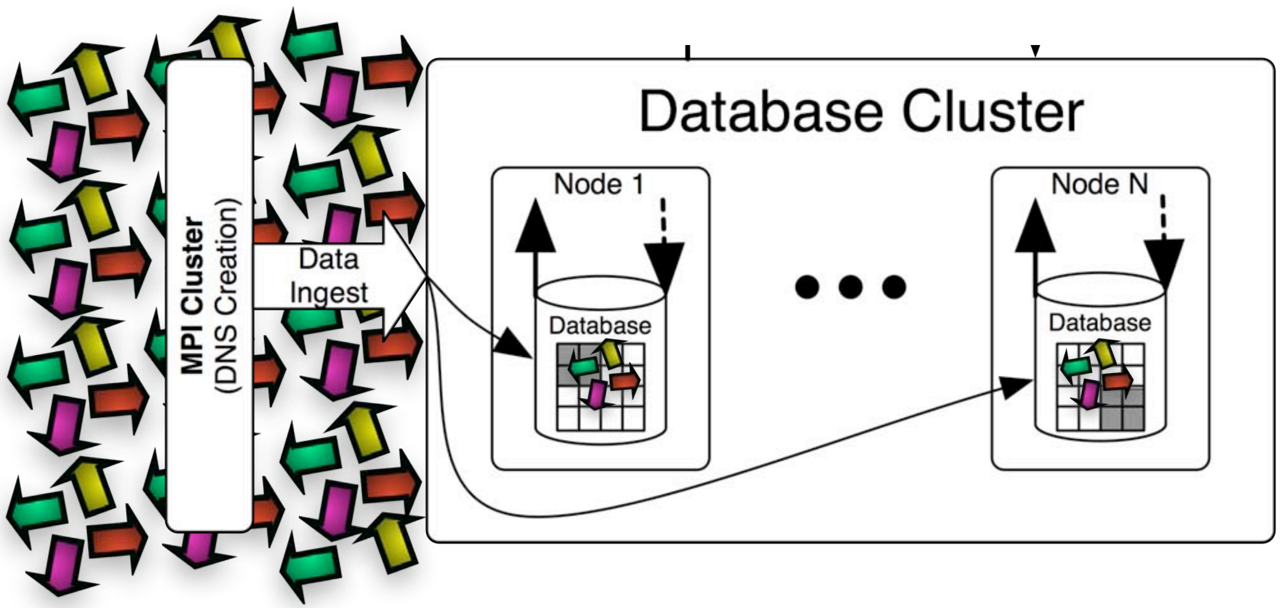
Dataset: *a 1,024^4 space-time history of isotropic turbulence*

- 1,024^3 DNS of forced isotropic turbulence, in $(2\pi)^3$ periodic box
- pseudo-spectral, de-aliased (phase-shifting - Rogallo)
- rms velocity: $u' = 0.679$
- Taylor microscale Reynolds number $R_\lambda = 425$
- Kolmogorov time scale: $\tau_K = 0.044$
- Kolmogorov length scale: $\eta = 2.86\text{E-}03$
- Large eddy turnover time: $T_L \approx 2.2$
- Simulation time-step: $\delta t = 0.0002$
- Every 10 steps stored at $\Delta t = 0.002$
- 1,024 times to be stored, in T_L

  (~ 1 integral time-scale)

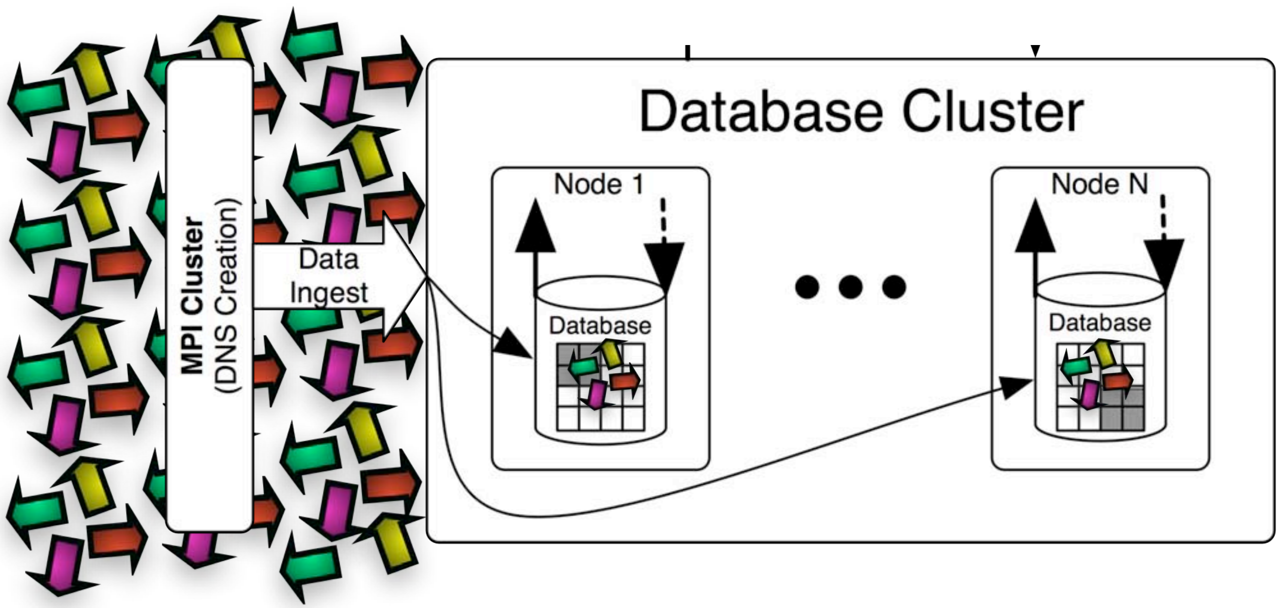


Database cluster architecture:



- **Each cluster node:**
  - Runs Windows 2003 Server and SQL Server 2005 (64-bit)
  - Data tables striped across 12-15 sets of mirrored disks (4 - 11 TB/node)
- **4 or 8 GB memory per node**
- **Code written in C# runs as a user defined procedure inside the SQL Server process using the CLR environment**

Data generation and ingest:

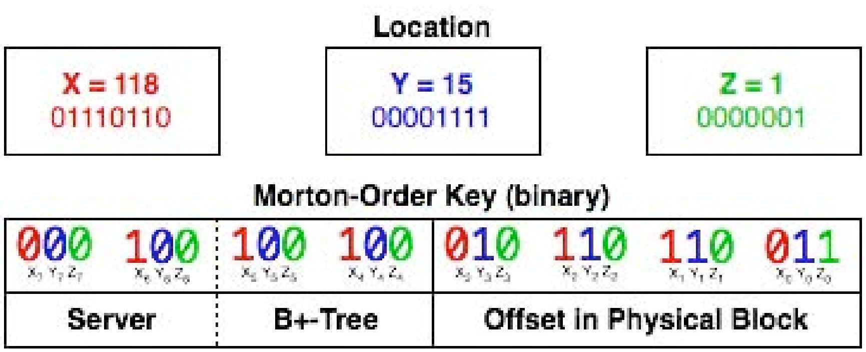


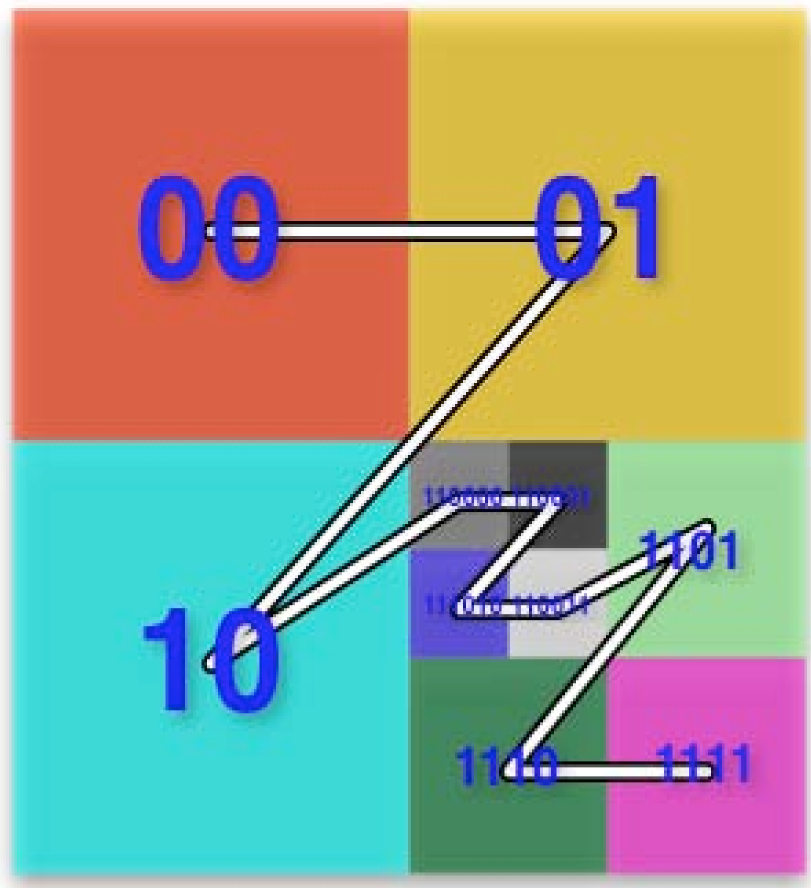
Will be 27 Tbytes

- **The DNS is run on a Linux cluster at JHU**
- **Output is stored as standard C/Fortran arrays**
- **Files are copied onto the data cluster for ingest**
- **Data are reshaped into smaller cubes and inserted into the database using "BULK INSERT"**
  - –We read 72x1024x1024 slices into memory at a time

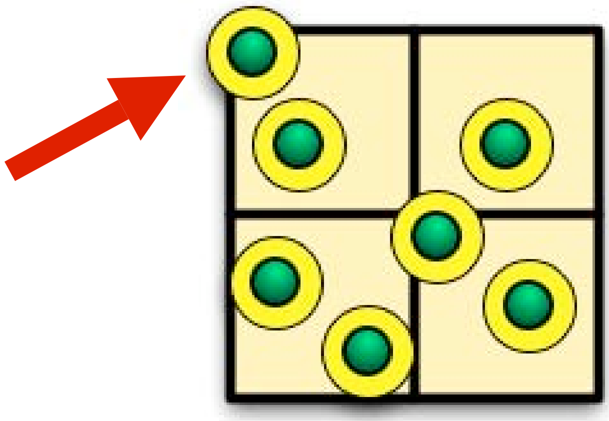
Unified addressing and partitioning

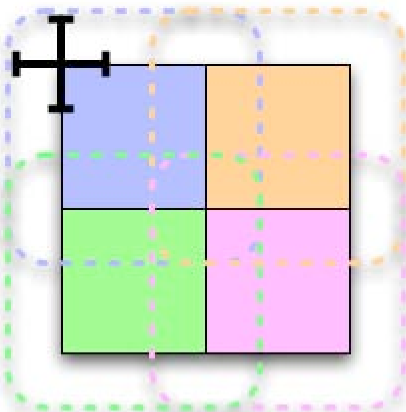
- **Oct-trees and Z-curves (Morton) are used for:**
  - Indexing space
  - Indexing workload
  - Data distribution
  - Data partitioning





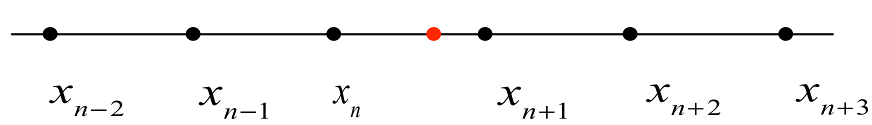
Kernel of Computation & Edge Replication





- **Most routines need "kernel of computation"**
  - **Spatial Interpolation**

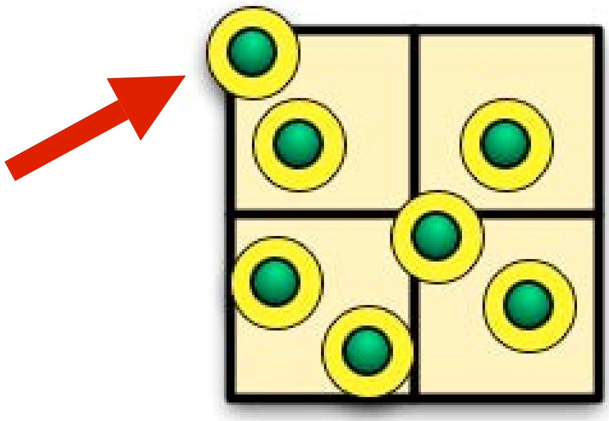
    Lagrange polynomials 6th and 8th order



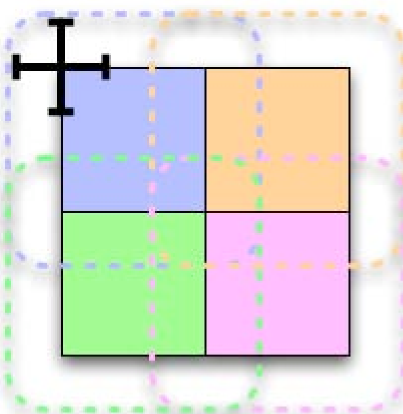
$$u(x) = \sum_{i=1}^{6} l_{n-3+i}(x)\, u(x_{n-3+i})$$

$$l_j(x) = \frac{\prod_{i=n-2, i\neq j}^{n+3} (x - x_i)}{\prod_{i=n-2, i\neq j}^{n+3} (x_j - x_i)}$$

Kernel of Computation & Edge Replication



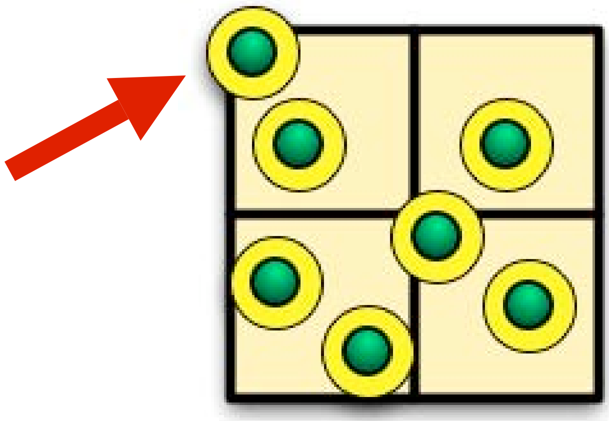
- **Most routines need "kernel of computation"**
  - **Spatial Interpolation**
    Lagrange polynomials 6th and 8th order
  - **Velocity gradients**
    Based on same Lagrange polynomials
  - **Spatial averages**
    Not yet implemented

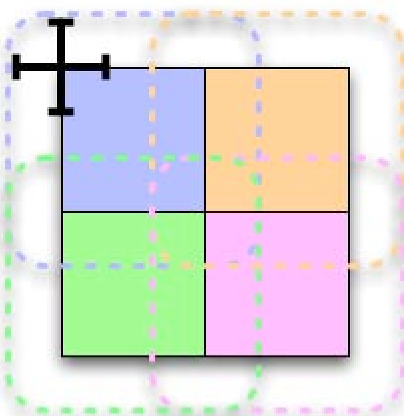


- **We replicate edge of each region**
  - **Data needs met with a single read**
  - **48% overhead (64^3 to 72^3)**
  - **Data "molecule" is 72^3 subcube**

- **Time interpolation:**
  - Piecewise Cubic Hermite Interpolation:
    needs 4 t-neighbors

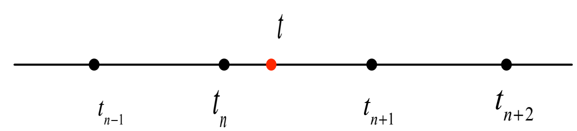
Kernel of Computation & Edge Replication





- **Most routines need "kernel of computation"**
  - **Spatial Interpolation**
    Lagrange polynomials 6th and 8th order
  - **Velocity gradients**
    Based on same Lagrange polynomials
  - **Spatial averages**
    Not yet implemented

- **Time interpolation:** Piecewise Cubic Hermite
  Interpolation: needs 4 t-neighbors



$$u(t) = a + b(t - t_n) + c(t - t_n)^2 + d(t - t_n)^2 (t - t_{n+1})$$

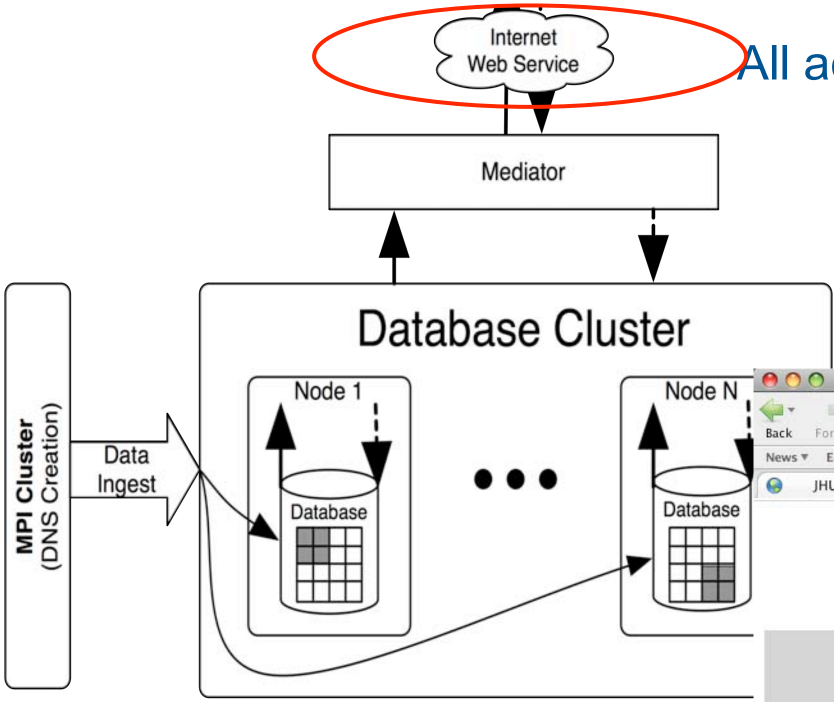
$$a = u(t_n) \qquad c = \frac{1}{\Delta t}\left(\frac{u(t_{n+1}) - u(t_n)}{\Delta t} - \dot{u}(t_n)\right)$$

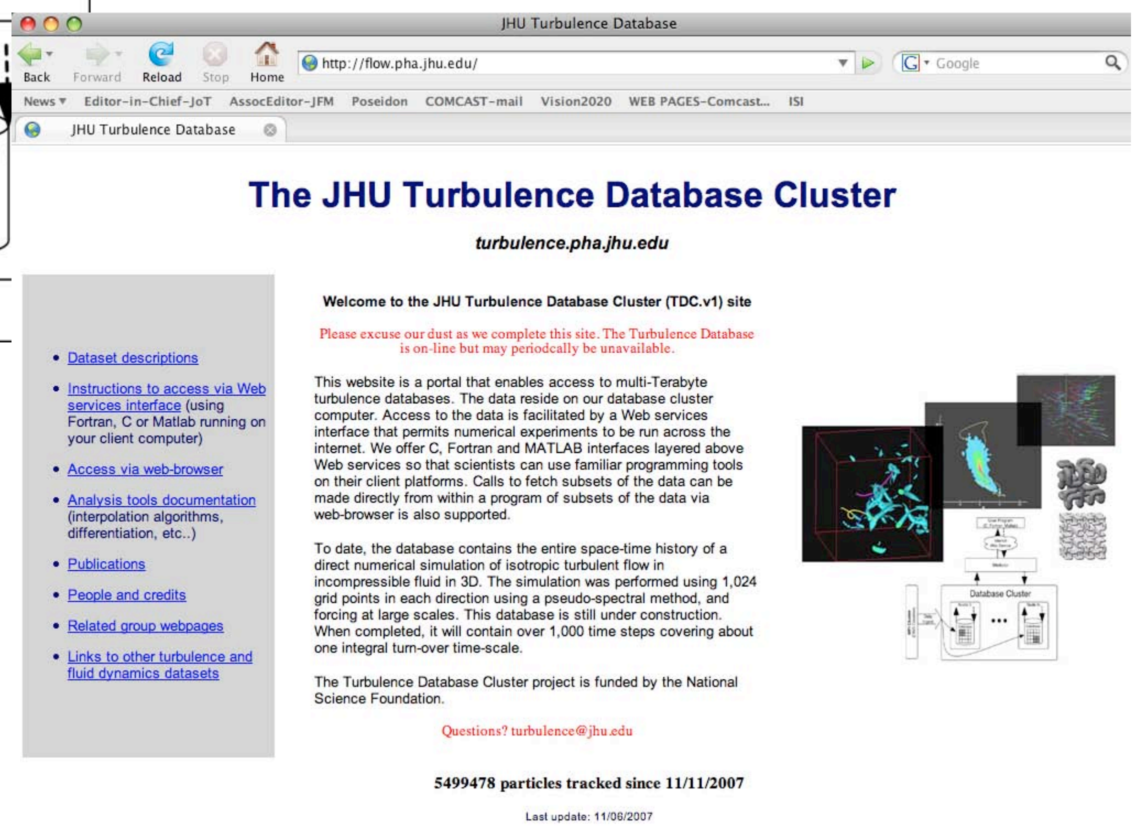
$$b = \dot{u}(t_n) = \frac{u(t_{n+1}) - u(t_{n-1})}{2\Delta t}$$

$$d = \frac{2}{\Delta t^2}\left\{\frac{1}{2}\left[\dot{u}(t_n) + \dot{u}(t_{n+1})\right] - \frac{u(t_{n+1}) - u(t_n)}{\Delta t}\right\}$$

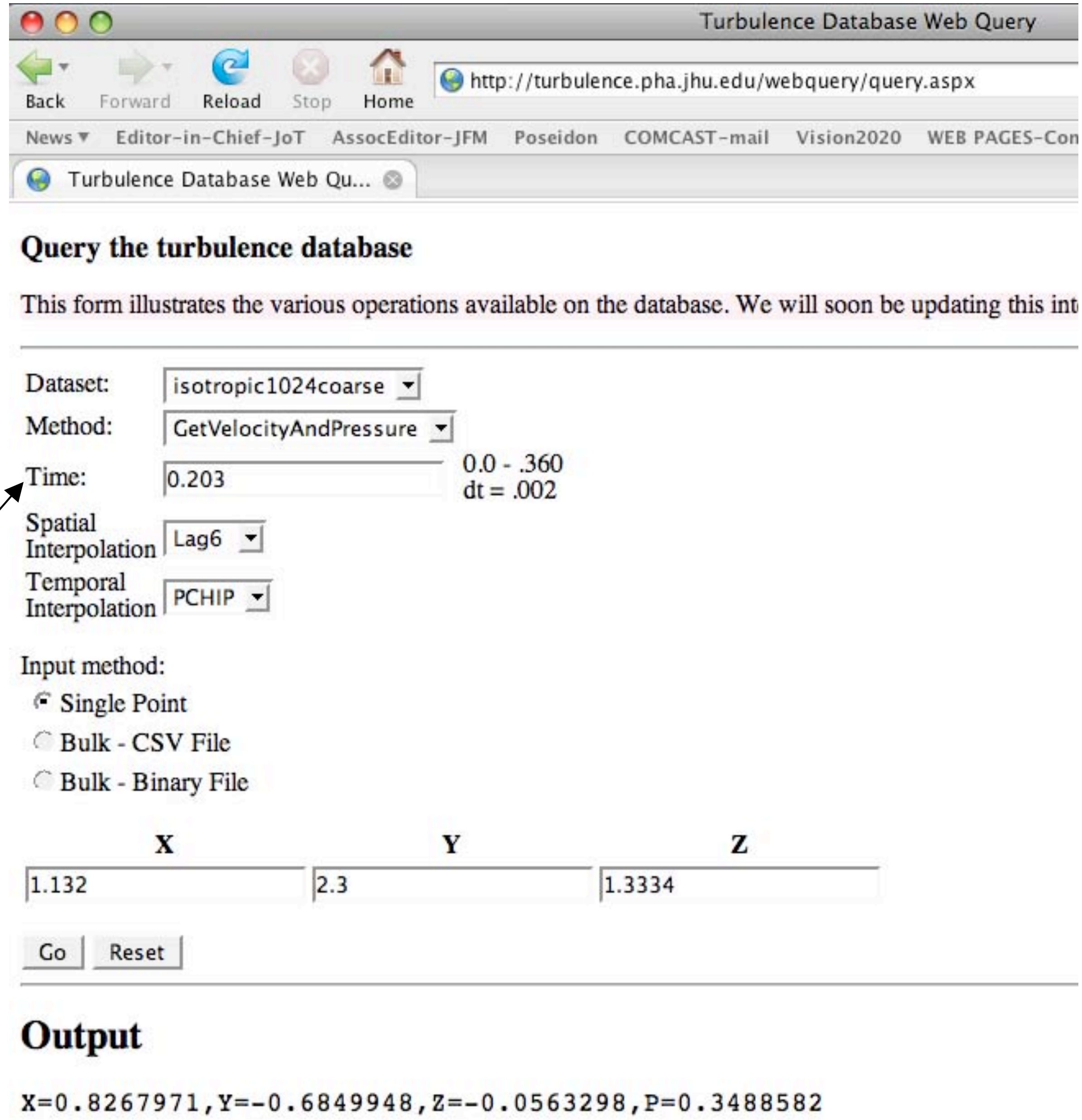
Accessing the database: http://turbulence.pha.jhu.edu

All access to database is done through Web Service





Demo: web accessible queries:

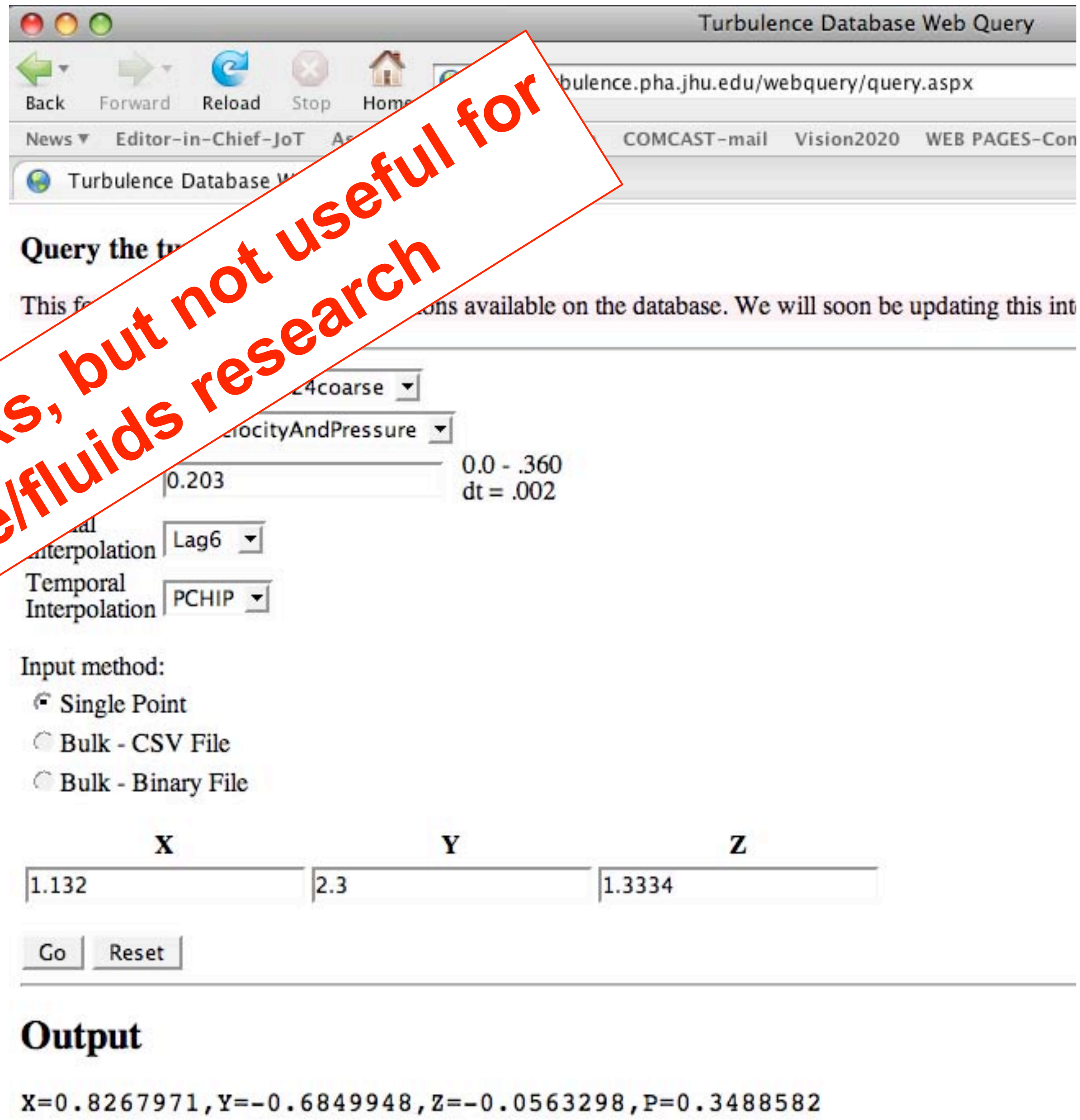


Enter time

Enter position

Get velocity and pressure output

Demo: web accessible queries:



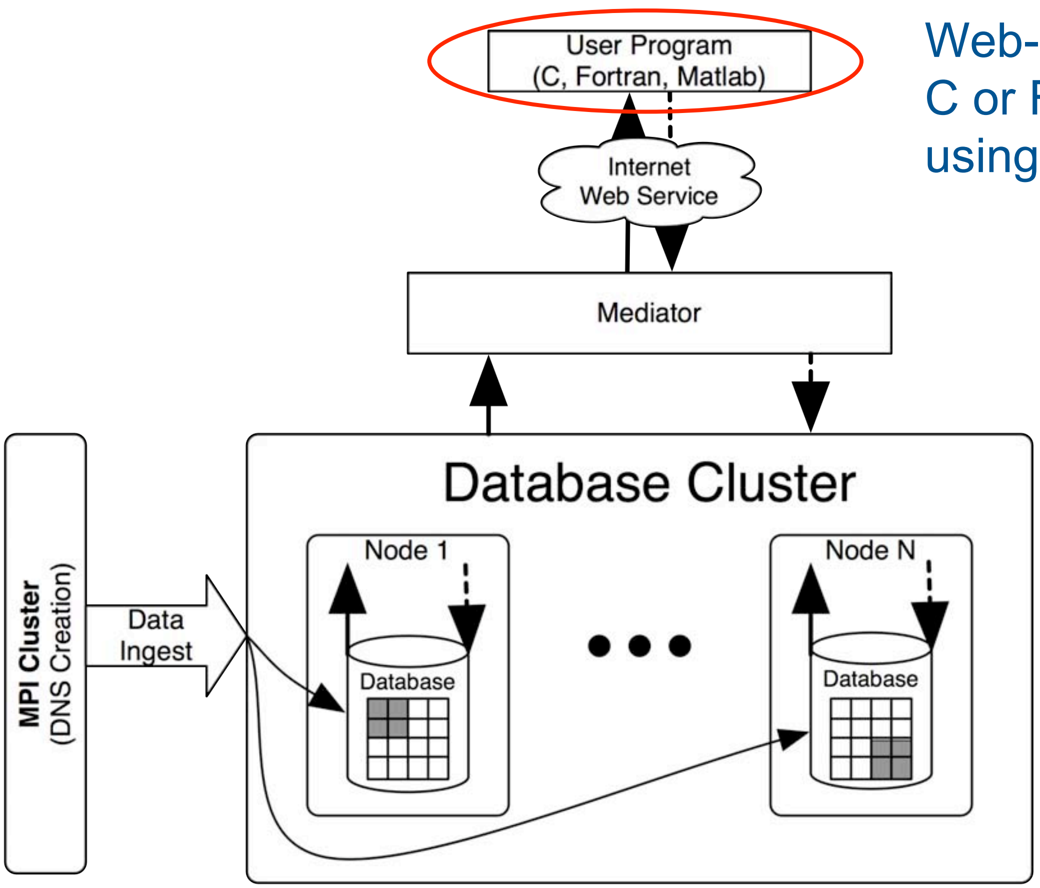
OK for casual checks, but not useful for serious turbulence/fluids research

ne

nter position

Get velocity and pressure output

New paradigm: client computer runs the analysis using fortran, C, matlab codes and they fetch data from database through a web-service

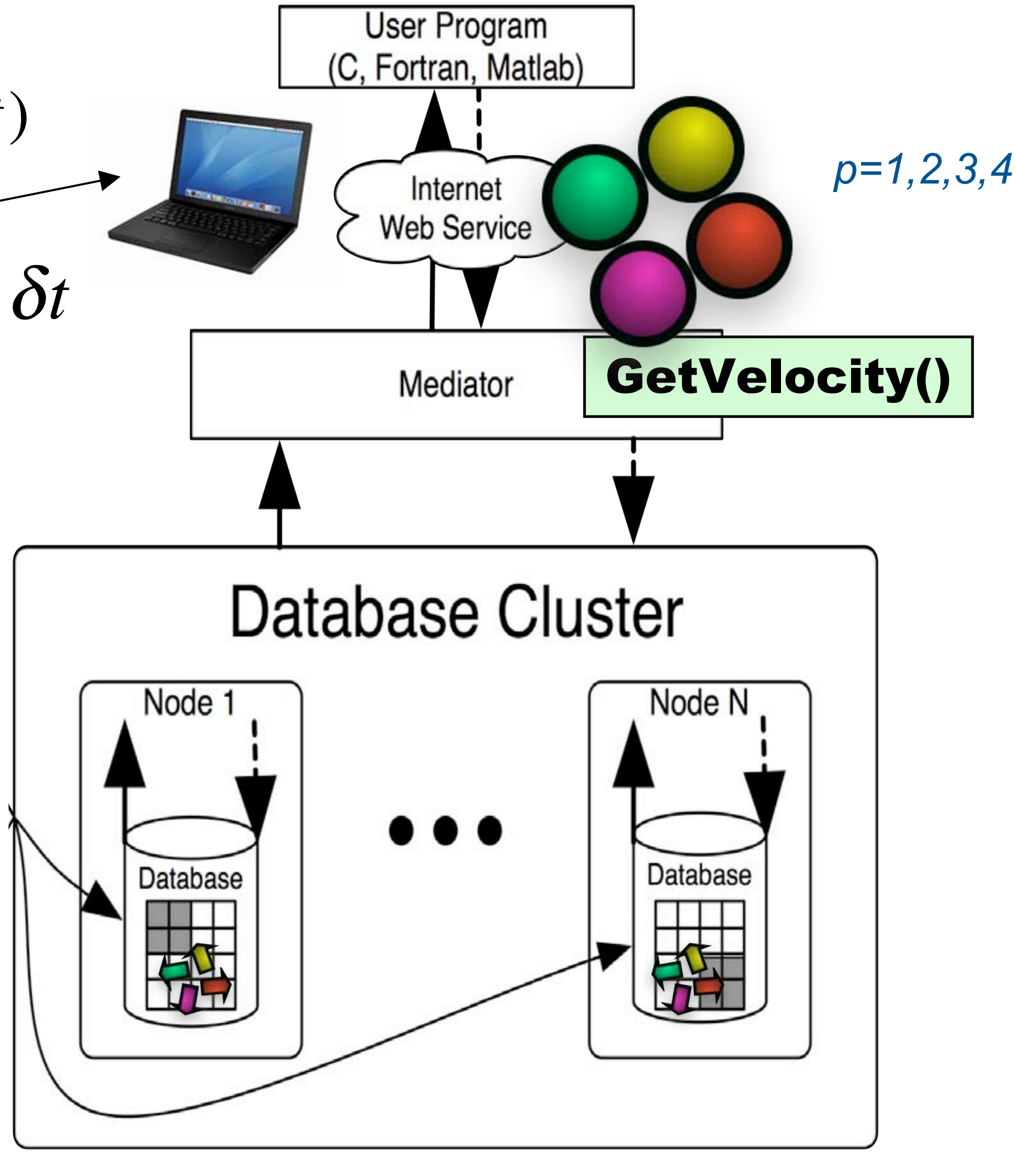


Web-Service calls can be made from C or Fortran on client computer using the gSOAP libraries…

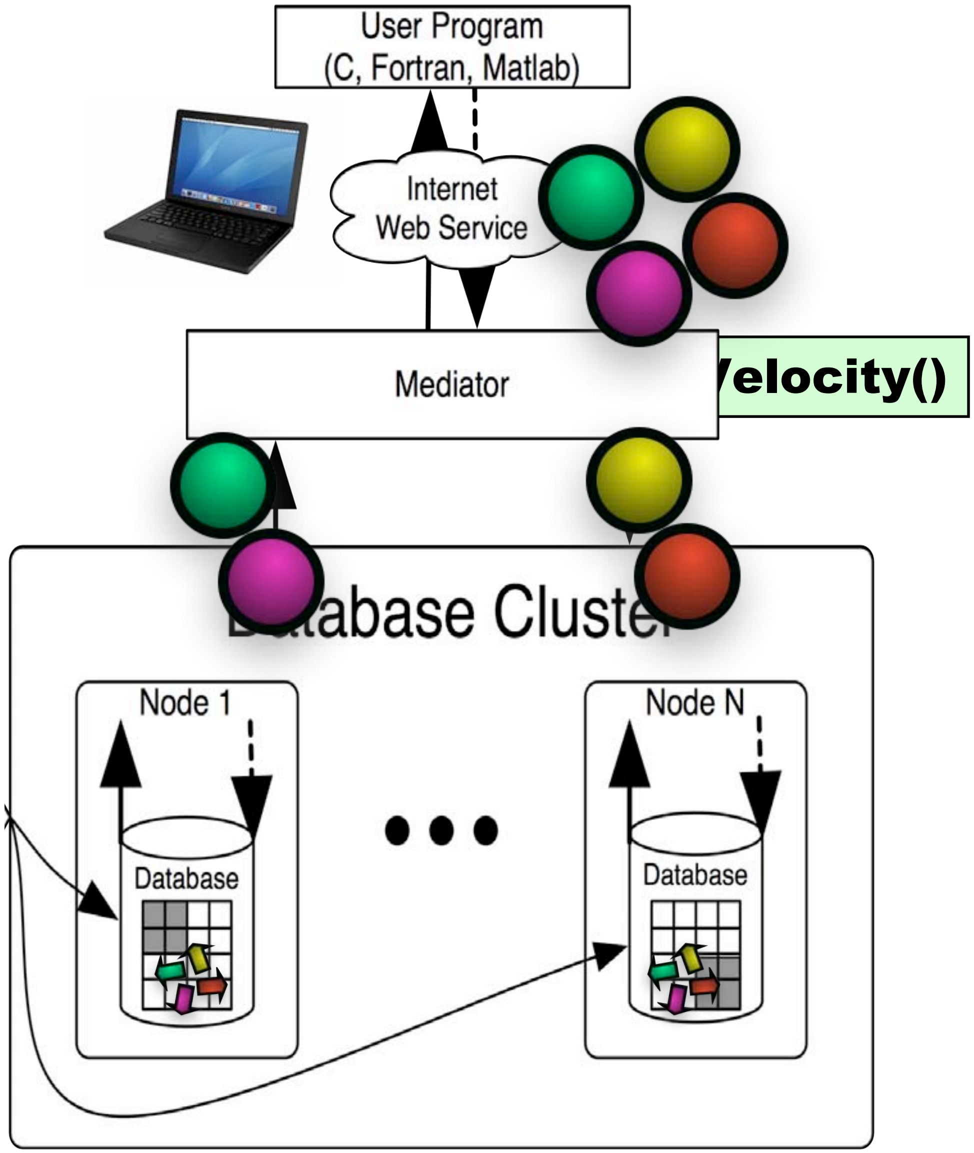
Web service request for GetVelocity

$$\frac{d\mathbf{x}_p(t)}{dt} = \mathbf{u}(\mathbf{x}_p(t),t)$$

$$\mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \mathbf{u}(\mathbf{x}_p^n, t^n)\ \delta t$$



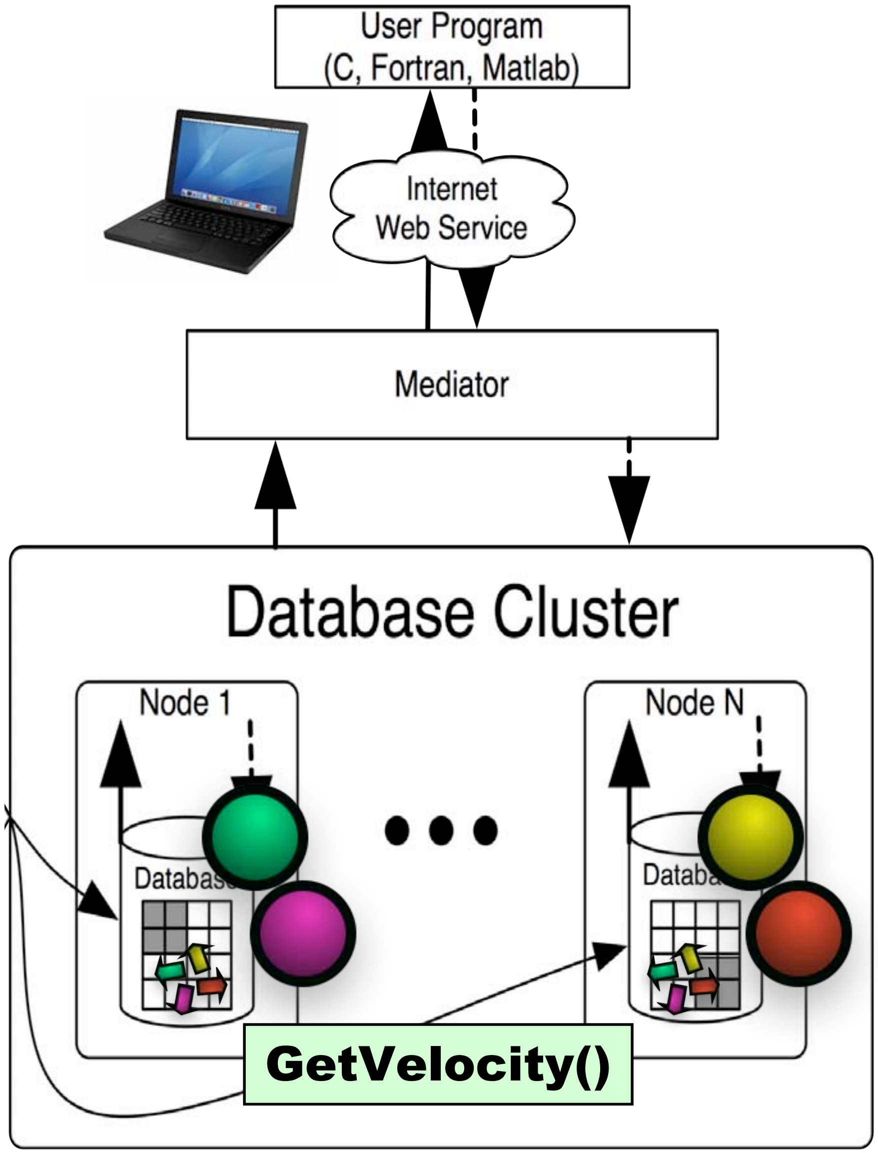
Mediator divides workload spatially



Request sent to each database node

Heavy local computations done close to the data:

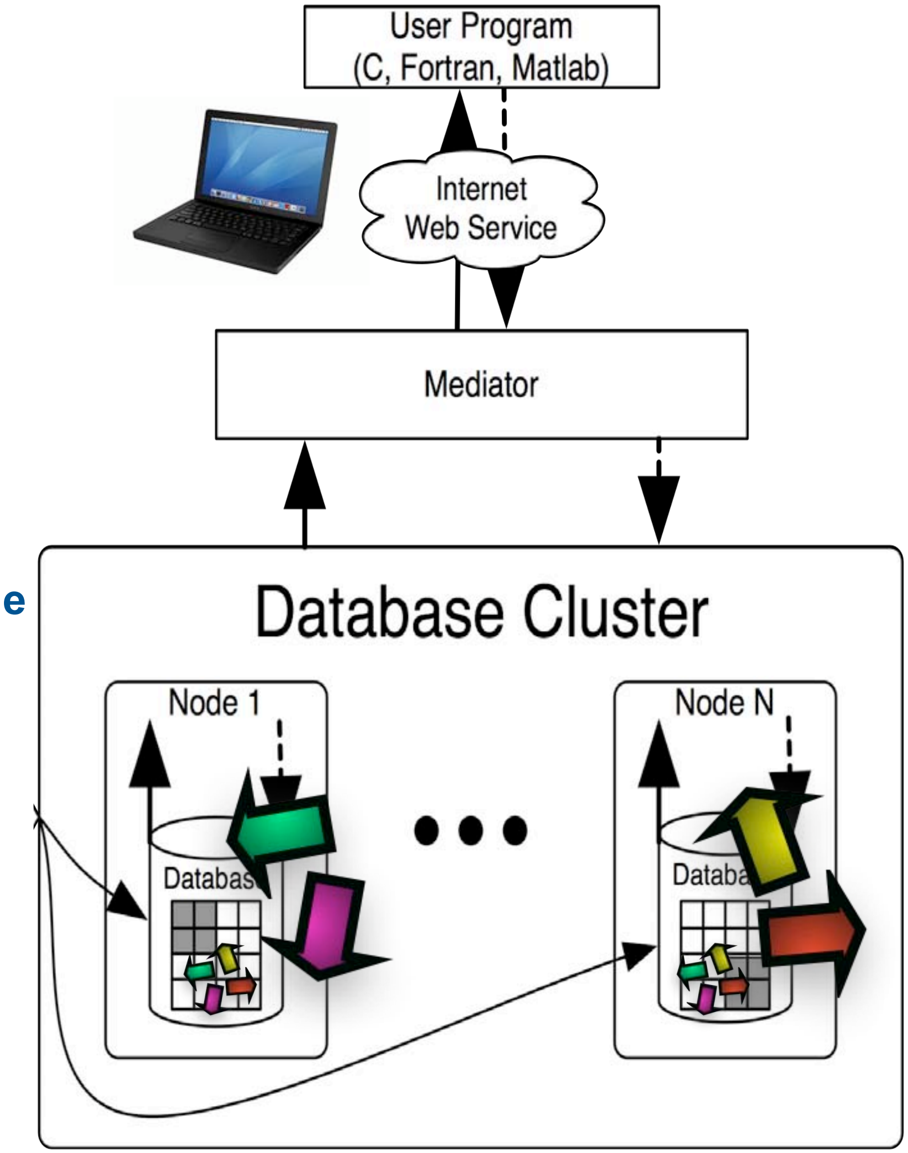
- **Lagrange-polynomial interpolation in space**
- **P-cubic Hermite interp in time**
- **Other functions:** vel gradient tensor, pressure, Hessian, Laplacian, pressure gradient…



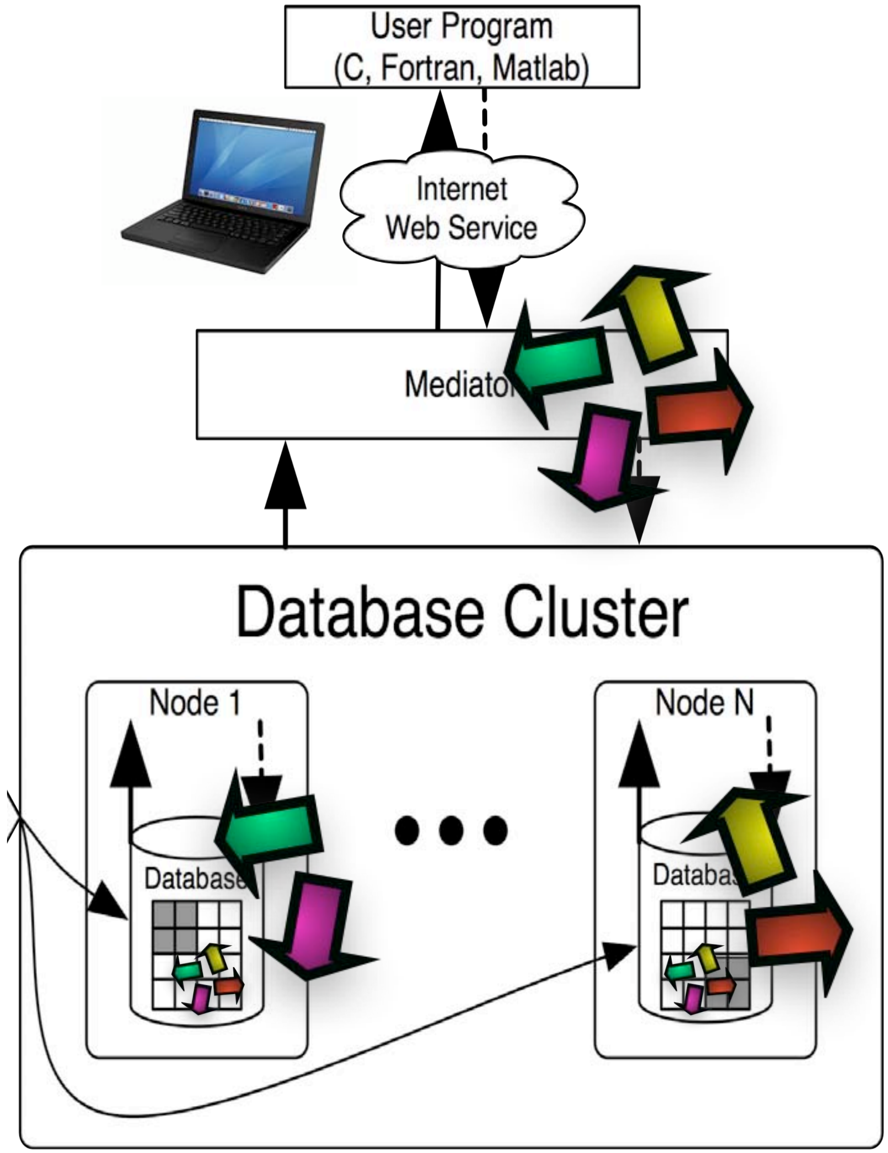
Velocity at each location is returned

Heavy local computations done close to the data:

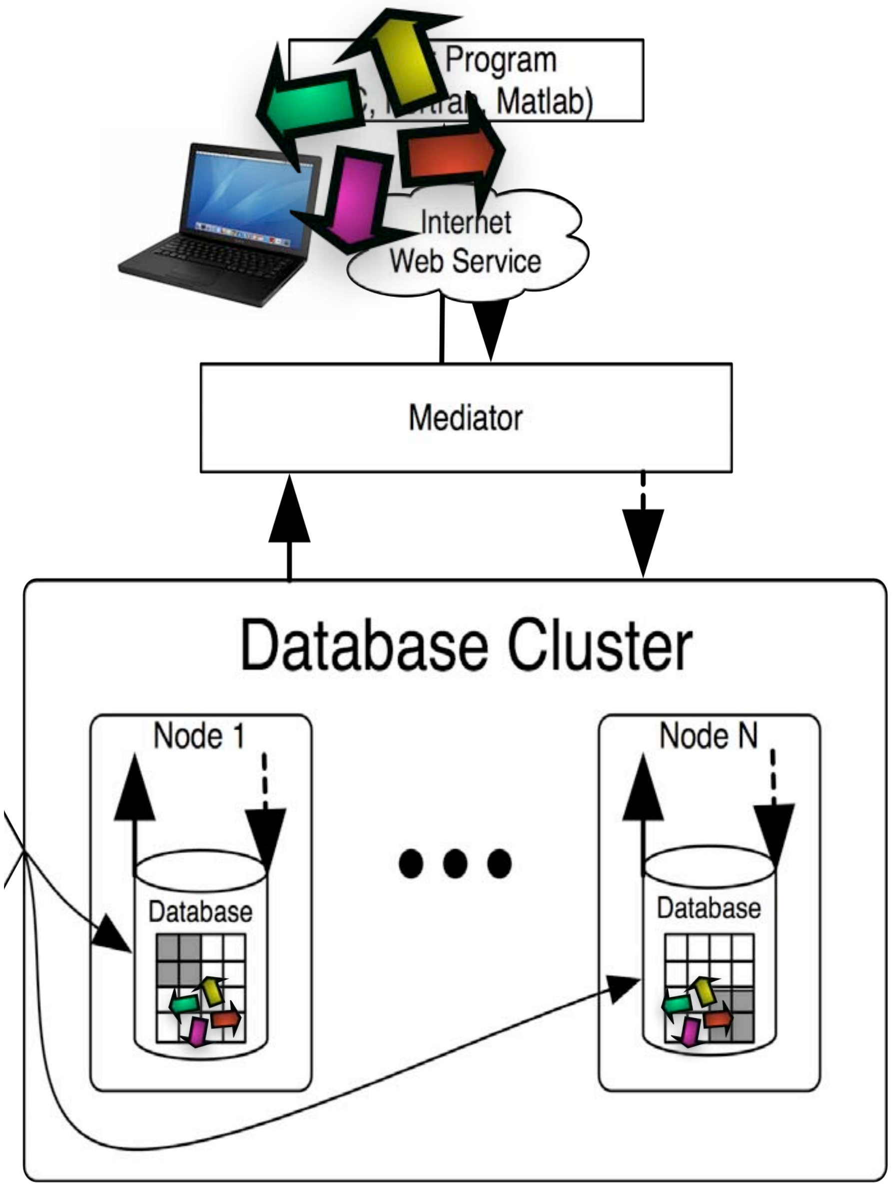
- **Lagrange-polynomial interpolation in space**
- **P-cubic Hermite interp in time**
- **Other functions:** vel gradient tensor, pressure, Hessian, Laplacian, pressure gradient…



Velocity values are collated by the mediator

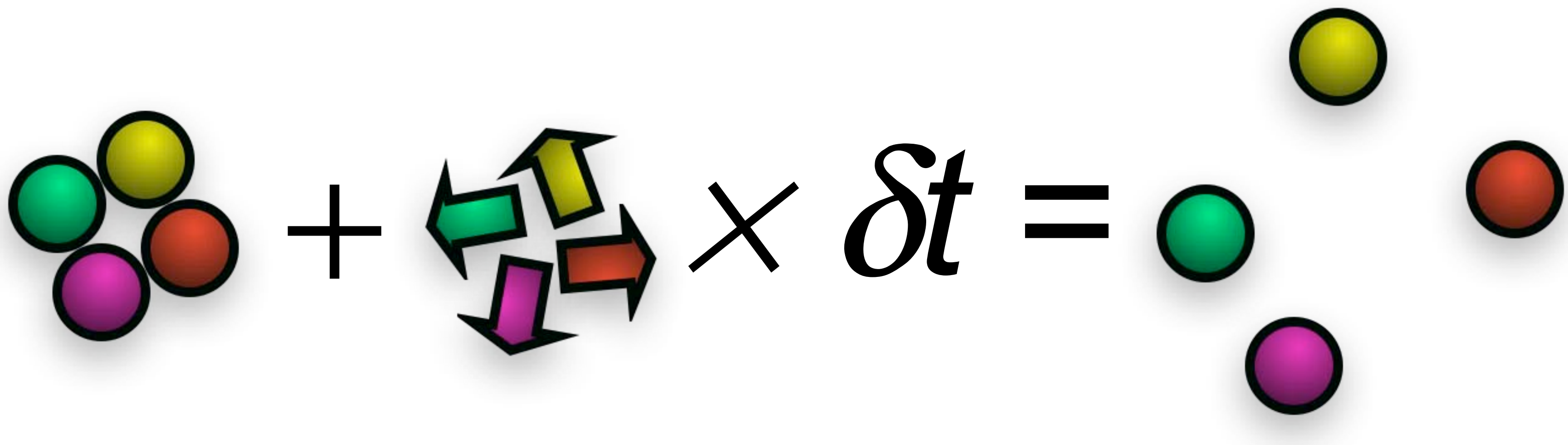


… and returned to the user



Client uses velocities to determine new positions

$$\mathbf{x}_p^n + \mathbf{u}(\mathbf{x}_p^n, t^n)\ \delta t = \mathbf{x}_p^{n+1}$$



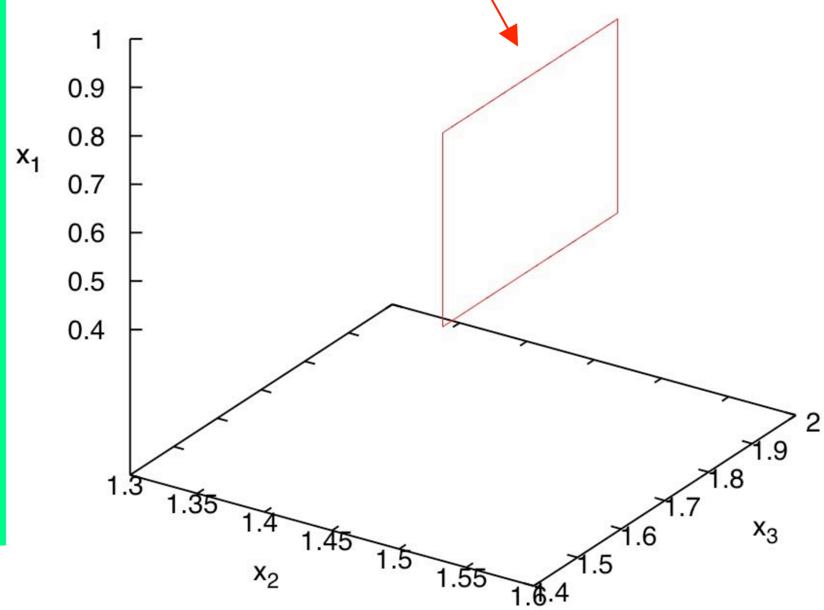
… and repeat with a new time *t*…

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

UNIX Terminal — tcsh — ttys000

```
 !
!-----------------------------------------------------------------------------
 if (icase.eq.1) then
!
 open (10,file='loop.dat',status='unknown')
!
 write(*, *) 'A test of the JHU Turbulence Database'
 write(*, *) 'Tracking a square loop'
!
 time=0.15
 deltat=0.003
 n=250
!
 do i=1,n,1
 points(1,i)=1.5
 points(2,i)=1.5
 points(3,i)=0.5+i*0.4/n
 end do
 do i=1,n,1
 points(1,i+n)=1.5
 points(2,i+n)=1.5+i*0.4/n
 points(3,i+n)=0.5+n*0.4/n
 end do
 do i=1,n,1
 points(1,i+2*n)=1.5
 points(2,i+2*n)=1.5+n*0.4/n
 points(3,i+2*n)=0.5+(n-i)*0.4/n
 end do
 do i=1,n,1
 points(1,i+3*n)=1.5
 points(2,i+3*n)=1.5+(n-i)*0.4/n
 points(3,i+3*n)=0.5
 end do
!
   do i=1,4*n
     write(10,*) points(1,i),points(2,i),points(3,i)
   end do
     write(10,*) points(1,1),points(2,1),points(3,1)
!
```

Pick 1,000 particle positions around a square loop



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

```
!
 do it = 1,15,1
!
   print *,'time = ',time
   time=time+deltat
!
   CALL getvelocity(authkey, dataset1, time, Lagrangian6thOrder , PCHIPInterpolation, 4*n, points, dataout)
 !
  do i=1,4*n
  do k=1,3
    points(k,i)=points(k,i)+dataout(k,i)*deltat
  end do
  end do
!
  if (it.eq.5.or.it.eq.10.or.it.eq.15) then
   do i=1,4*n
    write(10,*) points(1,i),points(2,i),points(3,i)
   end do
    write(10,*) points(1,1),points(2,1),points(3,1)
  endif
  write(10,*) '  '
  end do
!
  endif
```

advect and print-out results at subsequent times

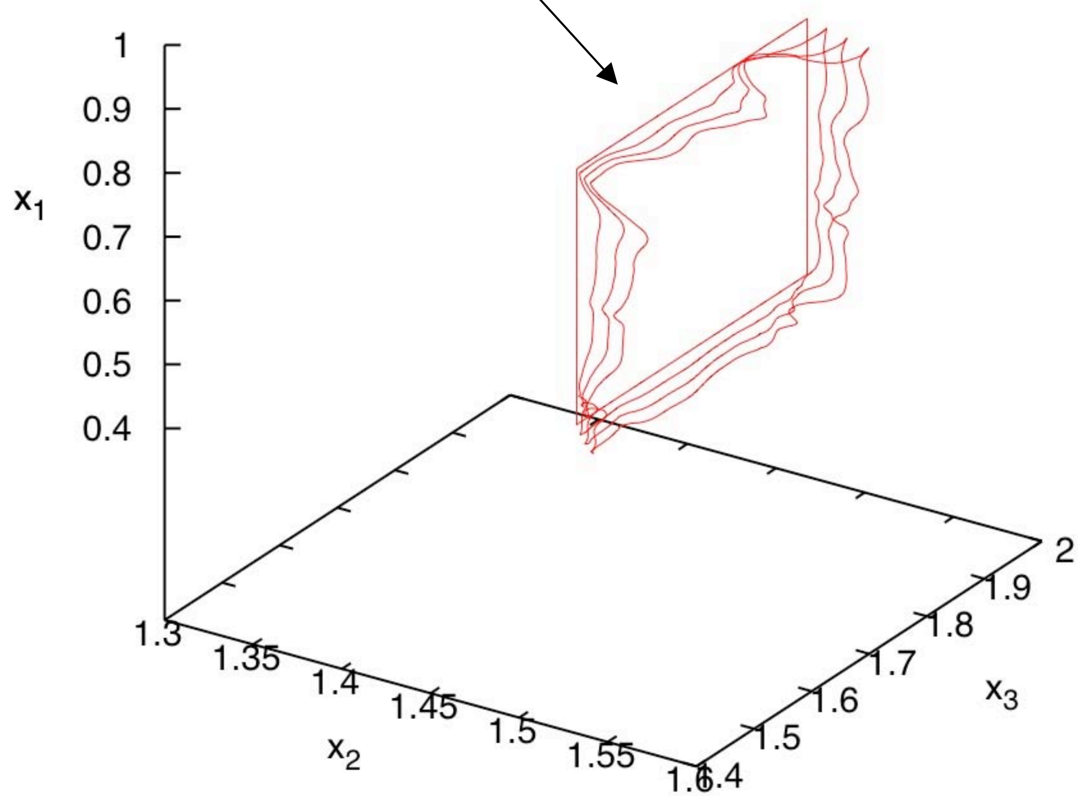
4.3 sec. wall-time/time-step
for 1,000 particles

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

```
!
 do it = 1,15,1
!
   print *,'time = ',time
   time=time+deltat
!
   CALL getvelocity(authkey, dataset1, time, Lagrangian6thOrder , PCHIPInterpolation, 4*n, points, dataout)
 !
  do i=1,4*n
  do k=1,3
    points(k,i)=points(k,i)+dataout(k,i)*deltat
  end do
  end do
!
  if (it.eq.5.or.it.eq.10.or.it.eq.15) then
   do i=1,4*n
    write(10,*) points(1,i),points(2,i),points(3,i)
   end do
    write(10,*) points(1,1),points(2,1),points(3,1)
  endif
  write(10,*) '   '
  end do
!
  endif
```

advect and print-out results at subsequent times

4.3 sec. wall-time/time-for 1,000 particles



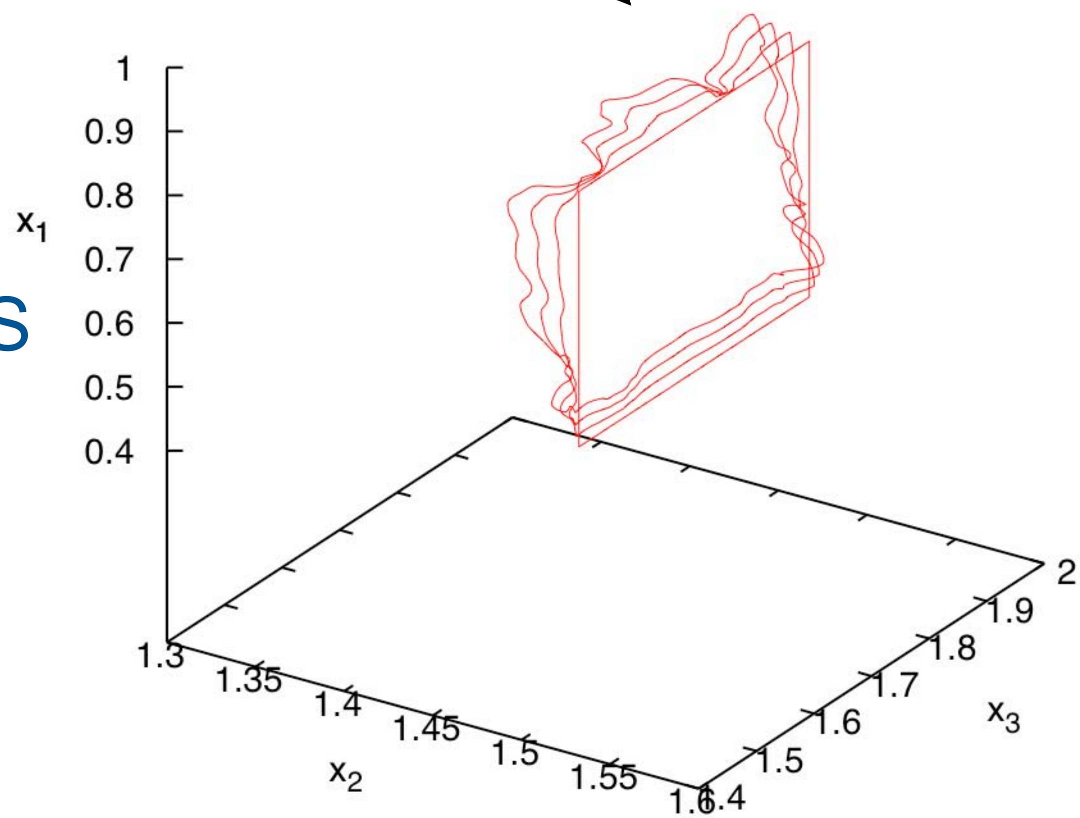
Sample code (gfortran 90) running on this Mac (unix)

```
!
 do it = 1,15,1
!
   print *,'time = ',time
   time=time-deltat
!
   CALL getvelocity(authkey, dataset1, time, Lagrangian6thOrder , PCHIPInterpolation, 4*n, points, dataout)
 !
  do i=1,4*n
  do k=1,3
     points(k,i)=points(k,i)-dataout(k,i)*deltat
  end do
  end do
!
  if (it.eq.5.or.it.eq.10.or.it.eq.15) then
   do i=1,4*n
     write(10,*) points(1,i),points(2,i),points(3,i)
   end do
     write(10,*) points(1,1),points(2,1),points(3,1)
  endif
  write(10,*) '  '
  end do
!
  endif
```

minus

advect backwards in time !

Not possible during DNS



Demo (gfortran 90) running on this Mac (unix)

advect 10 particles with and without inertia

(Euler 1st order)

$$\frac{d\mathbf{x}_f(t)}{dt} = \mathbf{u}(\mathbf{x}_f(t), t) \qquad \mathbf{x}_f^{n+1} = \mathbf{x}_f^n + \mathbf{u}(\mathbf{x}_f^n, t^n)\ \delta t$$

$$\frac{d\mathbf{v}_p(t)}{dt} = \frac{1}{\tau_p}\Big(\mathbf{u}(\mathbf{x}_p(t), t) - \mathbf{v}_p(t)\Big) \qquad \mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \mathbf{v}_p^n\ \delta t$$

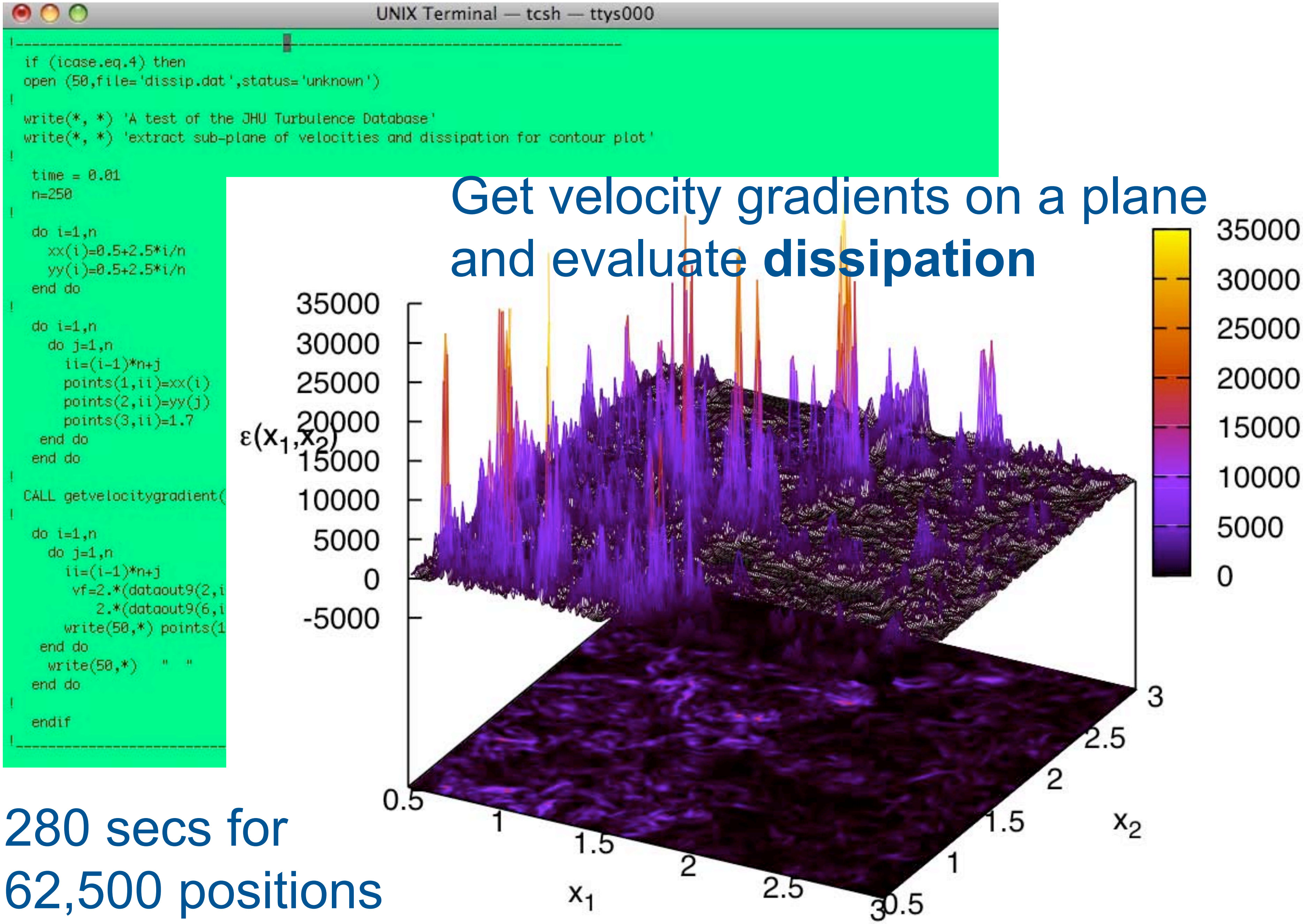
Sample code (gfortran 90) running on this Mac (unix)

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, Lagrangian6thOrder, NoTemporalInterpolation, n*n, points, dataout9)
!
   do i=1,n
     do j=1,n
       ii=(i-1)*n+j
        vf=2.*(dataout9(2,ii)+dataout9(4,ii))**2.+2.*(dataout9(7,ii)+dataout9(3,ii))**2.+ &
           2.*(dataout9(6,ii)-dataout9(8,ii))**2.+dataout9(1,ii)**2.+dataout9(5,ii)**2.+dataout9(9,ii)**2.
       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**

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



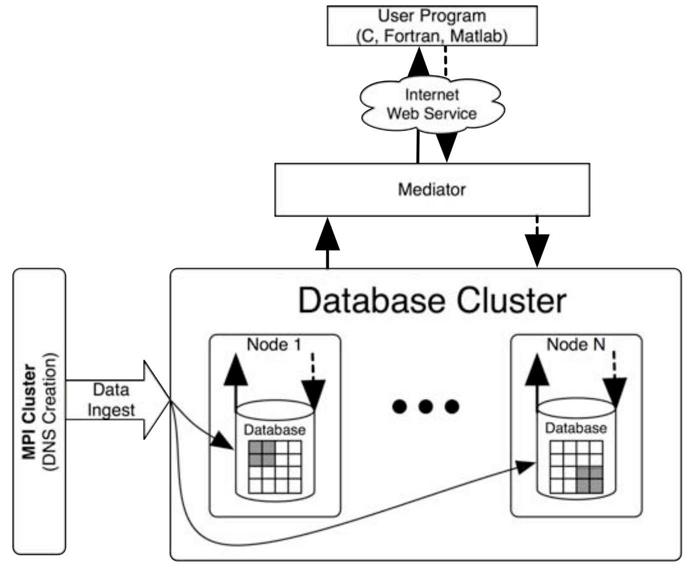
Get velocity gradients on a plane and evaluate **dissipation**

280 secs for 62,500 positions

Conclusions:

- **Built an environment well suited for explorations of turbulent flows**
- **Computationally expensive DNS are performed once and then ingested and archived for possible future use by *entire* community**
- **Generally applicable for analysis of time-dependent, "dense", 3-D vector and tensor fields, defined on a grid**
- **Brings the generic, common-place, computations** (differentiation, interpolation) **as close to the data as possible**
- **Maintains flexibility by letting user implement own computations**
- **But avoids having to give user accounts on server, HPC location, by providing instead a web service interfaces for Fortran, C, Matlab** (using gSOAP)
- **Cost-benefit: Slower analysis than when done "in core during DNS", but much more open and easily accessible.**

Outlook:



- **Complete the time-history - 27 Terabytes**
  (as of now: "840 more time-steps to ingest")

- **Further optimizations to accelerate service**

- **Develop additional functions, e.g. spatial filtering -**
  (but note: all-space transforms are a challenge in this approach)

- **Couple with visualization tools**

- **Include other datasets (other flows etc..)**

- **STAY TUNED AND VISIT US AT** http://turbulence.pha.jhu.edu

JOHNS HOPKINS
UNIVERSITY