Simulaciones numéricas directas en turbulencia de pared: Una visión global

Sergio Hoyas
Departamento de informática, Universidad de Valencia

Mark Simens, Javier Jiménez
ETSI Aeronáuticos, UPM

Funding:
DEISA, BSC,CICYT,PIC

¡9 Millones de horas!

¿Para qué y por qué necesitamos 9e6 horas y 50 Tb?

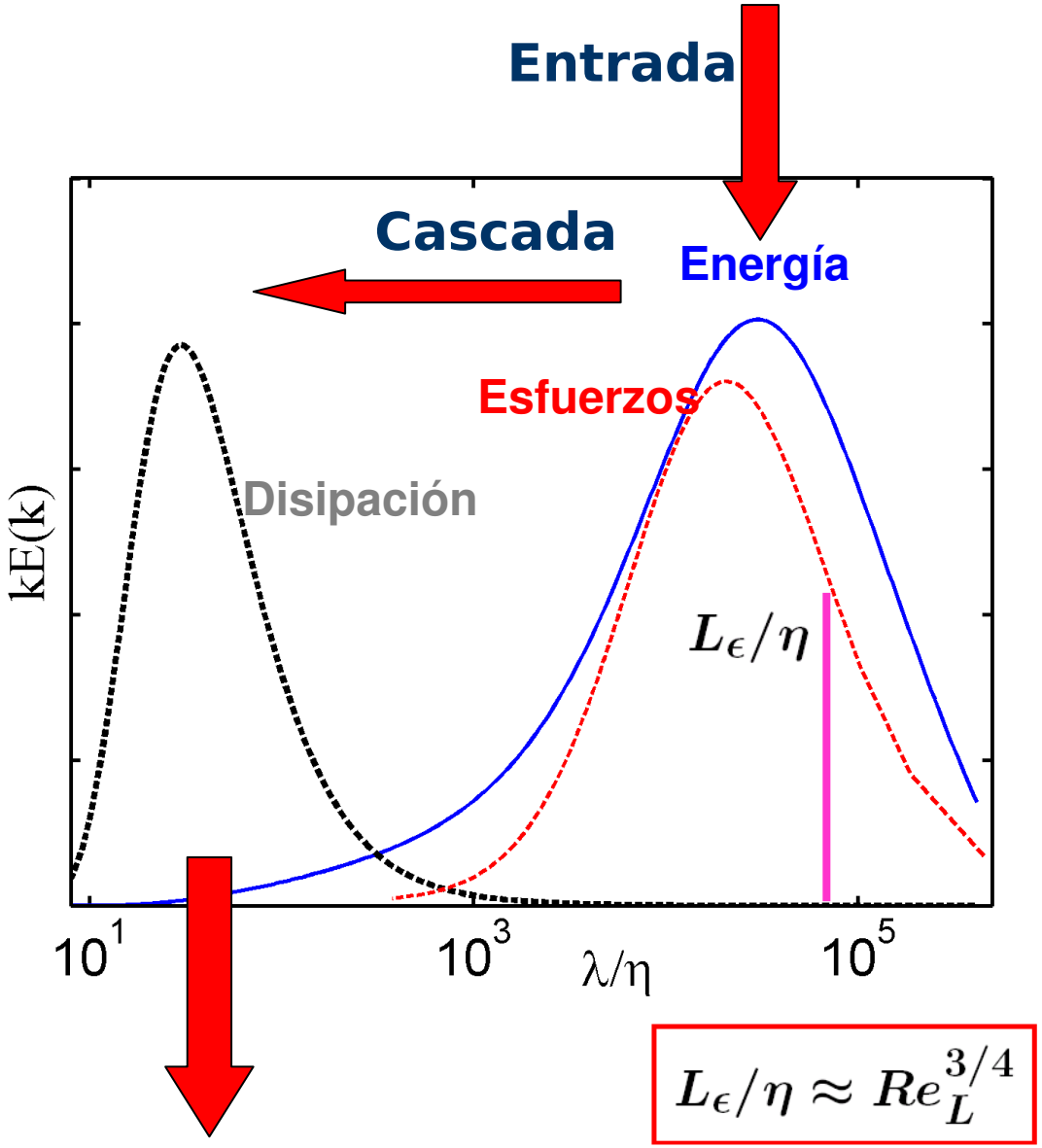
Cascada de Energía

Kolmogorov (1941)

Big whorls have little whorls
that feed on their velocity,
and little whorls have lesser whorls
and so on to viscosity.

Richardson

Flujo de energía en turbulencia isótropa



$$L_\epsilon/\eta \approx Re_L^{3/4}$$

$$\text{Dissipation: } 5 - 100\eta$$
$$\text{Energy: } 0.01 - 50L_\epsilon$$
$$\text{Stress: } 0.1 - 20L_\epsilon$$

Flujos industriales típicos

$$Re_L = 6000 - 1000000$$

Estela de una persona caminando

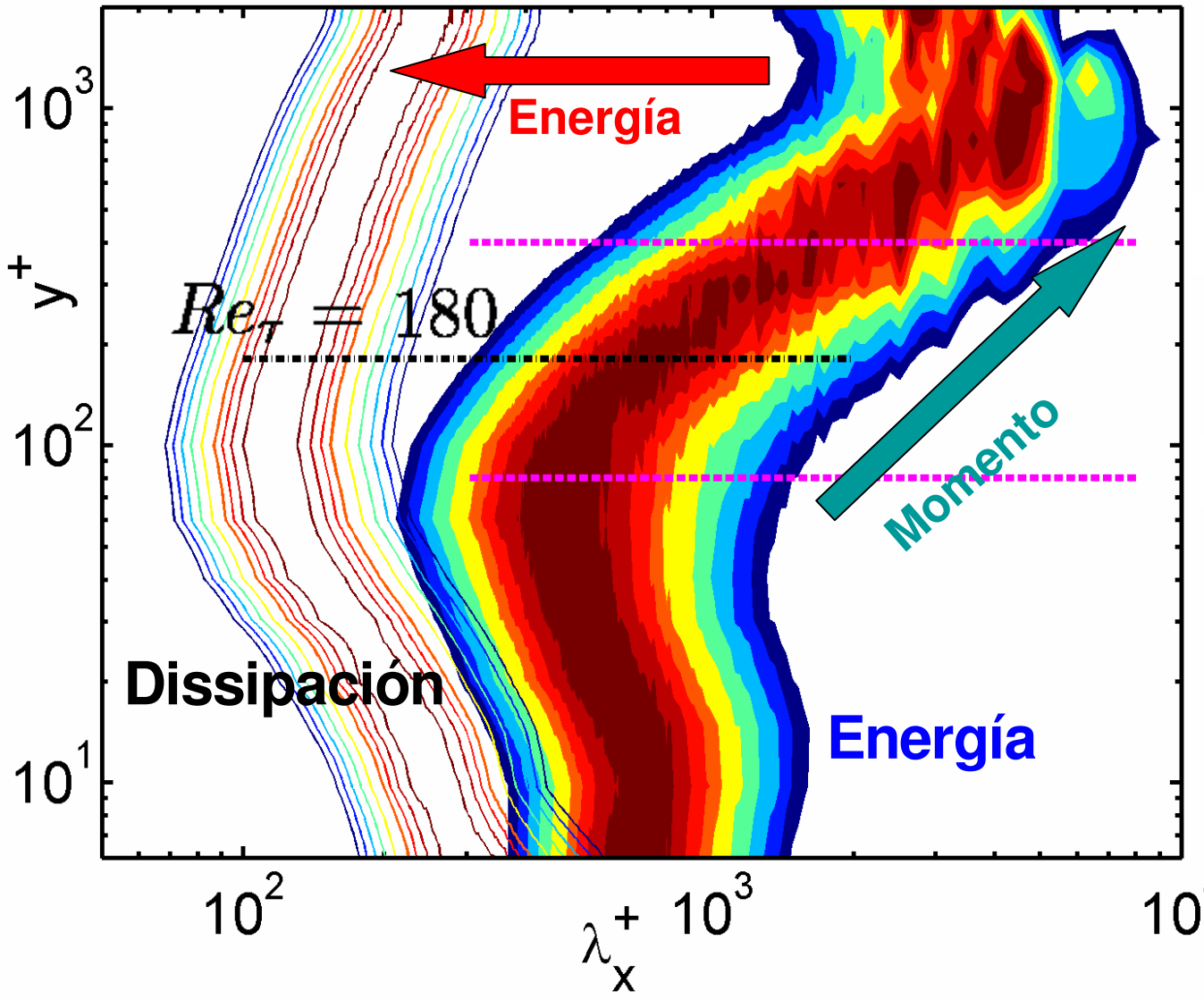
$$Re_L = 16000$$

Capa límite de un avión

$$Re_L = 6000000$$

Cascadas en la turbulencia de pared

$$Re_\tau = u_\tau h/\nu = 2000$$



Exterior

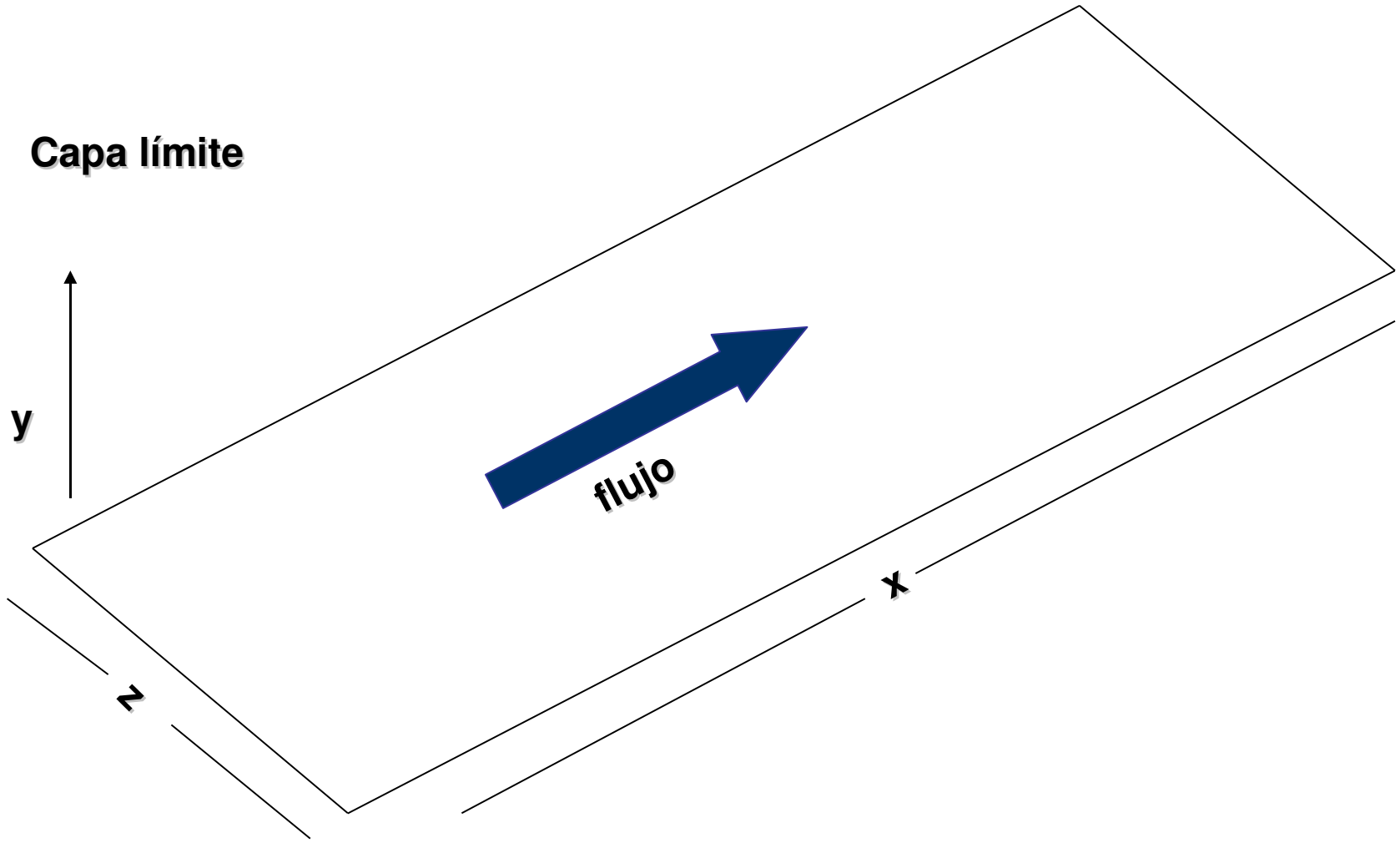
$y < 0.2h = 400^+$

Log

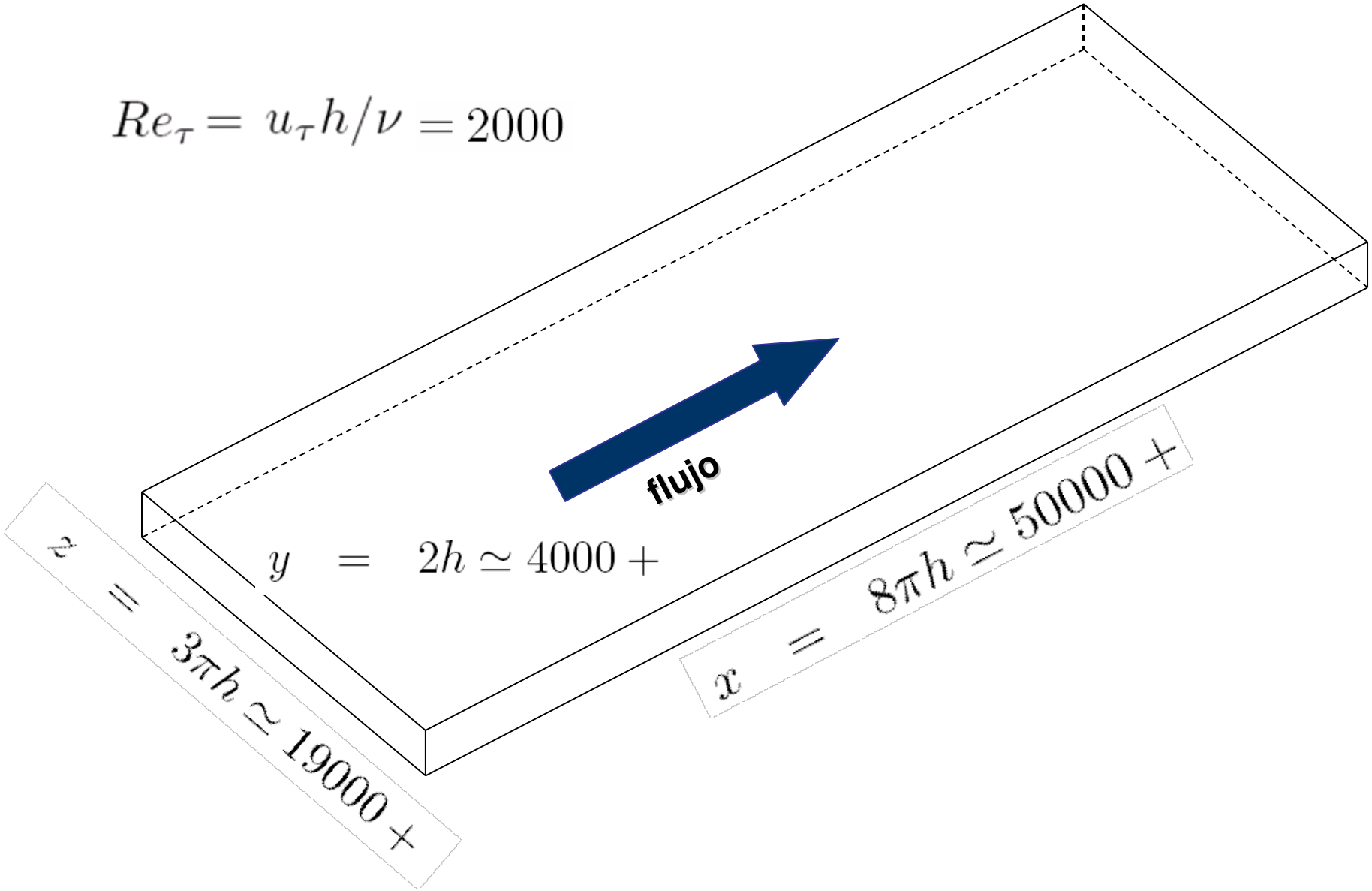
$y = 80^+$

Viscosa

Dominio



Dominio


$Re_\tau = u_\tau h/\nu = 2000$
flujo
$y = 2h \simeq 4000+$
$x = 8\pi h \simeq 50000+$
$z = 3\pi h \simeq 19000+$

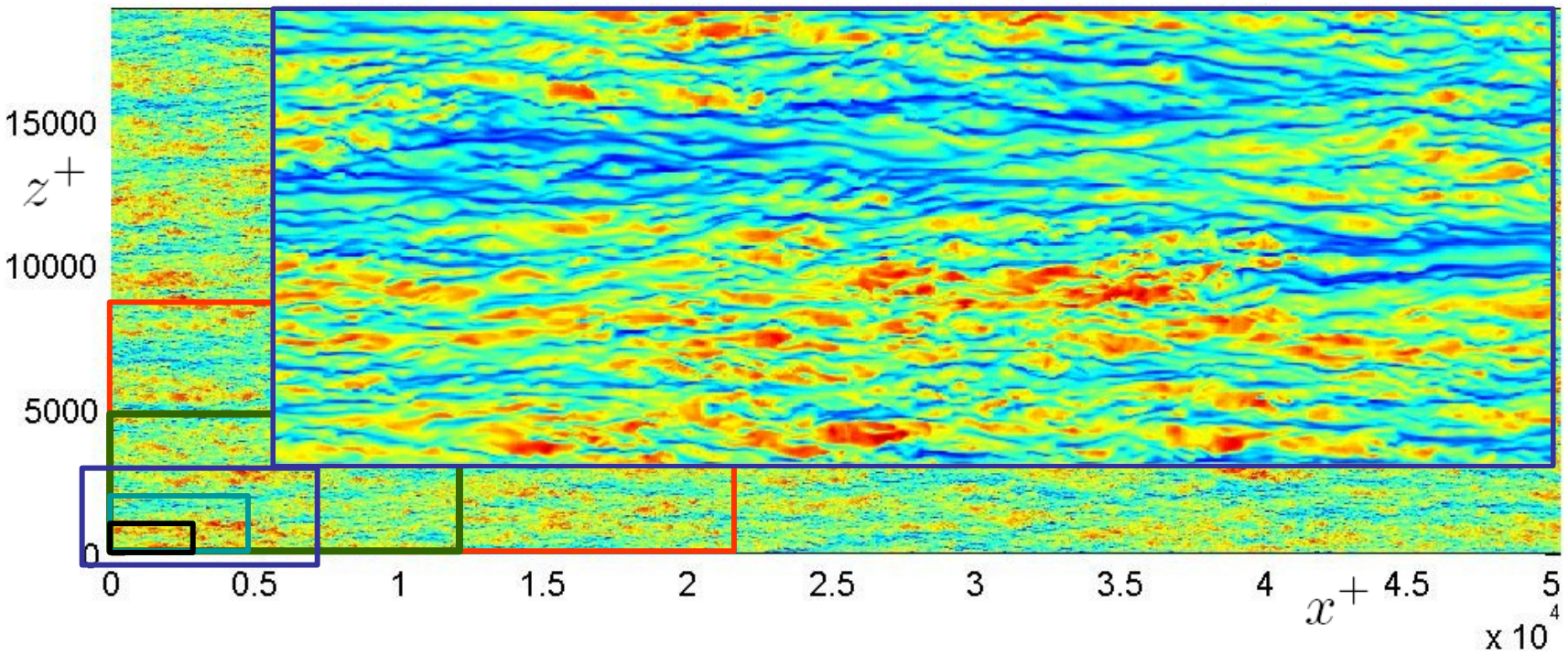
Mallado del canal

# Puntos (Fis.)	# Puntos (Fourier, R.)	# Puntos (Fourier, C.)
$N_x = 6144 = 2^{11} \times 3$	4096	2048
$N_y = 633$		
$N_z = 4608 = 2^9 \times 9$		3072

Espacio físico	Espacio de Fourier	
$\Delta x^+ \simeq 8.18$	$\Delta x^+ \simeq 12.27$	$\min(\Delta y^+) \simeq 0.32$
$\Delta z^+ \simeq 4.1$	$\Delta z^+ \simeq 6.13$	$\max(\Delta y^+) \simeq 8.88$

Memoria total 400GB (simple precision).

Simulaciones anteriores



Kim, Moin and Moser, 1987, 180 (Cray XMP, NASA Ames)

Del Álamo and Jiménez, 2003 (Kadesh, CEPBA) 180

Del Álamo and Jiménez, 2003 (Kadesh, CEPBA) 550

Del Álamo, Moser, Jiménez and Zandonade, 2004 (Blue Horizon) 950

Simulaciones anteriores

Ecuaciones de Navier-Stokes

$$
\begin{aligned}
\frac{\partial u_i}{\partial t} &= -\frac{\partial p}{\partial x_i} + H_i + \frac{1}{Re}\nabla^2 u_i, \\
\frac{\partial u_i}{\partial x_i} &= 0, \\
\\
\vec{u} &= (u_1, u_2, u_3) = (u, v, w) \\
\vec{\omega} &= \vec{\nabla} \times \vec{u} = (\omega_1, \omega_2, \omega_3) = (\omega_x, \omega_y, \omega_z) \\
\vec{H} &= (\vec{u} \times \vec{\omega}) - \frac{1}{2}\nabla\left(\vec{u} \cdot \vec{u}\right) = (H_1, H_2, H_3)
\end{aligned}
$$

Forma Velocidad-Vorticidad

$$\frac{\partial}{\partial t}\phi = h_v + \frac{1}{Re}\nabla^2\phi \qquad \phi = \nabla^2 v$$

$$\frac{\partial}{\partial t}\omega_y = h_g + \frac{1}{Re}\nabla^2\omega_y$$

70-80% of time

99% of communication

$$h_g = \frac{\partial H_1}{\partial z} - \frac{\partial H_3}{\partial x}$$

$$h_v = -\frac{\partial}{\partial y}\left(\frac{\partial H_1}{\partial x} + \frac{\partial H_3}{\partial z}\right) + \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial z^2}\right)H_2$$

Discretización en x y z

Discretización

Espacial: Fourier en x y z, diferencias finitas compactas en z

$$\varphi(x,y,z) = \sum_{\alpha'=0}^{N_x-1} \sum_{\beta'=0}^{N_z-1} \varphi_{\alpha',\beta'}(y) e^{ic_\alpha \alpha' x} e^{ic_\beta \beta' z}$$

con

$$c_\alpha = 2\pi L_y / L_X \qquad c_\beta = 2\pi L_y / L_z$$

tomando

$$k_x = c_\alpha \alpha' \qquad k_z = c_\beta \beta'$$

$$\varphi(x,y,z) = \sum_{kx} \sum_{kz} \varphi_{k_x,k_z}(y)\, e^{(k_x x + k_y y)i}$$

Ecuaciones en el espacio de Fourier

$$
\begin{aligned}
\partial_t \hat{\omega}_y &= \hat{h}_g - \frac{1}{Re}\left(k_x^2 + k_z^2 - \partial_{yy}\right)\hat{\omega}_y \\
\hat{\omega}_y\left(\pm 1\right) &= 0
\end{aligned}
$$

$$
\begin{aligned}
\partial_t \hat{\phi} &= \hat{h}_v + \frac{1}{Re}\nabla^2 \hat{\phi} \\
\nabla^2 \hat{v} &= \hat{\phi}, \\
\partial_n \hat{v}\left(\pm 1\right) &= 0, \hat{v}\left(\pm 1\right) = 0
\end{aligned}
$$

FFT. El problema del dealiasing

Problema clásico de la turbulencia: como calcular por ejemplo

$$H_1 = (v\omega_z - w\omega_y)$$

$$\begin{cases} v^j = \sum_{k=-N/2}^{N/2-1} \hat{v}_k e^{ikx_j} \\ \omega_z^j = \sum_{k=-N/2}^{N/2-1} \hat{\omega}_k^z e^{ikx_j} \end{cases} \qquad j = 0, 1, ..., N-1 \quad \Rightarrow \quad H_{1,j}^1 = v^j \omega_z^j,$$

como

$$\hat{H}_{1,k}^1 = \frac{1}{N} \sum_{j=0}^{N-1} H_{1,j}^1 e^{-ikx_j}, k = -\frac{N}{2}, ..., \frac{N}{2} - 1$$

$$\hat{H}_{1,k}^1 = \sum_{m+n=k} \hat{v}_k \hat{\omega}_k^z + \sum_{m+n=k\pm N} \hat{v}_k \hat{\omega}_k^z$$

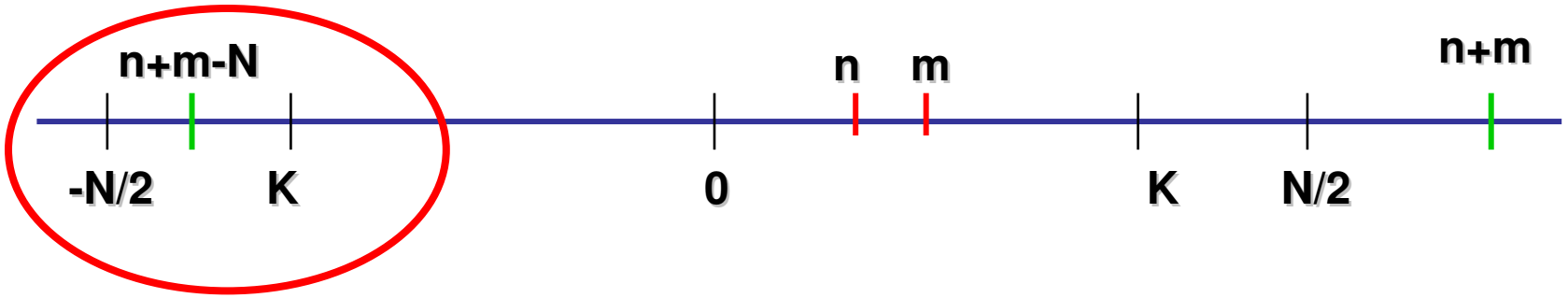
Error de aliasing

Dealiasing

Dos formas de evitarlo: phase shifts y truncación o 3/2

$$\tilde{v}_k = \begin{cases} \hat{v}_k & |k| \leq N/2 \\ 0 & N/2 < |k| \leq (2/3)\, N/2 \end{cases}$$

Truncación: agrandamos la transformada



$$\text{Si } m+n > N/2 \Rightarrow m+n-N/2 \in (-N/2, -K)$$
$$\text{Pedimos } m+n-N/2 < -K \Rightarrow 2K - N/2 < -K \Rightarrow K < N/3$$

Condición de Neumann para v

$$\partial_y v\,(\pm 1) = 0 \quad \longrightarrow \quad v = v_p + c_1 v_1 + c_2 v_2,$$

$$\partial_t \hat{\phi}_1 - \frac{1}{Re}\left(k_x^2 + k_z^2 - \partial_{yy}\right)\hat{\phi}_1 = 0, \qquad \nabla^2 \hat{v}_1 = \hat{\phi}_1,$$

$$\hat{v}_1\,(\pm 1) = 0,\ \phi_1\,(1) = 0, \phi_1\,(-1) = 1.$$

$$\partial_t \hat{\phi}_p = \hat{h}_v + \frac{1}{Re}\left(k_x^2 + k_z^2 - \partial_{yy}\right)\hat{\phi}, \qquad \nabla^2 \hat{v}_p = \hat{\phi}_p,$$

$$\hat{v}_p\,(\pm 1) = 0,\ \phi_p\,(\pm 1) = 0$$

~~$$\partial_t \hat{\phi}_2 - \frac{1}{Re}\left(k_x^2 + k_z^2 - \partial_{yy}\right)\hat{\phi}_2 = 0,$$~~

~~$$\hat{v}_2\,(\pm 1) = 0,\ \phi_2\,(1) = 0,\ \phi_2\,(-1) = 1,$$~~

~~$$\nabla^2 v_2 = \hat{\phi}_2.$$~~

$$v_2(y) = v_1(-y)$$

Discretización

Método: Runge-Kutta de tercer orden (Spalart et al, 1991)

$$\partial_t u = L(u) + N(u)$$

$$\begin{aligned}
u^1 &= u^0 + \Delta t \left[L\left(\alpha_1 u^0 + \beta_1 u^1\right) + \gamma_1 N^0 \right] \\
u^2 &= u^1 + \Delta t \left[L\left(\alpha_2 u^1 + \beta_2 u^2\right) + \gamma_2 N^1 + \xi_1 N^0 \right] \\
u^3 &= u^2 + \Delta t \left[L\left(\alpha_3 u^2 + \beta_3 u^3\right) + \gamma_3 N^2 + \xi_2 N^1 \right]
\end{aligned}$$

$$R_i = \frac{Re}{\beta_i \Delta t}, \quad Rk_i = \left(R^i + k_x^2 + k_z^2\right), \quad \rightarrow \textbf{118 e6 ecuaciones/paso}$$

$$\begin{aligned}
\left(\partial_{yy} - Rk_1\right) u^1 &= -R_1 \left[u^0 + \Delta t \left(\alpha_1 L u^0 + \gamma_1 N^0 \right) \right], \\
\left(\partial_{yy} - Rk_2\right) u^2 &= -R_2 \left[u^1 + \Delta t \left(\alpha_2 L u^1 + \gamma_2 N^1 + \xi_1 N^0 \right) \right], \\
\left(\partial_{yy} - Rk_3\right) u^3 &= -R_3 \left[u^2 + \Delta t \left(\alpha_3 L u^2 + \gamma_3 N^2 + \xi_1 N^1 \right) \right],
\end{aligned}$$

Discretización - y

Normal: Diferencias finitas compactas (Lele, 1991)

Primera derivad: malla de 7 puntos. Mapeada a la original

$$D_j + \sum_{m=1}^{M} b_m (D_{j+m} + D_{j-m}) = \frac{1}{h} \sum_{n=1}^{N} a_n (u_{j+n} - u_{j-n})$$

N=M=7

Segunda derivada: malla real

$$D_j^{(2)} + \sum_{m=1}^{M} b_m (D_{j+m}^{(2)} + D_{j-m}^{(2)}) = \frac{a_0 u_j + \sum_{n=1}^{N} a_n (u_{j+n} + u_{j-n})}{h^2}$$

N=M=5

Sistemas: métodos LU, sin pivotaje, adaptadas de “Numerical Recipes”

Discretización - y

$$\underbrace{\begin{pmatrix} * & * & * & & & & & & \\ * & * & * & * & & & & & \\ * & * & * & * & * & & & & \\ & * & * & * & * & * & & & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & & \\ & & & * & * & * & * & * & \\ & & & & * & * & * & * & * \\ & & & & & * & * & * & * \\ & & & & & & * & * & * \end{pmatrix}}_{D} \begin{pmatrix} u_1'' \\ u_2'' \\ \\ \\ \vdots \\ \\ \\ \\ u_n'' \end{pmatrix} = \underbrace{\begin{pmatrix} * & * & * & & & & & & \\ * & * & * & * & & & & & \\ * & * & * & * & * & & & & \\ & * & * & * & * & * & & & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & & \\ & & & * & * & * & * & * & \\ & & & & * & * & * & * & * \\ & & & & & * & * & * & * \\ & & & & & & * & * & * \end{pmatrix}}_{A} \begin{pmatrix} u_1 \\ u_2 \\ \\ \\ \vdots \\ \\ \\ \\ u_n \end{pmatrix}$$

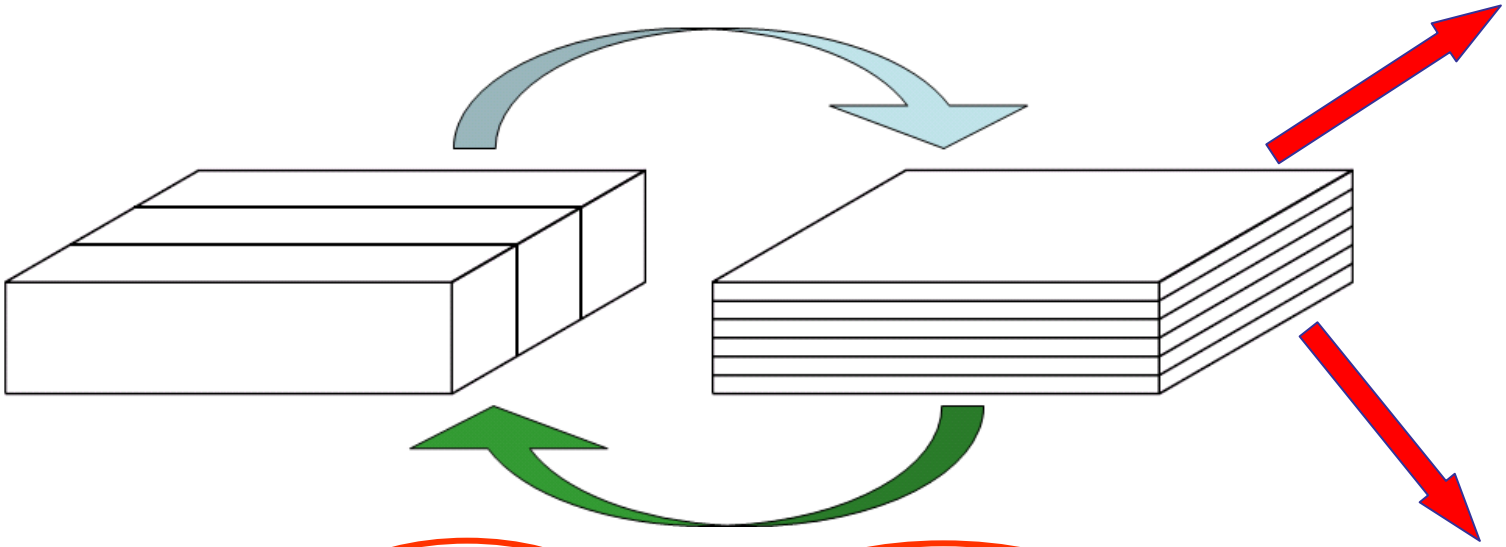
$$\begin{aligned} u^{n+1} - K\partial_{yy}u^{n+1} &= f(u^n) \\ Du^{n+1} - K\left(D\partial_{yy}u^{n+1}\right) &= Df(u^n) \\ Du^{n+1} - KAu^{n+1} &= Df(u^n) \\ \boxed{\left(D - KA\right)u^{n+1} &= Df(u^n)} \end{aligned}$$

Sistemas: métodos LU, sin pivotaje, adaptadas de “Numerical Recipes”

Esquema clásico de paralelización

$$\vec{H} = \vec{F}\left(v, \partial_y v_y, \phi, \omega_y, \partial_y \omega_y\right)$$

Solo podemos usar N_y procs



$$h_v = -\frac{\partial}{\partial y}\left(\frac{\partial H_1}{\partial x} + \frac{\partial H_3}{\partial z}\right) + \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial z^2}\right) H_2$$

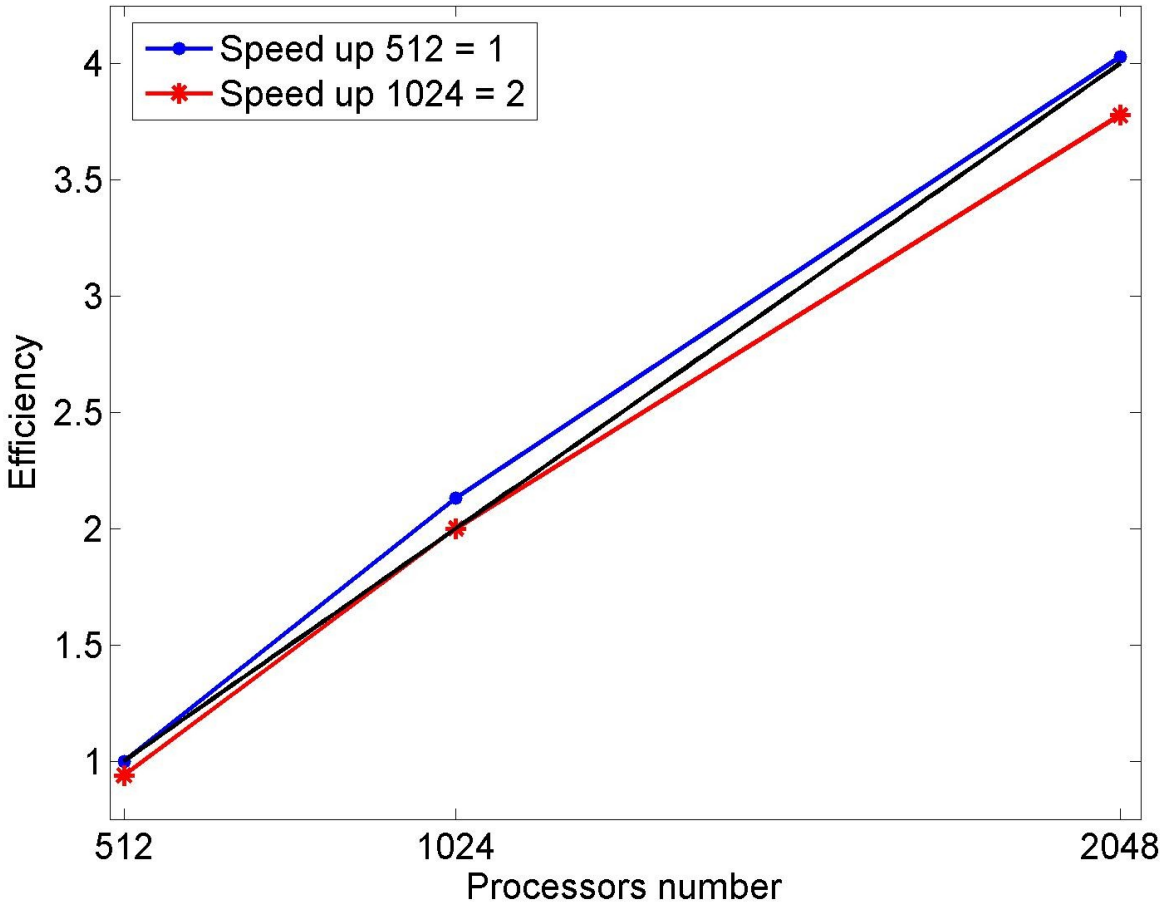
$$h_g = \frac{\partial H_1}{\partial z} - \frac{\partial H_3}{\partial x}$$

Problema dealiasing 2D

$$Re_\tau = 2000 \Rightarrow 650MB$$

$$Re_\tau = 4000 \Rightarrow 2.4GB$$

Speed-up del nuevo esquema



Importante

- Optimización de rutinas

Adaptado:

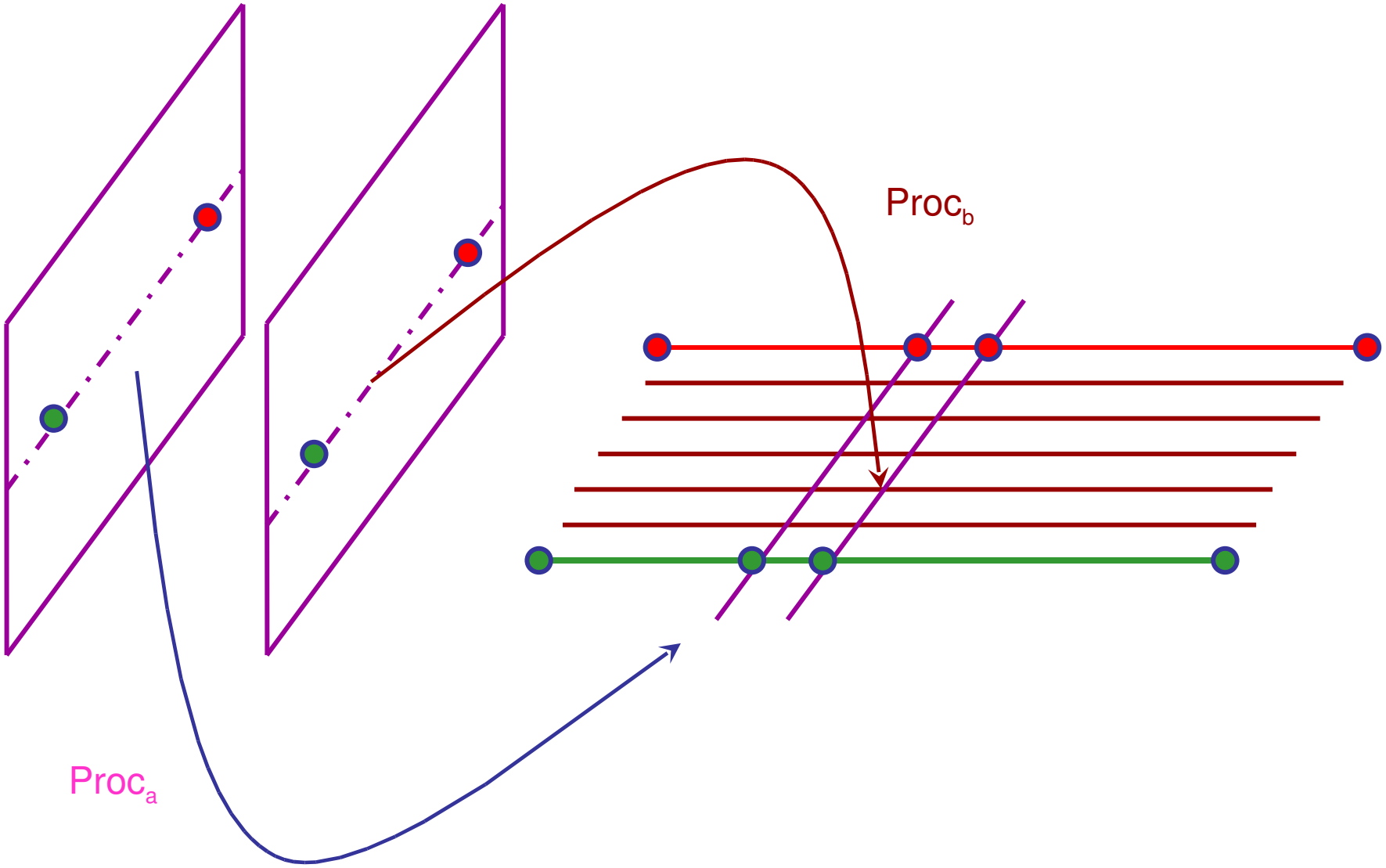
- XLF

- Arquitectura de MareNostrum

Claves

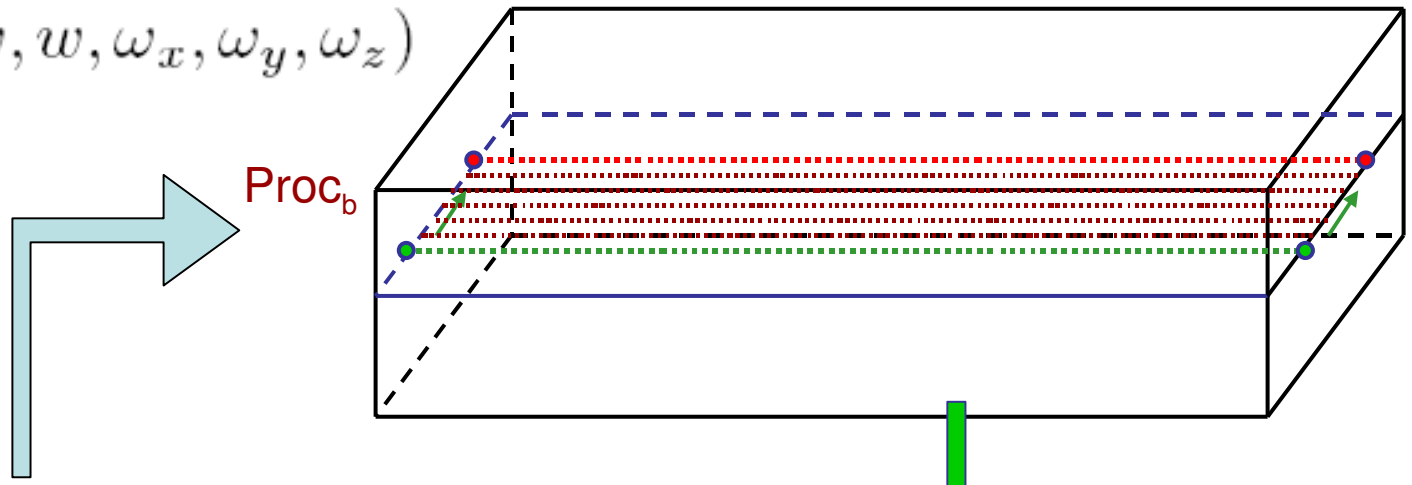
- **Input/ouput**
- **Comunicaciones**

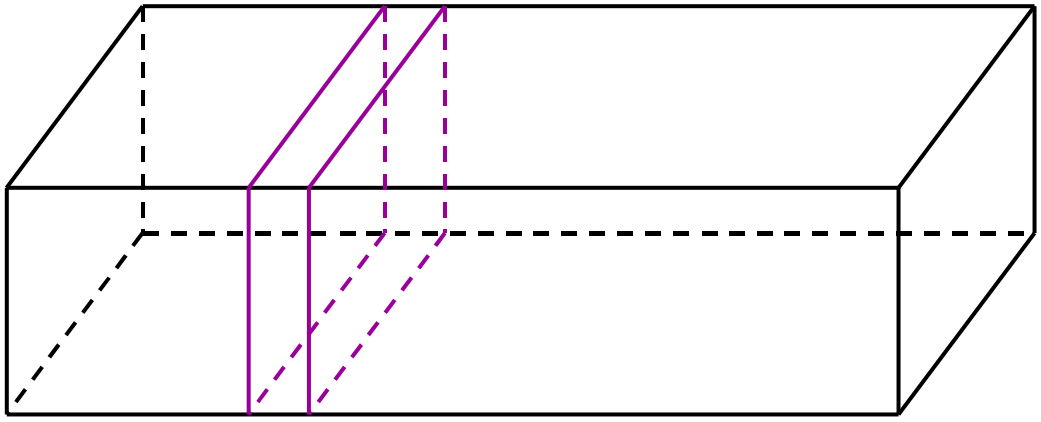
Esquema en lineas-planos



Paralelización en planos-líneas

$$\vec{H} = \vec{F}\left(u, v, w, \omega_x, \omega_y, \omega_z\right)$$

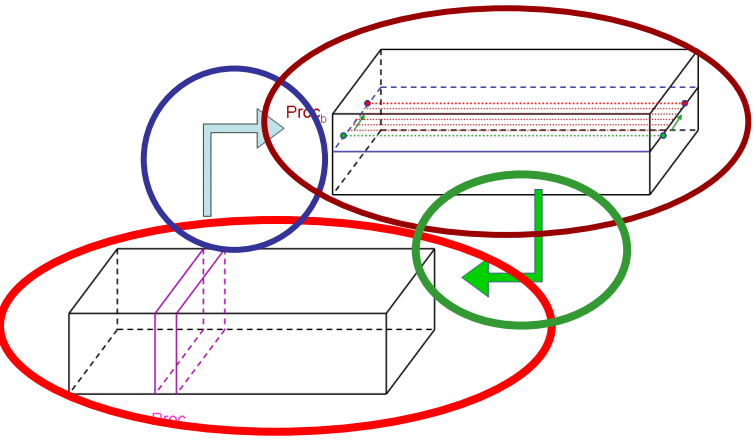




$$h_g = \frac{\partial H_1}{\partial z} - \frac{\partial H_3}{\partial x}$$

$$h_v = -\frac{\partial}{\partial y}\left(\frac{\partial H_1}{\partial x} + \frac{\partial H_3}{\partial z}\right) + \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial z^2}\right) H_2$$

Paralelización en lineas planos: esquema



1.- 10%

2.- 40% (133MB)

3.- 10%

4.- 20% (66 MB)

5.- 20%

Primera parte

1.- Calculamos vel. y vort. (F-P-F)
2.- Transformamos z al espacio físico

Segunda parte

Movemos de yz a líneas en x

Tercera parte

1.- Transformamos x a físico
2.- Cálculo de la helicidad
3.- Transfomamos la helicidad a Fourier

Cuarta parte

Movemos de líneas en x a yz

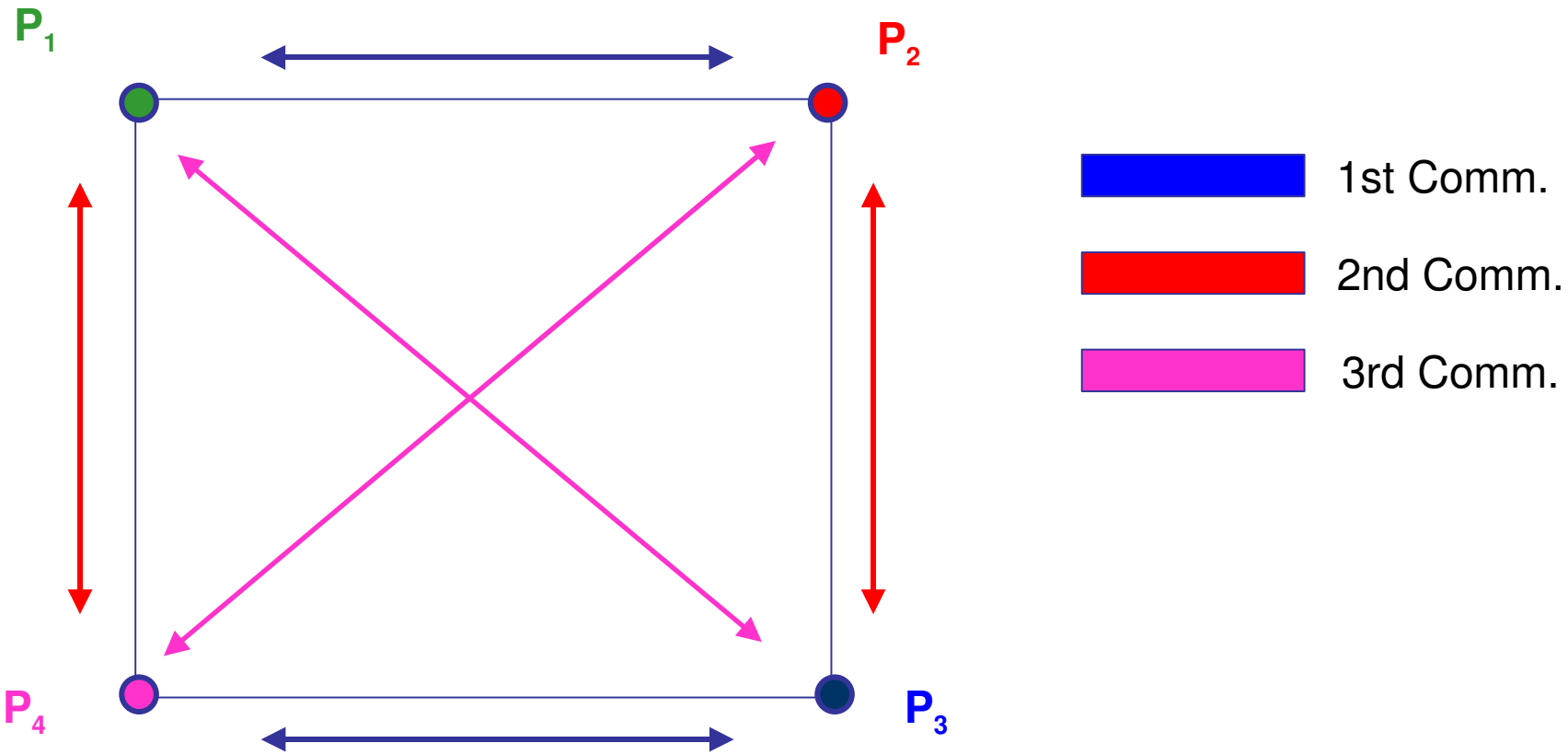
Quinta parte

1.- Transformamos a (F-P-F)
2.- Calculo del RHS de la ecuación
3.- Resolvemos los sistemas
4.- Avanzamos en tiempo

Topología de hipercubo para las comunicaciones

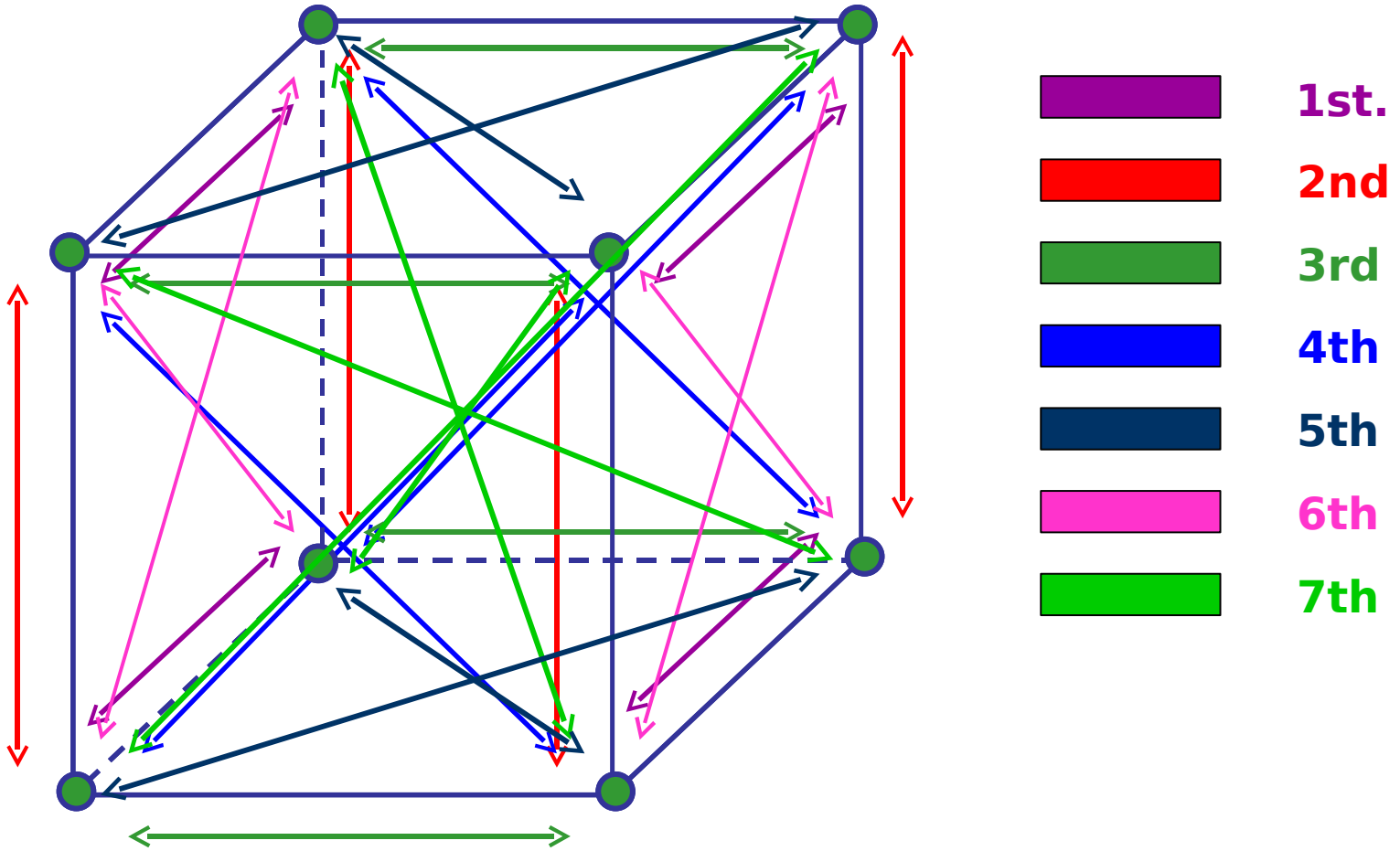
1. Si A está mandando datos a B, B tiene que estar esperando a A.
2. Nadie más tiene que comunicarse con A o B

#Procs = $4=2^2$



Hipercubo

#Procs = $8=3^2$



Hipercubo: seudocódigo

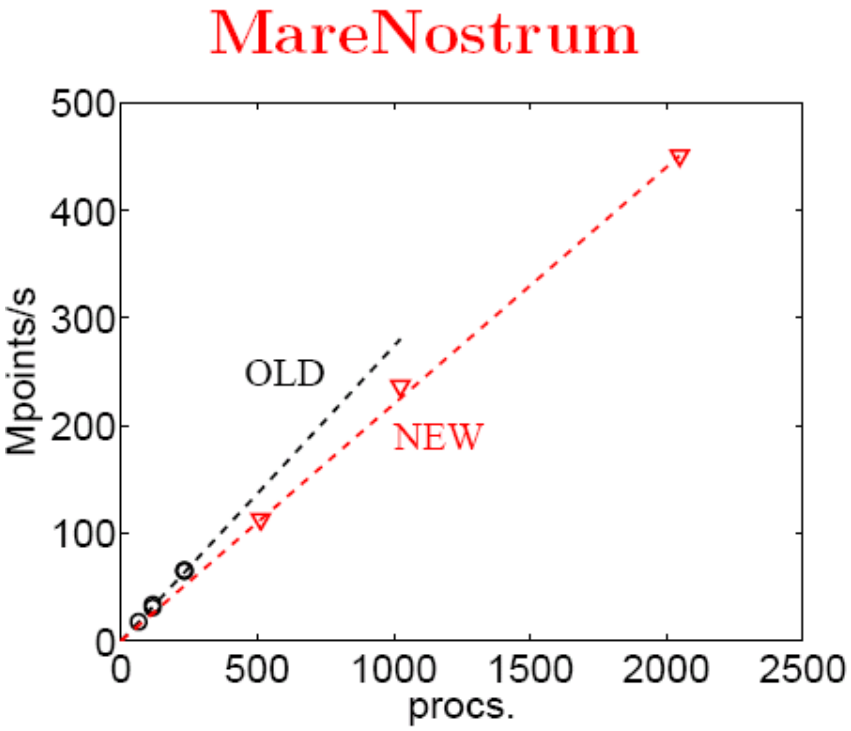
```
 ! numerop = number of processors
 ! pnodes = 2**n, pnodes >= numerop

 do pproc=1,pnodes-1
    iproc=ieor(myid,pproc)
    if (iproc<numerop) then
       do ii=pb,pe
          call MPI SENDRECV(bs,nsend,MPI REAL,iproc,0,
.                           bi,nrecv,MPI REAL,iproc,0,
.                           MPI_COMM_WORLD,istat,ierr)
       enddo
    endif
 enddo
```

Wrong!!!

```
 do pproc=1,numerop-1
    do ii=pb,pe
       call MPI SENDRECV(bs,nsend,MPI REAL,iproc,0,
                         bi,nrecv,MPI REAL,iproc,0,
.                        MPI_COMM_WORLD,istat,ierr)
    enddo
 enddo
```

Comparación entre códigos



OLD CODE

9 Fourier buffers

8 buffer transposes

Comm. Time/Total = 28%

$$N_{proc} \leq N_y$$

NEW CODE

12.5 Fourier buffers

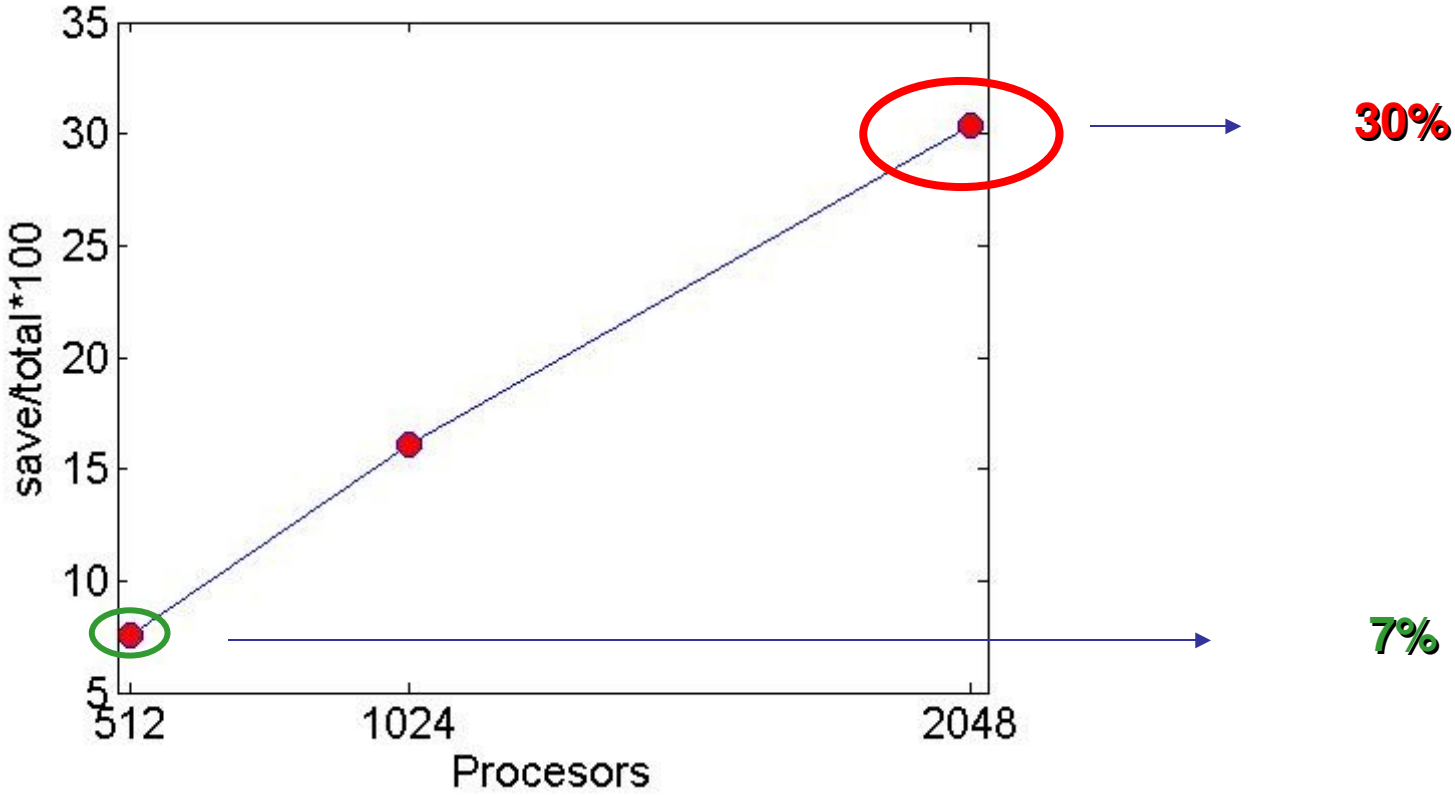
13.5 buffer transposes

Comm. Time/Total = 58%

$$N_{proc} \leq N_x/3$$

Entrada/salida de datos

- El código salva una imagen cada 200 pasos.
- El archivo pesa alrededor de 60GB
- Implementación clásica: esclavos a maestro
- Todos los procesadores tienen que esperar a que el archivo esté escrito

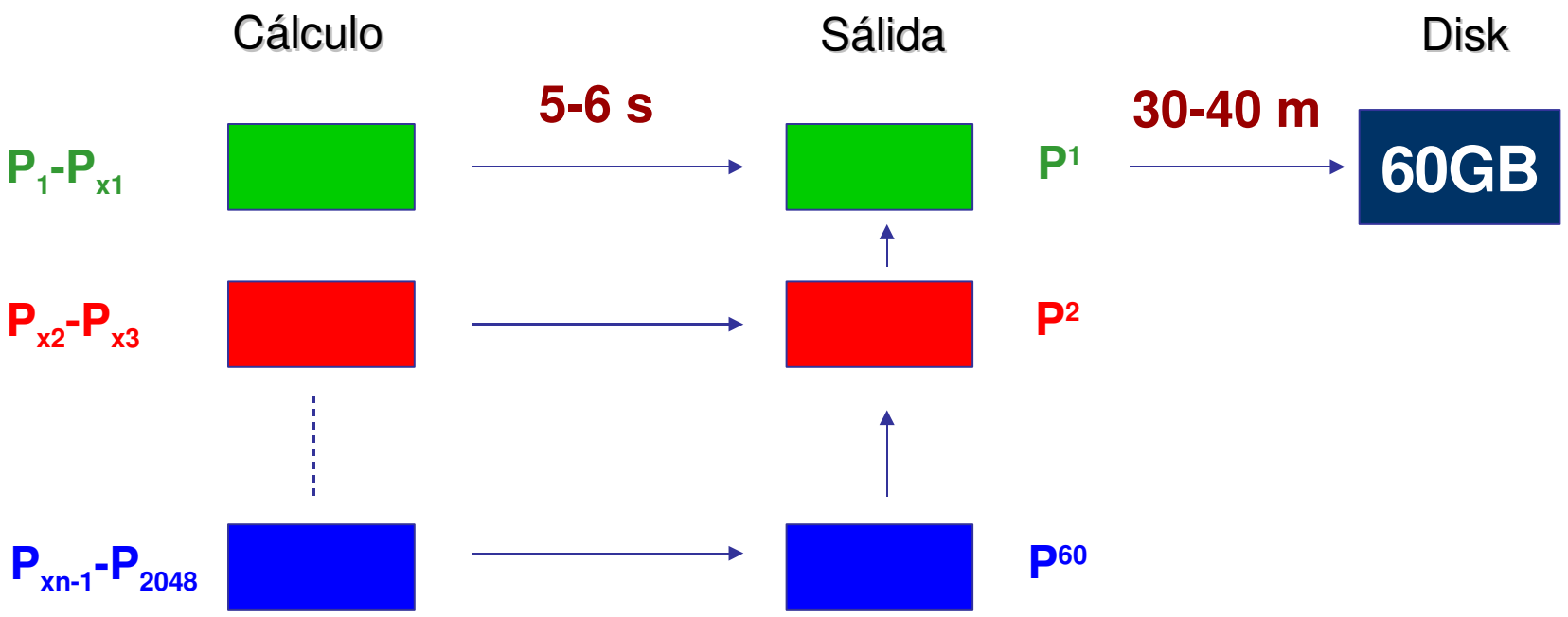


Dos soluciones

Propuesta por MN:

Cada procesador escribe su plano a un directorio
- No se implemento
- Problemas: 2048 procesadores intentando escribir al mismo tiempo.
  2048 archivos generados por cada imagen, 600.000 en total

Implementada: Pedimos 60 procs más y creamos dos MPI_GROUPS



Simulaciones cinemáticas de canales

NEAR-WALL

$$Re_\tau \gg 60$$

LOG LAYER

$$0.2Re_\tau \gg 60$$

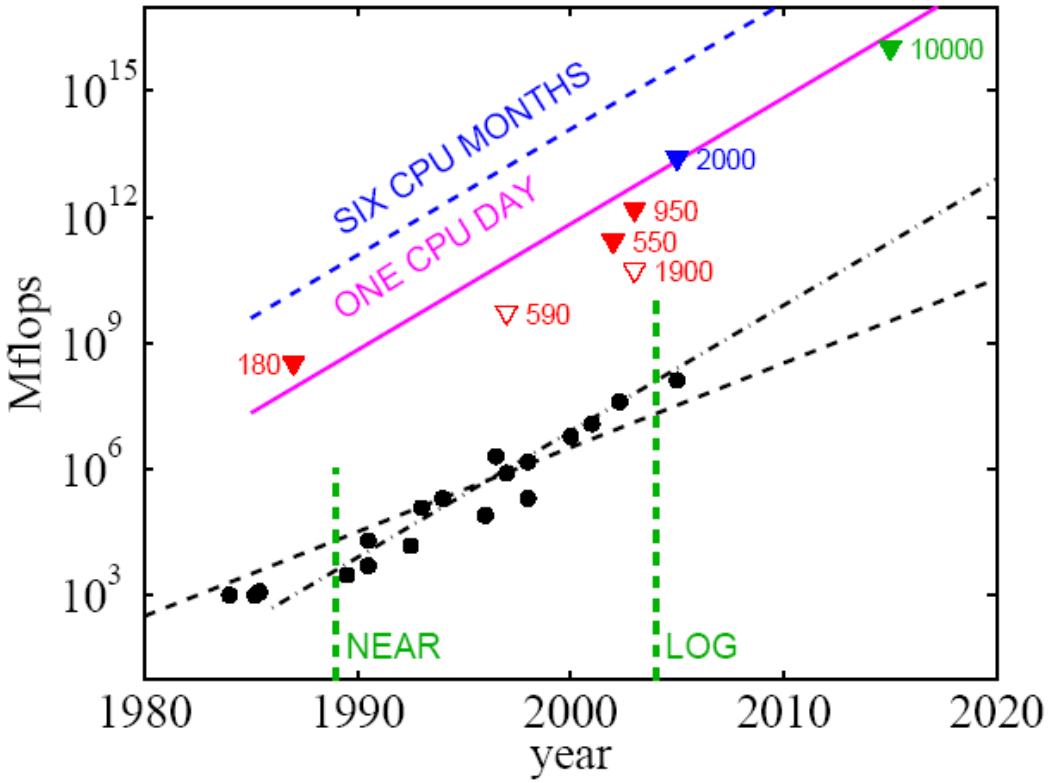
$$Re_\tau \gtrsim 1000$$

ASYMPTOTIC

Largest $\gg$ Streaks

$$10Re_\tau \gg 1000$$

$$Re_\tau \gtrsim 1000$$



!DNS son muy caras!

	Procesador	Total
Memoria	**0.2GB**	**400GB**
Pasos	***125.000***	***125.000***
Tiempo por cada paso del Runge-Kutta	**40s**	**40s**
CPU-hours totales	***2800h***	***6e6h (1.3e6)***
Horas humanas totales	**4 months**	**4months**
Transferencia de datos entre procesadores	***0.6GB***	***1.2PB***
Total de datos transmitidos	**73PB**	**145EB**
Base de datos obtenidad	***25TB***	***25TB***
Flops conseguidos	**50GF**	**3TF**
Flops totales	***18.3PF***	***3.6EF***

Mare Nostrum

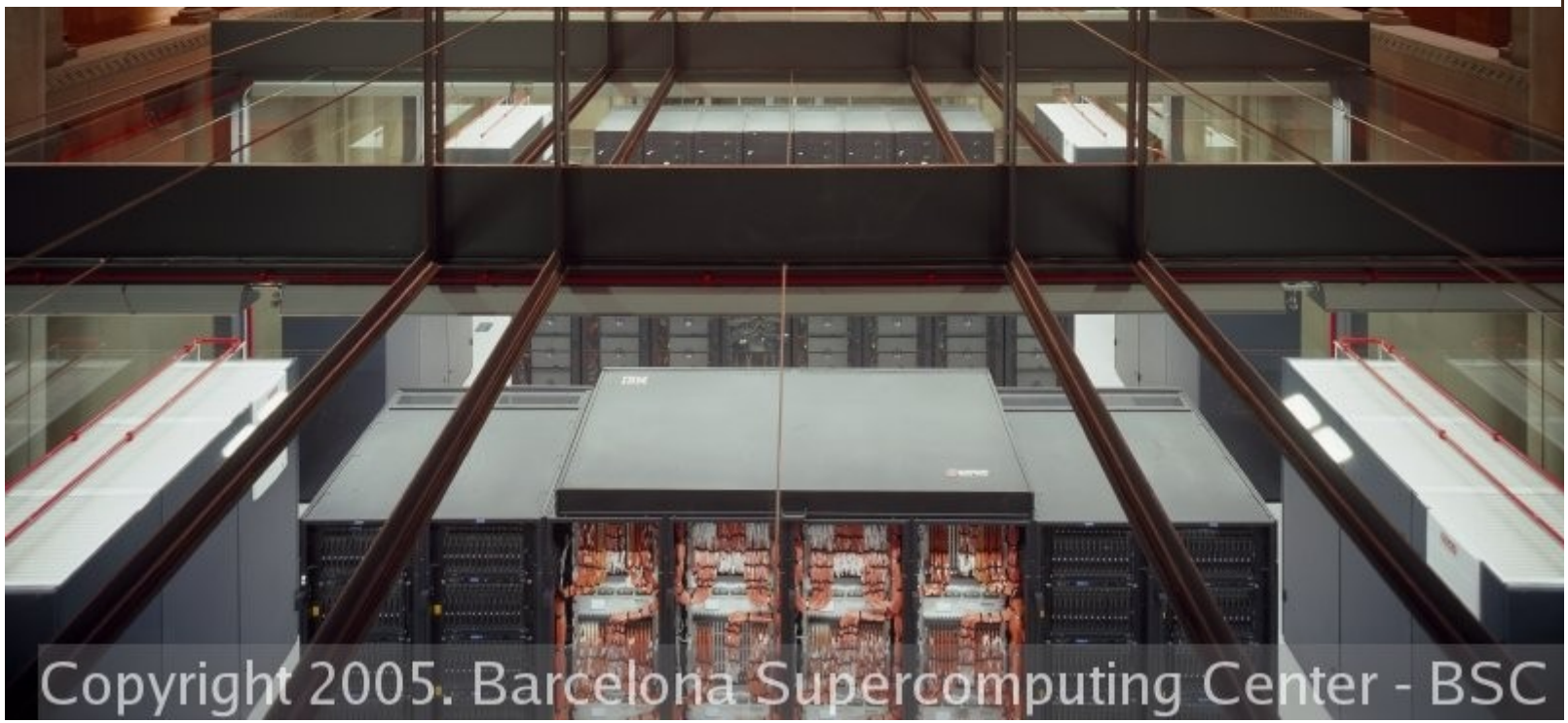
Supercomputador perteneciente al centro nacional de supercomputación.

4812 procesadores PowerPC 970FX a 2,2 GHz

9.6 TB de memoria ram. 236 TB de disco

Potencia mantenida de 38 Tflops.

Información: www.bsc.es



Agradecimientos

Almacenamiento: Port d'informacio cientifica,ww.pic.es. Castor: Capacitad 1.5 PB. Idea del centro: Llegar a 10PB

Agradecimientos

BSC

José María Cela: FFTW and general optimization.

Sergi Girona: Input/output routines and many discussions about MareNostrum.

Jesús Labarta: Communications routines.

5.000.000 CPU-H assigned to run the simulation.

PIC

Manuel Delfino: Storage of the results of the simulations.

25 TB of permanent storage.

DEISA

800.000 CPU-H assigned through a project.

Conclusions

- We have made a new algorithm for a DNS of a turbulent channel flow, capable of use several thousands of processors, showing an excellent speed-up.
- The code has been written in Fortran90, C++ for the fft and MPI for the communication routines.
- This DNS has been a very expensive simulation, but not more that one experiment of the same magnitude, and we can compute almost any imaginable quantity.
- We have obtained 25TB of data that we are analyzing: Pressure, Energy balances…
- We have confirmed some trends but we also have found new questions.

Future work?

When a channel 4000 ?

- Grid size

  (12288,901,9216)

- Estimated time per step:

  140s on 4096 processors, 280 on 2048

- Number of steps needed

  250.000

- Total time

  20 million CPU-Hours, between 800 and 1400 days

- Do you have a new MareNostrum?

¡Gracias!

Fluid Dynamics Lab

Escuela de Aeronáutica, UPM

http://torroja.dmt.upm.es