Computación de Alto Rendimiento en Mecánica Computacional. Aplicaciones y desarrollo de herramientas de Software Libre.

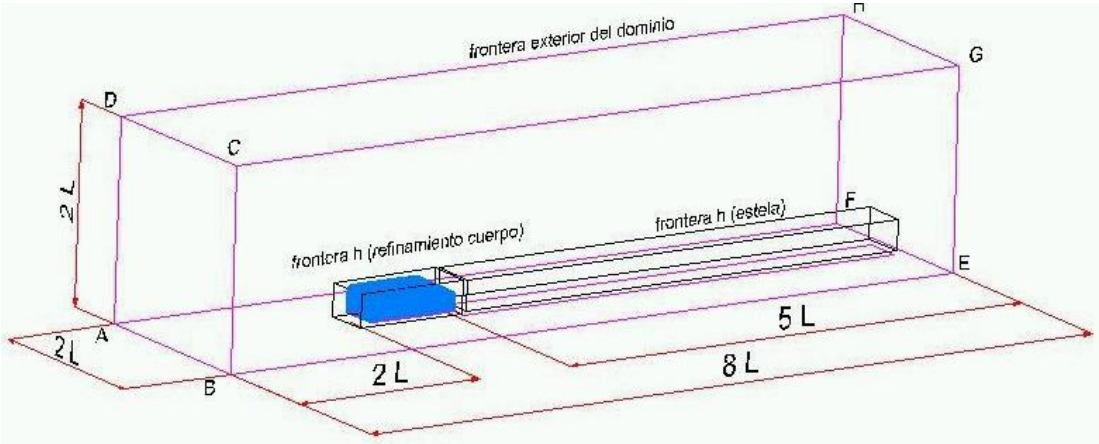
by Mario Storti, Norberto Nigro, Lisandro Dalcín and Rodrigo Paz

Centro Internacional de Métodos Numéricos
en Ingeniería - CIMEC
INTEC, (CONICET-UNL), Santa Fe, Argentina
<{mstorti,nnigro,dalcinl, rodrigop}
@intec.unl.edu.ar>
http://www.cimec.org.ar/mstorti

(document-version "cluster-conf2-0.0.3") (document-date "2005/11/22 03:20:41 UTC")

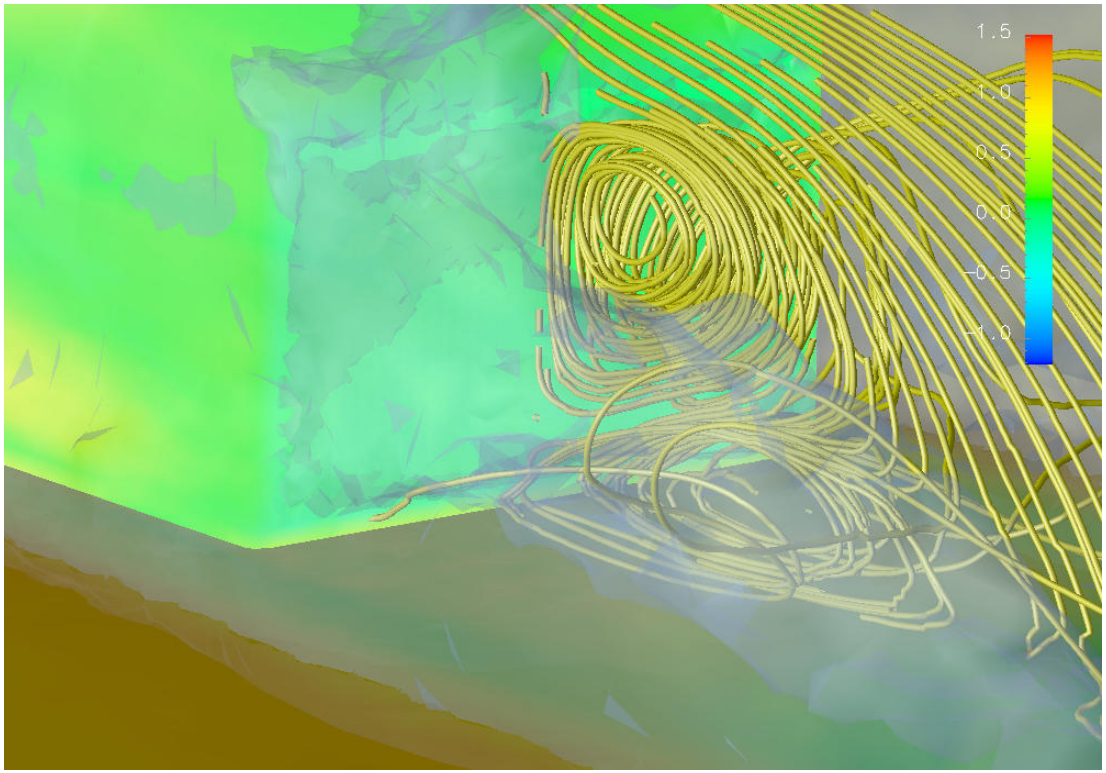
Aerodinámica exterior

- **Resolver las ecuaciones de la dinámica de gases alrededor de un cuerpo. *Objetivo:* Determinar fuerzas aerodinámicas como resistencia al avance y sustentación. En vehículos deportivos para obtener mayores velocidades y fuerza geotrópica. En vehículos de calle para disminuir el consumo de combustible.**
- ***Ecuaciones a resolver:* Ecuaciones de Navier-Stokes incompresible. *Campos involucrados:* velocidad y presión.**



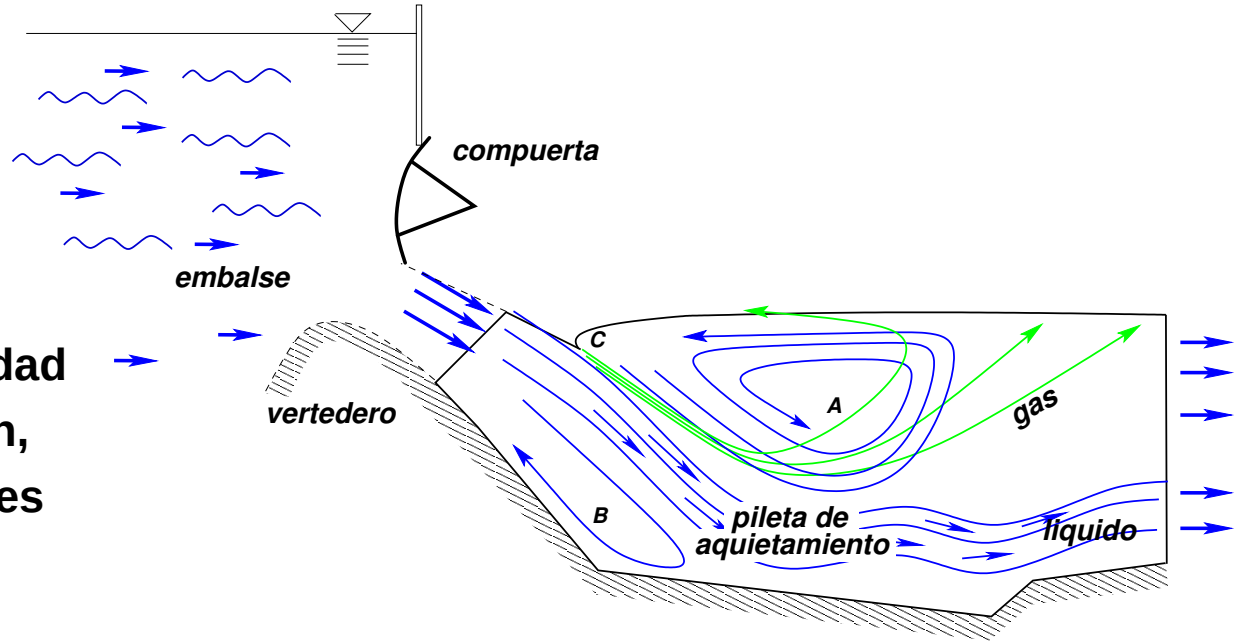
Aerodinámica exterior (cont.)

- **La simulación numérica permite obtener todas las propiedades del flujo: campos de velocidad y presión, esfuerzos de corte, zonas de desprendimiento del flujo, ...**
- **Líneas de corriente en la parte trasera del vehículo.**

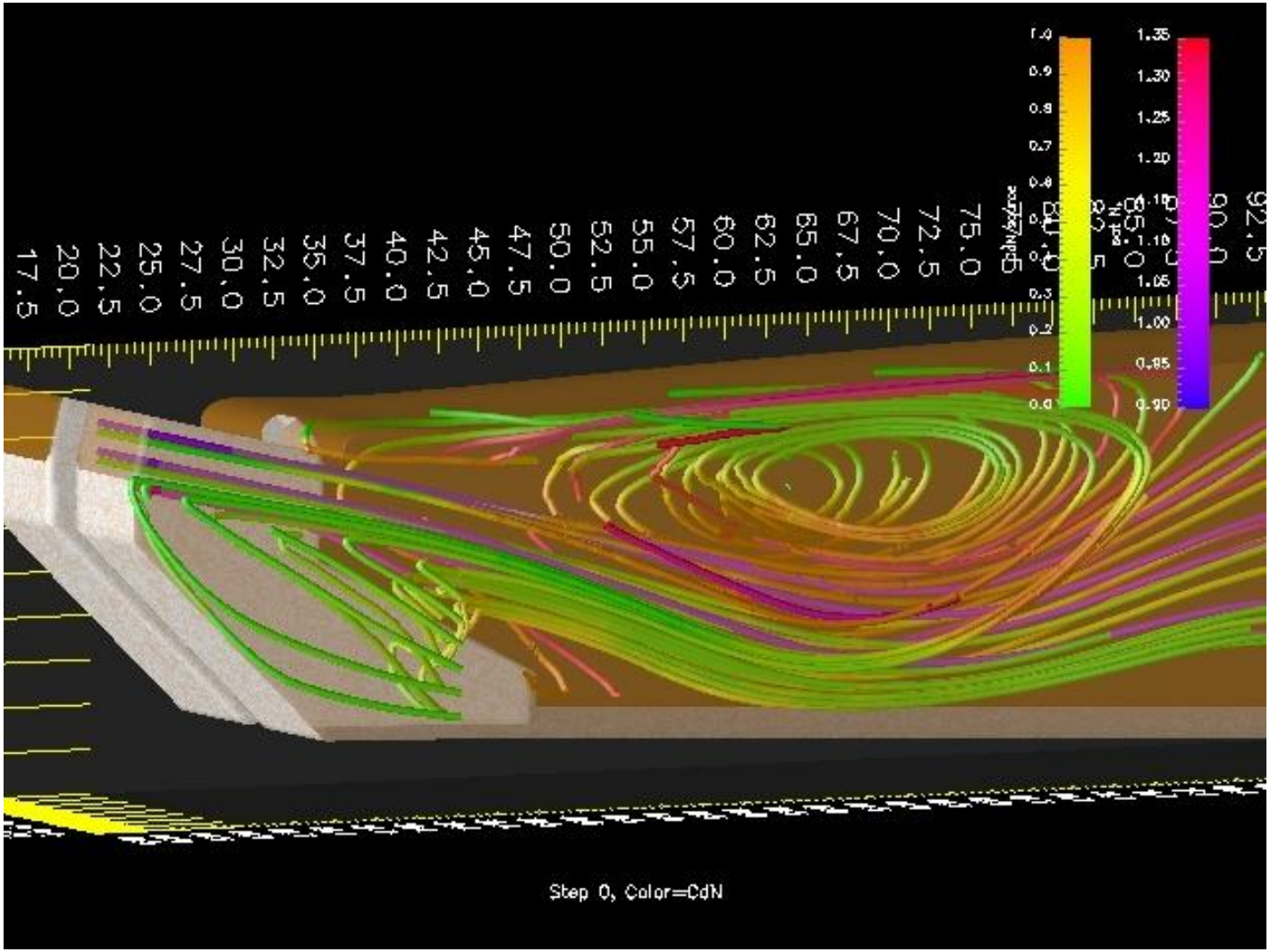


Estudio de impacto ambiental en Yacyreta.

- ***Objetivo:*** **Calcular concentración de gases disueltos en la pileta de aquietamiento (aguas abajo del vertedero). (Altas concentraciones de N2 y O2 disuelto puede producir embolia en especies ictícolas.)**
- ***Ecuaciones a resolver:*** **Ecuaciones de flujo bifásico para la mezcla líquido/aire.**
  - ***Campos involucrados:*** **velocidad de líquido, velocidad de gas, presión, concentraciones de N2 y O2 en líquido y gas.**

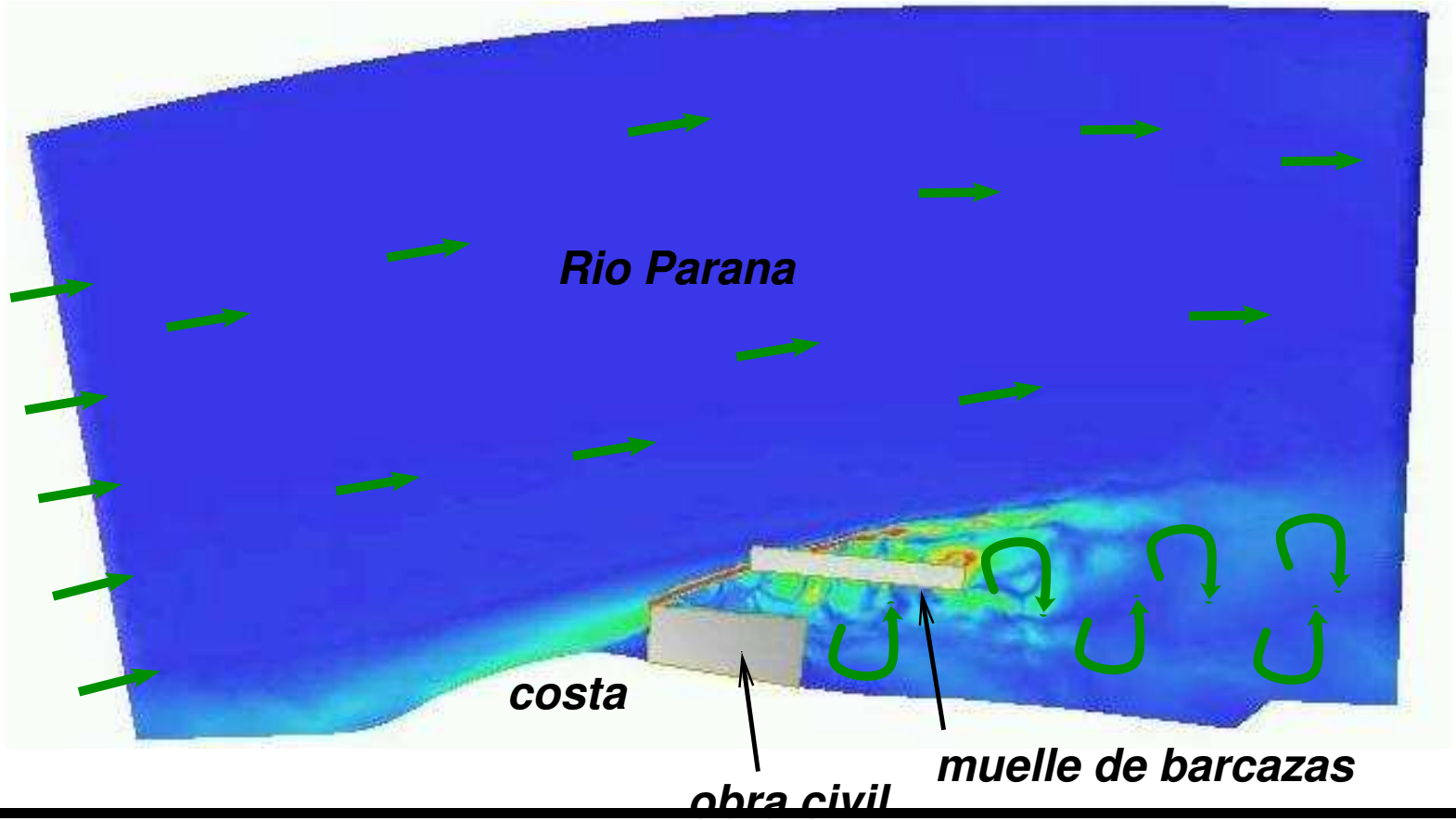


Estudio de impacto ambiental en Yacyreta. (cont.)



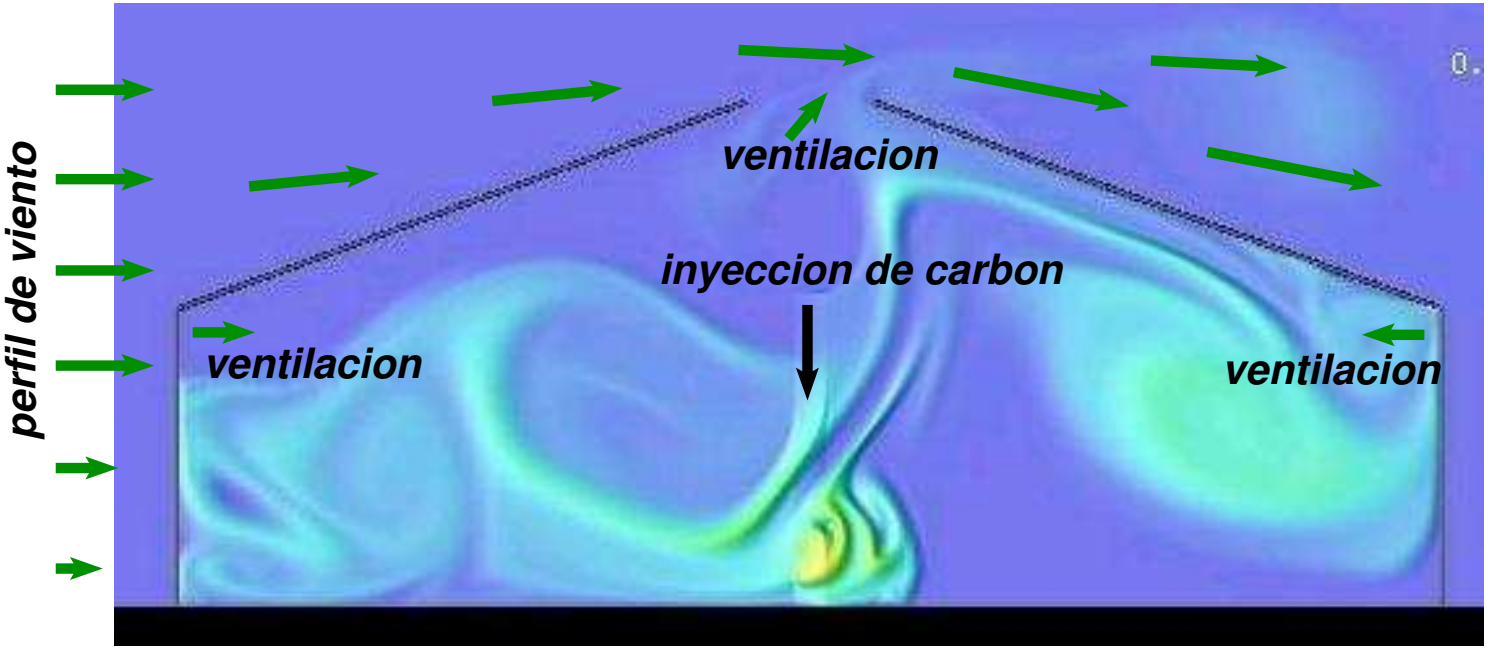
Impacto de obra civil sobre puerto de barcazas.

- ***Objetivo:*** **Calcular intensidad de turbulencia cerca del muelle producida por construcción civil en la costa.**
- ***Ecuaciones a resolver:*** **Ecuaciones de Navier-Stokes 3D.**



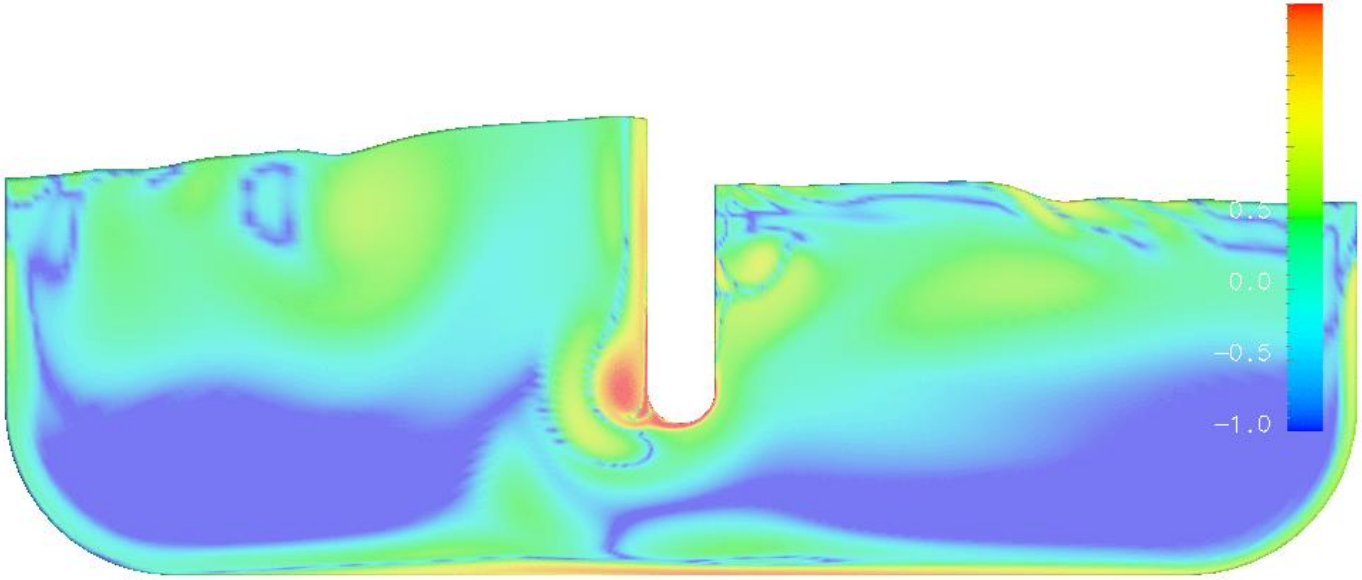
Impacto ambiental de planta procesadora de carbón.

- ***Objetivo:*** **Determinar concentraciones de carbón que escapan a la atmósfera por ventilación natural.**
- ***Ecuaciones a resolver:*** **Ecuaciones de flujo bifásico para la mezcla aire/sólido.**
- ***Campos involucrados:*** **velocidad de líquido, velocidad de partículas sólidas, presión, concentración de partículas sólidas.**



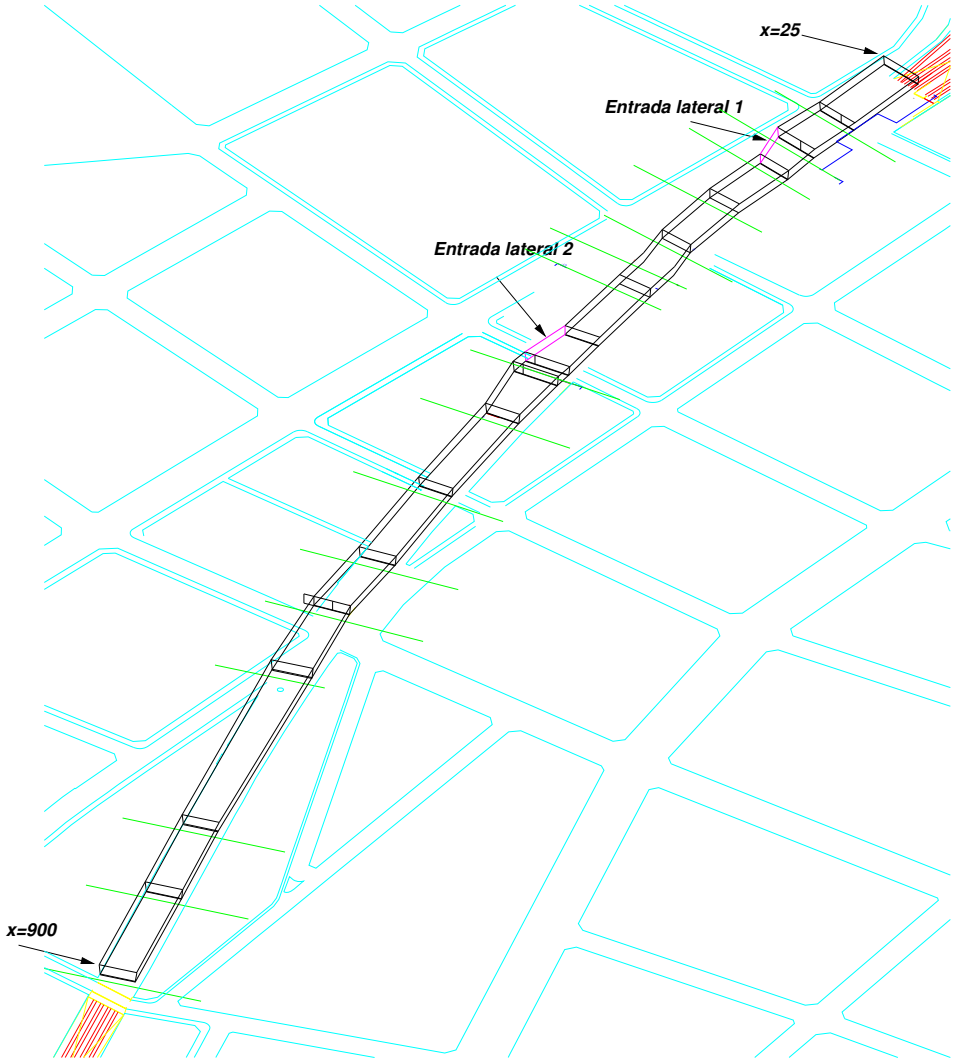
Fuerzas sobre un transporte de líquidos

- ***Objetivo:*** **Determinar fuerzas sobre un contenedor que tranporta líquido ante una desaceleración brusca.**
- ***Ecuaciones a resolver:*** **Ecuaciones de Navier-Stokes con superficie libre.**
- ***Campos involucrados:*** **velocidad de líquido, elevación de la superficie libre.**



Prevención contra incendios

- **La ciudad de BA planea cubrir el ferrocarril Sarmiento en un tramo de 800 m a partir de la estaciónd e Once.**
- **Se necesita predecir el Tiempo Disponible para Escape (TAE *"Time Available for Escape"*) y el Tiempo Necesario para Escape (TNE = *"Time Needed for Escape"*) en el caso de un incendio accidental. Este desarrollo es solicitado por los constructores (Vialmani S.A., HP-IC S.A. y KB Eng. S.A.).**
- **Se simula un fuego con un desarrollo estándar en el vagón (27 MW, 1500 C, 6 % CO, 3×10^7 part.sól./min @ 10 micras, 4×10^3 part.sól./min @ 100 micras).**


x=25
Entrada lateral 1
Entrada lateral 2
x=900

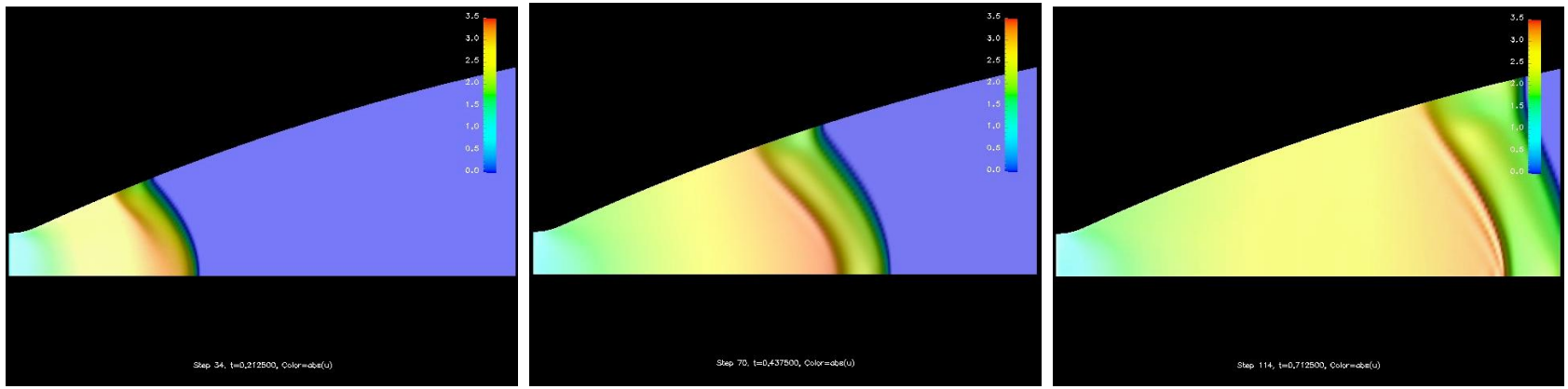
Prevención contra incendios (cont.)

Llenado de la tobera divergente de expansión

Llenado de la tobera divergente de expansión (cont.)

Se modela la ignición de una tobera divergente de escape de un motor de cohete in atmósfera de baaja presión. El gas está inicialmente en reposo (143 Pa, 262 $^\circ\mathrm{K}$). A $t = 0$ el motor se enciende y de la garganta de la tobera sale un gas a $6 \times 10^5\ \mathrm{Pa}$, $4170^\circ\mathrm{K}$. Un *onda de choque* fuerte (intensidad $p_1/p_2 > 1000$) se propaga desde la garganta hasta la salida, llenando la tobera con un flujo a Mach mayor que 1. En el estado estacionario se encuentra un flujo supersonico que empieza desde Mach=1 en la garganta hasta Mach=4. El objetivo de la simulación es predecir las características del flujo estacionario y el tiempo de llenado.

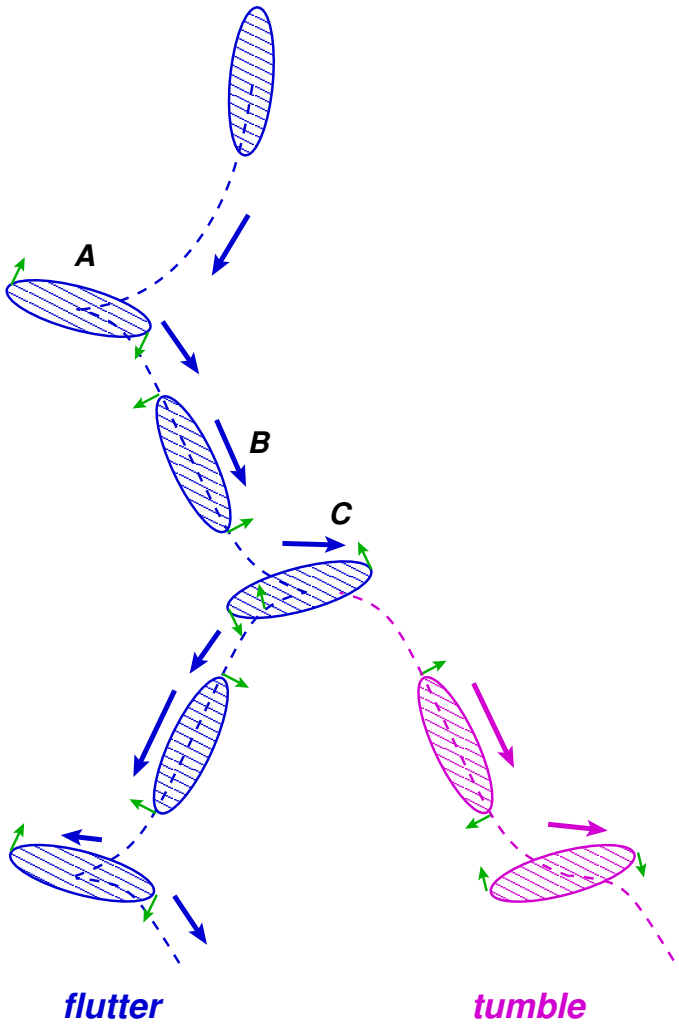
Llenado de la tobera divergente de expansión (cont.)



Este cálculo se ha realizado por encargo de la ESTEC/ESA (European Space Research and Technology Centre-European Space Agency, Noordwijk, Holanda) a través de la empresa Open-Engineering (Liege, Belgium). The predicted mean velocity was 2621 m/s to be compared with the experimental value of 2650+/-50 m/sec.

Object falling at supersonic speed

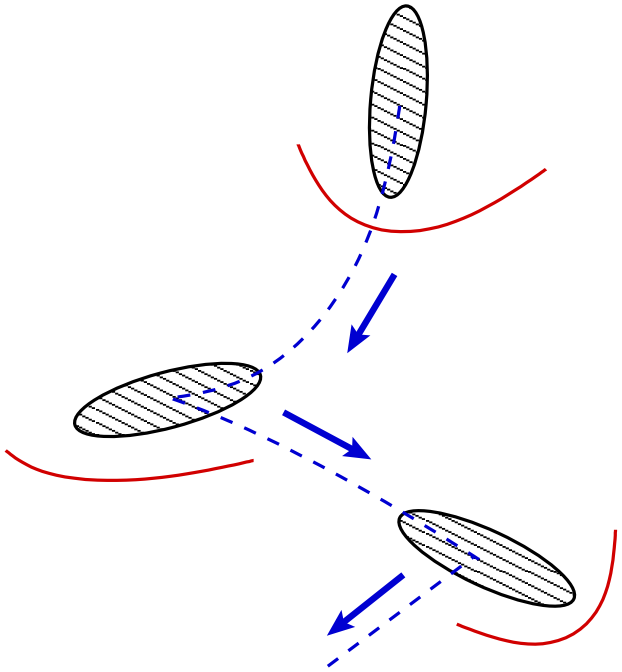
Consider, for simplicity, a two dimensional case of an homogeneous ellipse in free fall. As the body accelerates, the pitching moments tend to increase the angle of attack until it stalls (A), and then the body starts to fall towards its other end and accelerating etc... (*"flutter"*). However, if the body has a large angular moment at (B) then it may happen that it rolls on itself, keeping always the same sense of rotation. This kind of falling mechanism is called *"tumbling"* and is characteristic of less slender and more massive objects.



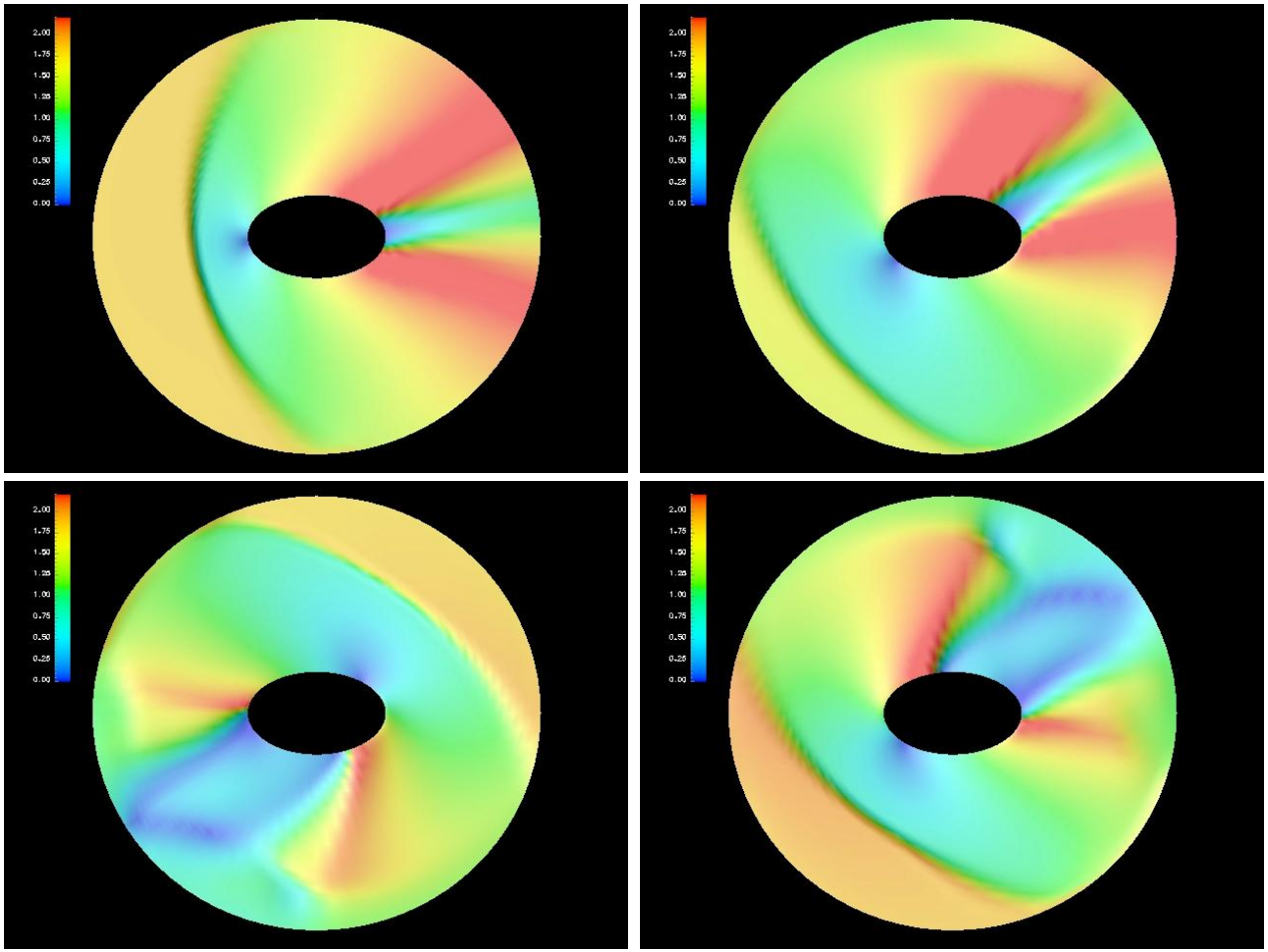
Object falling at supersonic speed (cont.)

Under certain conditions in size and density relation to the surrounding atmosphere it reaches supersonic speeds. In particular as form drag grows like L^2 whereas weight grows like L^3, larger bodies tend to reach larger limit speeds and eventually reach supersonic regime. At supersonic speeds the principal source of drag is the shock wave, we use slip boundary condition at the body in order to simplify the problem.

We also do the computation in a non-inertial system following the body, so that non-inertial terms (Coriolis, centrifugal, etc...) are added. In this frame some portions of the boundary are alternatively in all the conditions (subsonic incoming, subsonic outgoing, supersonic incoming, supersonic outgoing).
Again, the ideal would be to switch **dynamically** ***from one condition to the other during the computation.***



Object falling at supersonic speed (cont.)



Simulación numérica

- **Resolver las ecuaciones que gobiernan el fluido (Ecuaciones de Navier-Stokes/Euler/Flujo Potencial).**
- **Dividir el dominio en pequeños “elementos”.**
- **Asumir variaciones simples de las variables dentro de cada elemento.**
- **Esto lleva a sistemas de ecuaciones lineales con tantas incógnitas como grados nodos por campos incógnita existen.**
- **No confundir simulación numérica con los (hoy muy comunes) efectos especiales usados en películas, etc...!! Estos resultados se verifican con resultados obtenidos en mediciones experimentales.**
- **La simulación numérica es uno de las disciplinas relacionadas con la computación más antiguas. De hecho, uno de los principales usos de las primeras computadoras fue la simulación numérica en la industria de la aviación, nuclear, etc...**

Ventajas de la simulación numérica

- **El costo de las simulaciones numéricas es menor que las simulaciones experimentales y tiende a disminuir constantemente.**
- **Permite conocer el estado del fluido en todo punto del dominio.**
- **No hay problemas de escala o peligrosidad.**
- **Los mismos recursos (hardware y software) pueden ser usados para una variedad de problemas de ingeniería.**

Desventajas

- **El modelo matemático a resolver no es perfectamente conocido para ciertos problemas.**
- **El grado de refinamiento a veces no es suficiente para obtener una representación adecuada.**

Técnicas de computación de alta-performance (HPC)

- **En la búsqueda de modelizaciones cada vez más precisas, se van desarrollando técnicas para incrementar la capacidad de cálculo.**
- **Una de las técnicas más comunes y prometedoras es la de dividir el problema en subproblemas más pequeños y resolver cada uno de los problemas en un procesador por separado. A esto se le da el nombre de *"procesamiento distribuido"*.**
- **En el mejor de los casos, si cada uno de los subproblemas es independiente del otro (están *"desacoplados"*) entonces el tiempo de cálculo en n procesadores es $T_n = T_1/n$.**
- **Debido a que, en general, los problemas no están desacoplados es necesario pasar cierta información de un procesador a otro. Es decir que $T_n = T_1/n + T_{\mathrm{comm}}$.**

Técnicas de computación de alta-performance (HPC) (cont.)

- **El factor de ganancia al paralelizar se llama *"factor de aceleración"* (*"speedup"*) y se define como $S_n = T_1/T_n$. En el caso ideal en que el problema es completamente desacoplado entonces $S_n = n$. La *"eficiencia"* de la paralelización $\eta = S_n/n < 1$.**
- **Una lista de las más veloces computadoras (la lista de los *"Top 500"*) es mantenida por Jack Dongarra en `http://www.netlib.org`. Todas estas máquinas usan procesamiento distribuido, contando con hasta miles de procesadores.**

Clusters Beowulf

- **Con el abaratamiento de las PC's y el advenimiento de software libre surgió la posibilidad de crear clusters de PC's completamente dedicados a cálculo. Estos son los llamados clusters Beowulf.**
- **De *"How to build a Beowulf"* (Sterling, T.L. et.al., MIT Press, 1999) un *"cluster Beowulf"* es *" Un cluster the 'mass-market commodity off-the-shelf' (M2COTS) PC's interconectadas por tecnología LAN de bajo costo corriendo un OS open source de tipo Unix y ejecutando aplicaciones en paralelo con una librería de paso de mensajes estándar en la industria."* El *"Proyecto Beowulf"* fue desarrollado originalmente en el *Goddard Space Flight Center* (GSFC). También son populares los clusters con procesadores DEC/Alpha.**

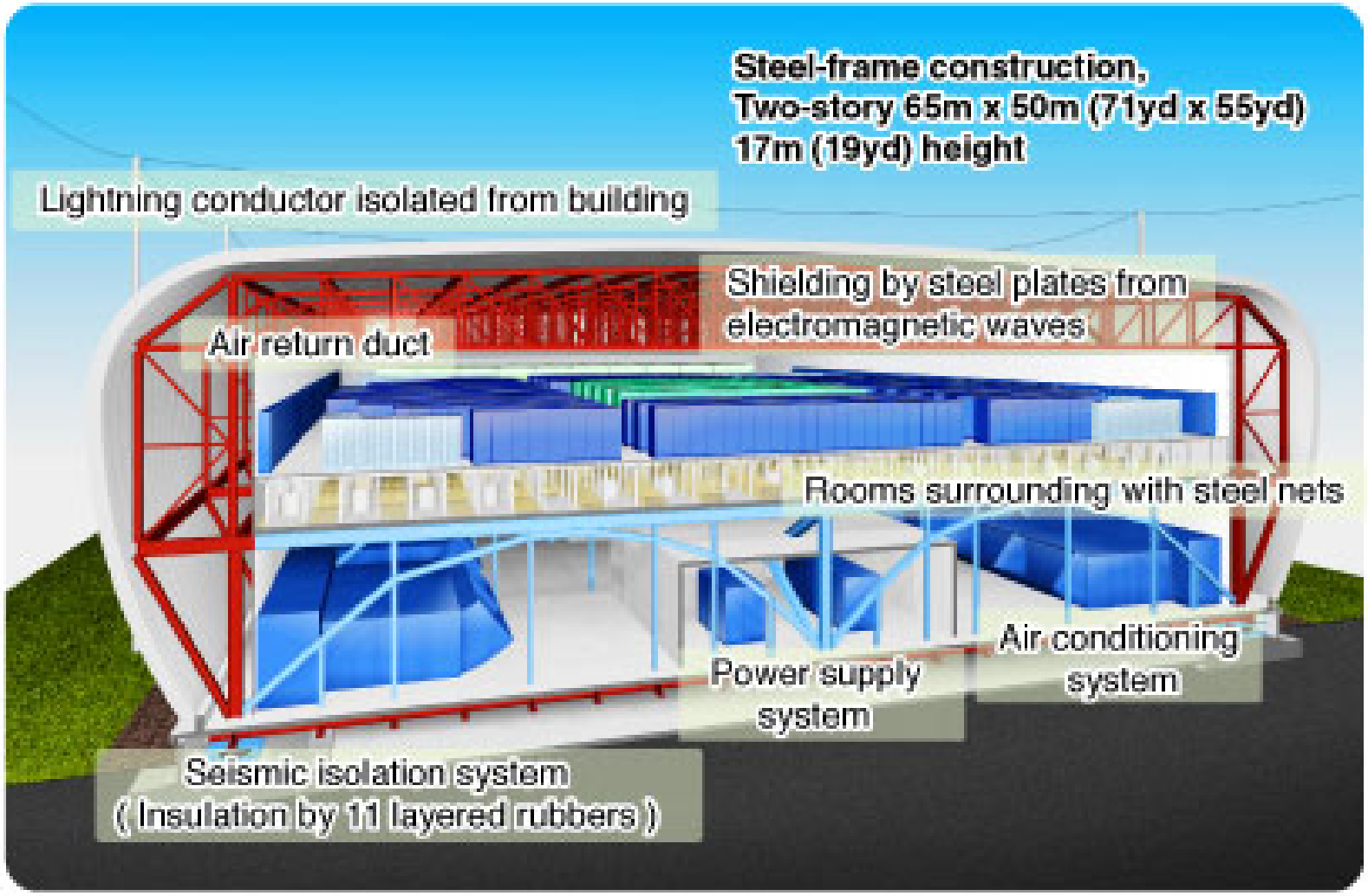
Top 10 list

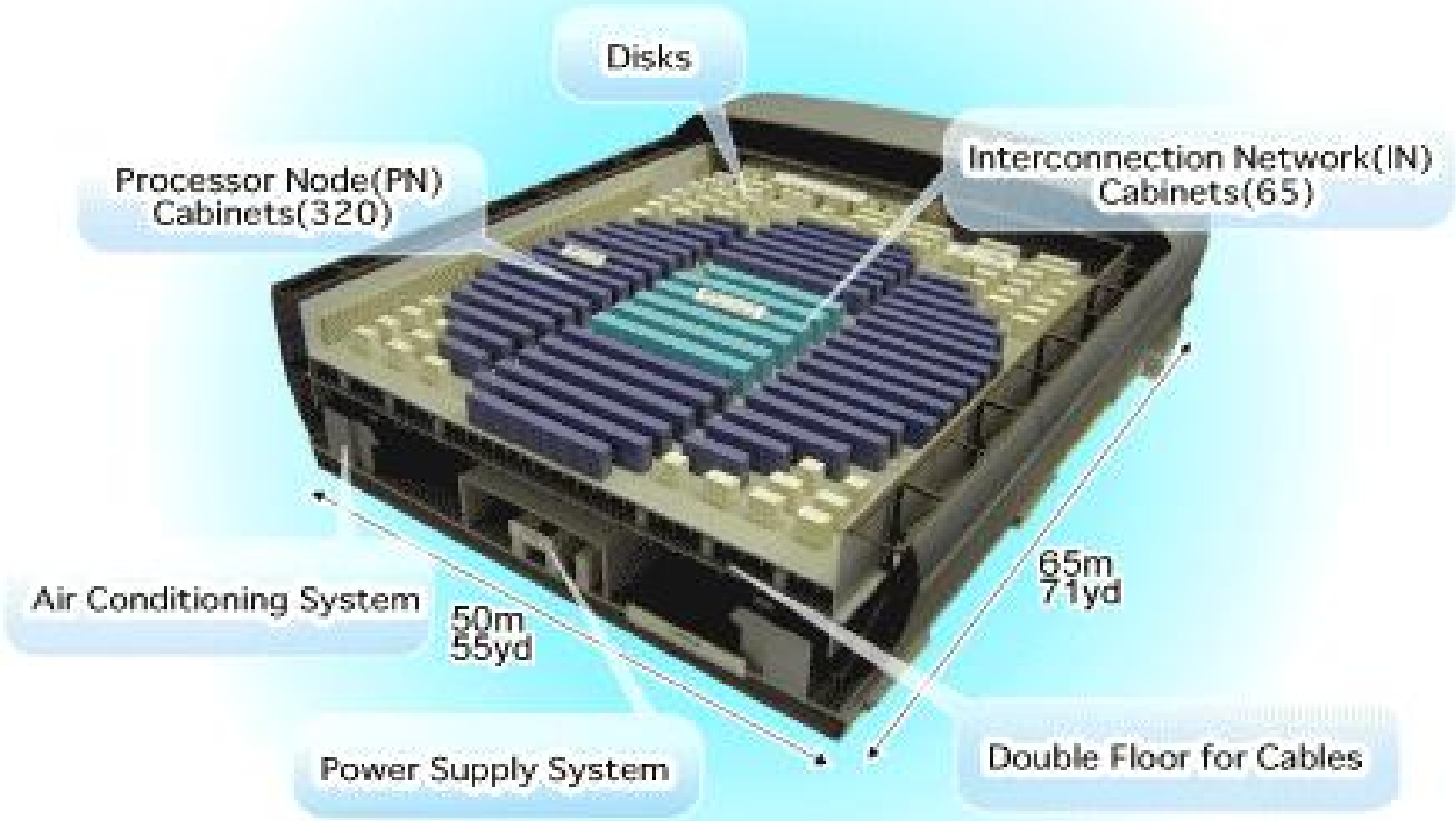
(sacado de www.top500.org)

- **The Earth Simulator, built by NEC, remains the unchallenged #1 at 40 Tflops. (4e12 flop/sc)**
- **ASCI Q at Los Alamos is still #2 at 13.88 TFlop/s.**
- **The third system ever to exceed the 10 TFflop/s mark is Virgina Tech's X measured at 10.28 TFlop/s. This cluster is built with the Apple G5 as building blocks and is often referred to as the 'SuperMac' in media reports. It uses a Mellanox network based on the new Infinband technology as interconnect.**
- **The fourth system is also a cluster. The Tungsten cluster at NCSA is a Dell PowerEdge-based system using a Myrinet interconnect. It just missed the 10 TFlop/s mark with a measured 9.82 TFlop/s.**
- **The list of clusters in the TOP10 continues with the upgraded Itanium2-based Hewlett-Packard system, located at DOE's Pacific Northwest National Laboratory, which uses a Quadrics interconnect.**

Earh simulator (1st pos.)

- 5,120 (640 8-way nodes) 500 MHz NEC CPUs
- 8 GFLOPS per CPU (41 TFLOPS total)
- 2 GB (4 512 MB FPLRAM modules) per CPU (10 TB total)
- shared memory inside the node
- 640 640 crossbar switch between the nodes
- 16 GB/s inter-node bandwidth
- 20 kVA power consumption per node

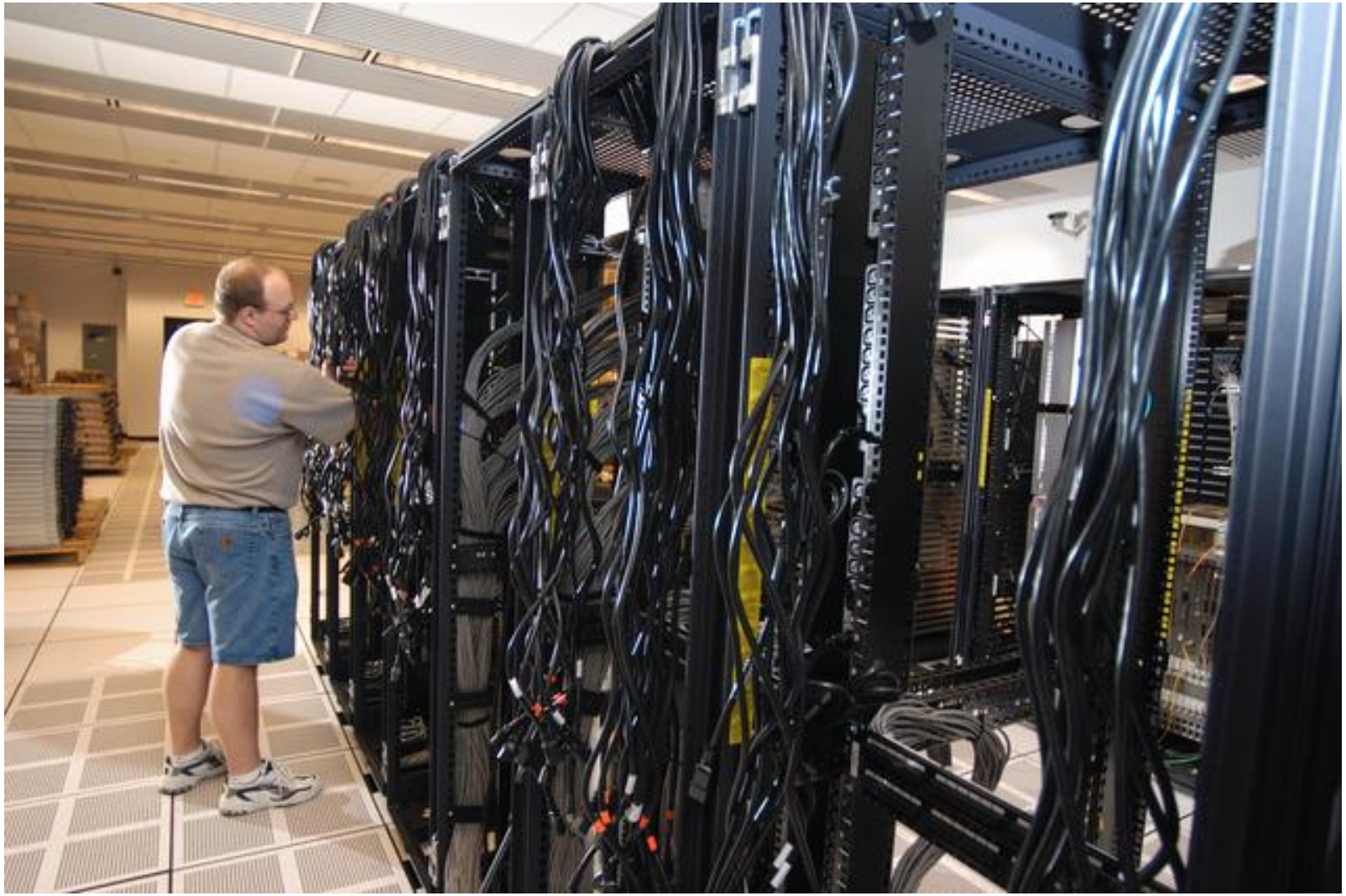

Steel-frame construction,
Two-story 65m x 50m (71yd x 55yd)
17m (19yd) height
Lightning conductor isolated from building
Shielding by steel plates from electromagnetic waves
Air return duct
Rooms surrounding with steel nets
Air conditioning system
Power supply system
Seismic isolation system
(Insulation by 11 layered rubbers)


Disks
Processor Node(PN)
Cabinets(320)
Interconnection Network(IN)
Cabinets(65)
Air Conditioning System
50m
55yd
65m
71yd
Power Supply System
Double Floor for Cables

Tungsten (4th pos.)

Tungsten, NCSA's newest cluster, will employ more than 1,450 dual-processor Dell PowerEdge 1750 servers running Red Hat Linux, a Myrinet 2000 high-speed interconnect fabric, an I/O subcluster with more than 120 terabytes of DataDirect storage, and a dedicated 64-node applications development environment. Tungsten is expected to have a peak performance of 17.7 teraflops (17.7 trillion calculations per second).


DELL



CLIC. Chemniz University Cluster

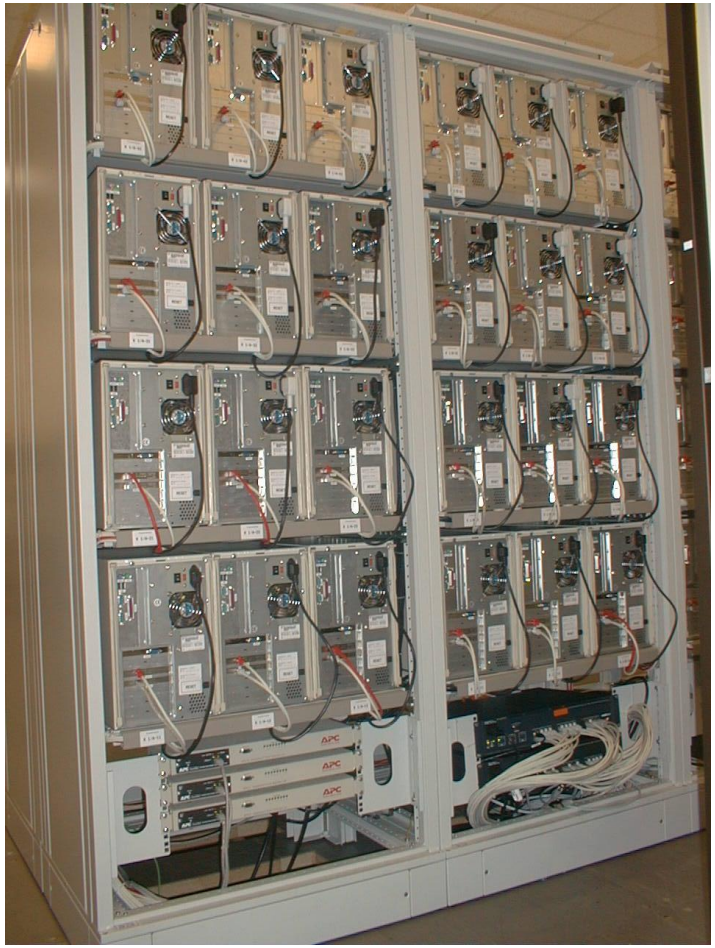
Hardware: Compute Nodes, 528 units, each equipped with

- **Intel PentiumIII, 800 MHz**
- **ASUS-Mainboard CUBX**
- **512 MB SDRAM PC100**
- **harddisk Seagate Barracuda II 20,4 GB**
- **floppy**
- **gfxcard ATI 3D Carger 4MB**
- **2 x network adapter Level One FNC 0108TX**

CLIC. Chemniz University Cluster (cont.)

Server Nodes, 2 units, each equiped with

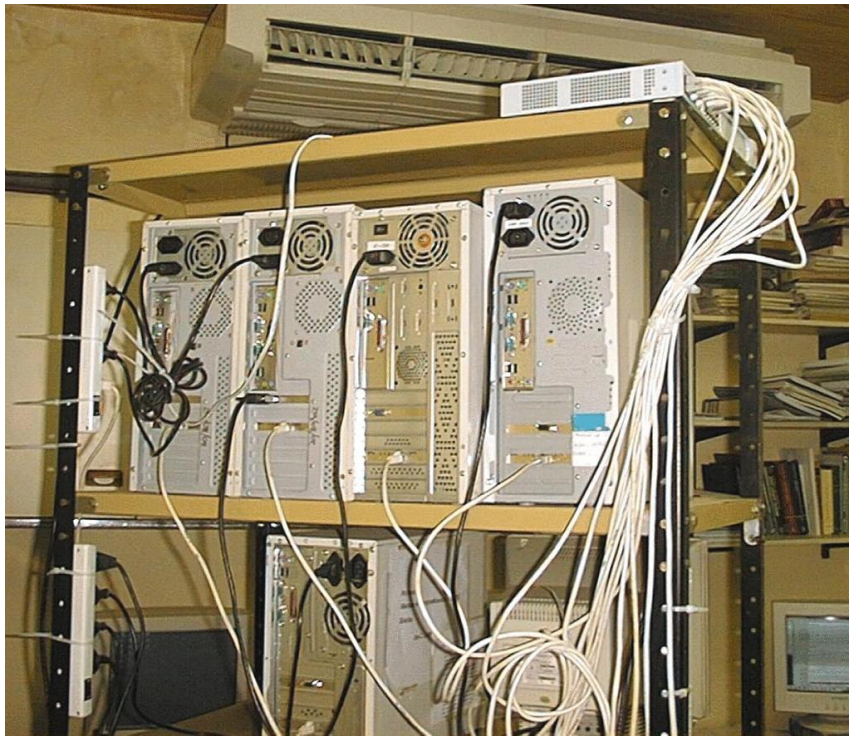
- **Intel PentiumIII, 800 MHz**
- **ASUS-Mainboard CUBX**
- **1 GB SDRAM PC100**
- **harddisk Seagate Barracuda II 20,4 GB**
- **floppy**
- **gfxcard ATI 3D Carger 4MB**
- **2 x Gigabit Ethernet Server Adapter SK 9843-SK-Net GE SX**
- **CD-ROM**



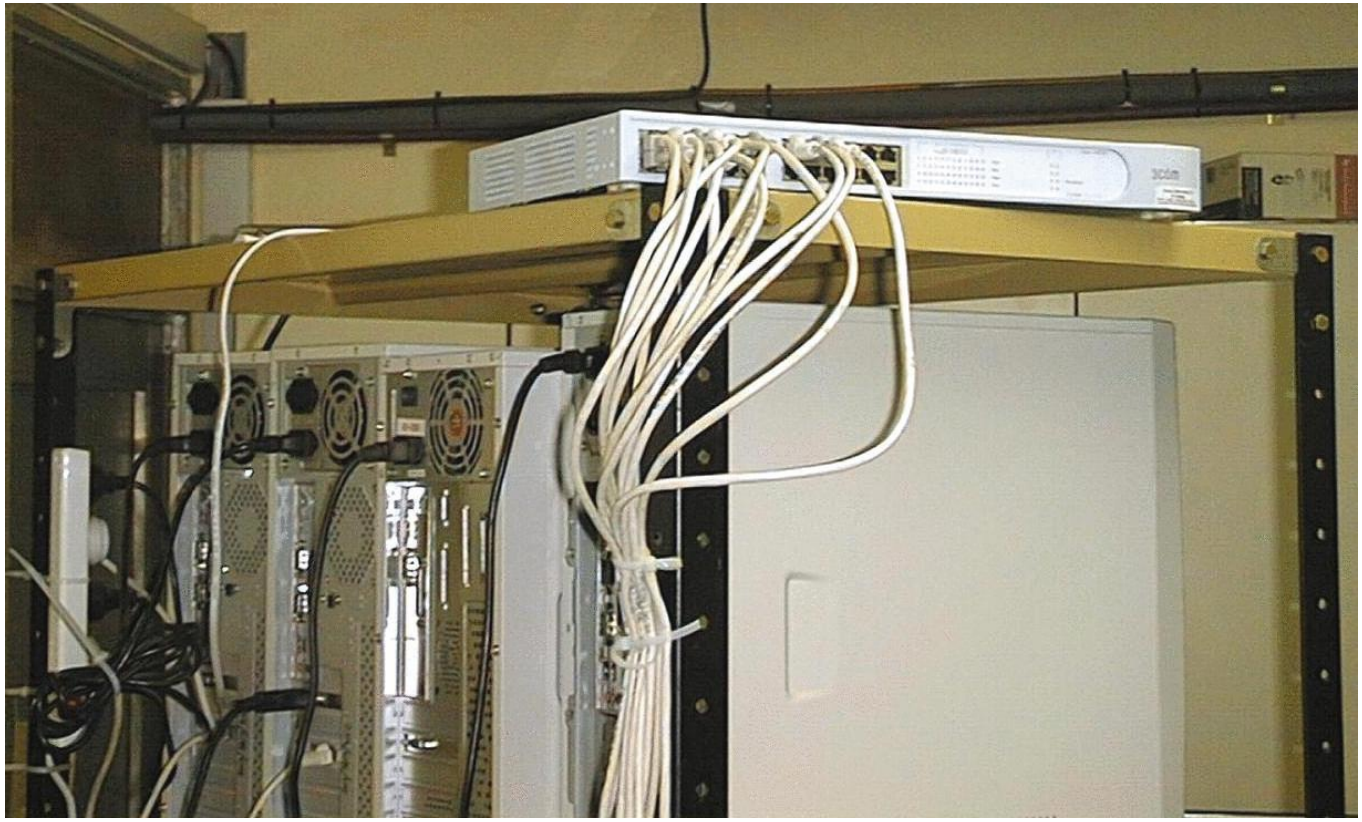
El cluster “Geronimo” en el CIMEC

- **El CIMEC (Centro Internacional de Métodos Numéricos en Ingeniería, ubicado en Santa Fe, dependiente del CONICET) desarrolla tareas de investigación en métodos numéricos desde 1982.**
- **Desde el año 1997 se vienen desarrollando experiencias en procesamiento distribuido. Originalmente en 4 procesadores DEC/Alpha 500/333 Mhz.**
- **Desde 1999 este esfuerzo se ha orientado a los cluster de PC corriendo bajo GNU/Linux. Actualmente, el cluster cuenta con**
  - **Frontend: P IV 1.7 GHz con 768 MB RAM RIMM. 2x120GB HD, 2 3COM NIC cards.**
  - **16 compute nodes P IV 3.0 GHz con 2GB RAM DDR 400 MHz**
  - **All compute nodes have 3COM NIC Cards**
  - **Switch Encore ENH924-AUT+ Fast Ethernet 100MB/sec switch**

El cluster “Geronimo” en el CIMEC (cont.)



El cluster “Geronimo” en el CIMEC (cont.)



Proyecto en ejecución

- **En cooperación con otros cuatro grupos de Santa Fe se esta ejecutando un proyecto PME de la Agencia Nacional de Promoción Científica y Tecnológica (ANPCyT, www.anpcyt.secyt.gov.ar) por us$ 100,000**
  - **Grupo de Física del INTEC**
  - **IMAL (Instituto de Matemática Aplicada del Litoral)**
  - **GEHA (Grupo de Estudios Hidrológicos y Ambientales, FICH-UNL)**
  - **Lab de estudios Hidro-Meteorológicos (FICH-UNL)**

Proyecto en ejecución (cont.)

El cluster Aquiles contará con 86 nodos con las siguientes características

- **Nodos de cálculo con procesador P4 3.0GHz-HT Prescott. FSB 800Mhz.**
- **Placa madre D915PGN.**
- **Placa de red 3C-2000-T Gigabit Ethernet.**
- **2GB memoria Kingston 400 Mhz**
- **2 Switches Gigabit Ethernet 3COM 3870, stackeables.**
- **Fedora Linux 3. Ware-wulf cluster software. RAID.**
- **MPICH, PETSc, PETSc-FEM, OpenMosix, PBS, ...**

Arquitectura

- La configuración del cluster es “disk-less” es decir que los nodos no tienen disco duro.
- Los nodos cargan el kernel via protocolo PXE.
- El uso de la configuración disk-less permite una administración mucho más simple e implica un considerable ahorro ya que no requiere disco duro.
- Nuevos nodos pueden ser instalados en menos de 5 minutos/nodo.

Software utilizado

- **OS: GNU/Linux RedHat 7.1.**
- **Desarrollamos un código de elementos finitos escrito en C++ llamado PETSc-FEM. Actualmente tiene +81,000 líneas de código.**
- **Como librería de paso de mensajes usamos MPI (*"Message Passing Interface"*, `http://www.mcs.anl.gov/mpi`) en su implementación MPICH (versión 1.2.2, `http://www.mcs.anl.gov/mpich`) desarrollados en el *Argonne National Laboratory* (ANL).**

Software utilizado (cont.)

- **En general MPI no es llamado directamente sino a través de PETSc (versión 2.1.6) *Parallel Extensible Toolkit for Scientific Computations* que es un paquete orientado a métodos numéricos en procesamiento distribuido, y permite operaciones abstractas como definir vectores y matrices distribuidos y resolver los sistemas lineales asociados.**
- **Para el álgebra lineal se utiliza los paquetes estándar Lapack y Blas distribuidos por Netlib (`http://www.netlib.org/lapack/`).**
- **Sistema de manejo de colas PBS (Portable Batch System.)**

Características de PETSc-FEM

- **GPL, accesible en `http://venus.ceride.gov.ar/petscfem`, versión actual es `petscfem-beta-3.34`**
- **Programa de elementos finitos de uso general y multi-física.**
- **Procesamiento en paralelo via uso de PETSc/MPI/Metis.**
- **Escrito en C++ con una concepción OOP, con especial énfasis en la eficiencia.**
- **Muchos tipos de elementos pueden ser usados en la misma aplicación, agrupando los elementos del mismo tipo en "elemsets".**
- **Propiedades de elementos pasadas a las rutinas de elementos vía diccionarios.**

Características de PETSc-FEM (cont.)

Características de PETSc-FEM

- **Condiciones de contorno Dirichlet/Newman/mixtas/periódicas o restricciones generales.**
- **Cálculo de jacobianos numéricos.**
- **Actualmente implementados módulos de Navier-Stokes, shallow-water, Euler, sistemas advectivos generalizados y ec. de Laplace.**
- **Perfiles de sistemas de ecuaciones calculados automáticamente.**
- **Librerías de matrices rápidas usando “caches” para las direcciones de memoria.**
- **Balance de carga**

Scripting via extension languages

- **Currently, user data file has a tree-like data structure with nodes meaning of the form (key,value) assignments.**
- **As the syntax of the user data files grew in complexity and more preprocessing is needed (calling functions, system calls...) the syntax *evolves eventually in a language*.**

```
global_options

alpha 1. # Integration rule param.

#if !$restart # Conditional processing...
nstep 100000
#else
nstep 1
#endif

tol_newton 1e-10 # Iterative parameters
atol 0
rtol 1e-7
viscosity <:=$U*$L/$Re:> # handle math exps
...
```

Scripting via extension languages (cont.)

Adding features to the user data file syntax and making it a language has many disadvantages.

- **Developer has to define a correct language (syntax, ...) and implement it.**
- **Users have to learn a new language/syntax.**

This happened with a series of Open Source projects (Apache, PHP, Perl, ...). Developers started with a small language definition and ended with a full featured language. A better solution is to use an already existing high level scripting language and extend it with the functions of the new library. In this way the user needs not to deal with coding in a low level language like C/C++ directly. Candidates for extension language are Perl, Python, Lisp/Scheme, Octave...

Scripting via extension languages (cont.)

The subject was at the heart of a war over the net (the *"Tcl war"*, see `http://www.vanderburg.org/Tcl/war/`) and the Free Software Foundation proposed to design a language specific to be an extension language for Free Software projects. This is Guile, a complete implementation of the Scheme language, a dialect of Lisp. Advantages of Guile are

- **It implements Scheme a full featured language. It implements dynamic typing, supports functional programming, has first class procedures, hygienic macros, first class continuations. There are many implementations (interpreters) of Scheme, Guile is just one of them.**
- **It is the extension language proposed by the FSF.**
- **Full support for loading C functions in libraries.**
- **Can call C functions from Scheme and vice-versa.**

Scripting via extension languages (cont.)

User data file will be a Guile/Scheme script

```
(use-modules (petsc-fem base))
(use-modules (petsc-fem ns))
(use-modules (petsc-fem elasticity))

(define global-options
  '(alpha . 1.0) ;;; Integration rule param.
  (nstep . ,(if restart 1000 1)) ;;; Cond. proc. . .
  (tol-newton . 1e-10) ;;; Iterative parameters
  (atol . 0)
  (rtol . 1e-7)
  ;;; Handles arbitrarily complex math exps
  (viscosity . ,(/ (* rho U L) Re)))
(define coords (elemset-load "./coords.dat"))
(define elemset (elemset-load "./connect.dat"))
(define ns-problem
  (ns-init coords elemset global-options))
. . .
```

Scripting via extension languages (cont.)

- **Currently best candidates are Scheme and Python.**
- **Written wrappers for MPI and PETSc libraries in Python (Lisandro Dalcín). [Dalcín, L., Paz, R., Storti, M. "MPI for Python", Journal of Parallel and Distributed Computing, 65/9, pp. 1108-1115 (2005)]. Can exchange arbitrary Python objects between processors (via `cPickle` module).**
- **Basic MPI wrappers in Scheme have been written.**
- **Sends/receives special vector objects (PETSc-FEM `dvector<T>` class).**
- **Sends complex Scheme objects with serialization via write/read (or *hash-comma* syntax for extended objects (SRFI-10)).**

```
(use-modules (mpi))
(use-modules (dvector))
(mpi-initialize)

;;; Get rank and size
(define my-rank (mpi-rank))
(define size (mpi-size))

;;; Standard parallel hello...
(format t "Hello world I am ~A of ~A\n" my-rank size)

;;; Define vectors v,w, fill v with random
(define N 1000)
(define v (make-dvdbl N))
(define w (make-dvdbl ))
(dv-rand! v)

;;; Rotates data (sends to myrank+1 and
;;; receives from myrank-1, cyclically)
(cond ((even? my-rank)
      (mpi-send v (modulo (+ my-rank 1) size))
      (mpi-recv w (modulo (- my-rank 1) size)))
     (else
      (mpi-recv w (modulo (- my-rank 1) size)))
      (mpi-send v (modulo (+ my-rank 1) size)))

(mpi-finalize)
```

Functional programming style

Functional programming promotes intermix of code and data (CODE=DATA slogan).

```
;;; Saves state file to disk
;;; each 10 time steps
(define nsave 10)

;;; Save each 10 or if large variations in
;;; state vector are detected
(define nsave
  (lambda (step)
    (if (or (= (modulo step 10) )
            (check-state step)))))
```

Functional programming style (cont.)

```
(define visco 1e-3) ;;; Fixed value for viscosity

(define visco  ;;; Use Sutherland's law
  (sutherland-law 1e-3 300 110 120))

;;; This takes the physical parameters
;;; and returns a function of temperature.
(define (sutherland-law muinf Tinf T1 T2 expo)
  (let ((T1 T1) (T2 T2) (expo expo))
    (lambda (T)
        (if (< T T2)
            (* (muinf (/ T Tinf)))
            (* muinf
               (/ T2 Tinf)
               (expt (/ T T2) expo)
               (/ (+ Tinf T1) (T T1)))))))
```

MPI for Python

- **Another possibility for scripting language is Python.**
- **Python is a powerful programming language, easy to learn, with a large library set and a simple and strongly integrated Object Oriented Programming system.**
- **It is an ideal candidate for writing higher-level parts of large scale scientific applications and driving simulations in parallel architectures.**
- **Can call C functions from Python and vice-versa.**
- **Previous to extend PETSc-FEM with Python we worked on adding some parallel functionality to Python. This evolved in the package *"Mpi4py"*.**
- **Related work is found in the OOMPI, Pypar, pyMPI, Scientific Python, Numeric, Numarray, PyFort, SciPy and SWIG projects.**

MPI for Python (cont.)

- First, a special Python interpreter was created. The Python interpreter that comes with the `python` package would be OK for non-interactive use. In interactive use the interpreter must read a line on the master node and broadcast to the slaves. Then the line is evaluated, as is in he usual REPL (Read-Eval-Print loop).
- Any Python object can be transmitted using the standard `cPickle` module. The object to be transmitted is first serialized. After that, string data is communicated (using MPI CHAR datatype). Finally, received strings are unpacked and the original object is restored. Serialization process introduces some overheads: dynamic memory allocations, heavier messages and extra processing. However, this methodology is easily implemented and quite general. Direct communication, i.e., without serialization, of consecutive numeric arrays is feasible but not currently supported. This issue will be addressed in the near future.

MPI for Python (cont.)

- *Comm* class implemented with methods: *Get_size()*, *Get_rank()*, *Clone()*, *Dup()*, *Split()*.
- Set operations with Group objects like *Union()*, *Intersect()* and *Difference()* are fully supported, as well as the creation of new communicators from groups. Virtual topologies (*Cartcomm* and *Graphcomm* classes, both being a specialization of *Intracomm* class) are fully supported.

MPI for Python (cont.)

- **Methods `Send()`, `Recv()` and `Sendrecv()` of communicator objects provide support for blocking point-to-point communications within `Intracomm` and `Intercomm` instances. Non-blocking communications are not currently supported.**
- **Methods `Bcast()`, `Scatter()`, `Gather()`, `Allgather()` and `Alltoall()` of `Intracomm` instances provide support for collective communications. Global reduction operations `Reduce()`, `Allreduce()` and `Scan()` are supported but naively implemented.**

MPI for Python (cont.)

- **Error handling functionality is almost completely supported. Errors originated in native MPI calls will throw an instance of the exception class `Exception`, which derives from standard exception `RuntimeError`.**

MPI for Python (cont.)

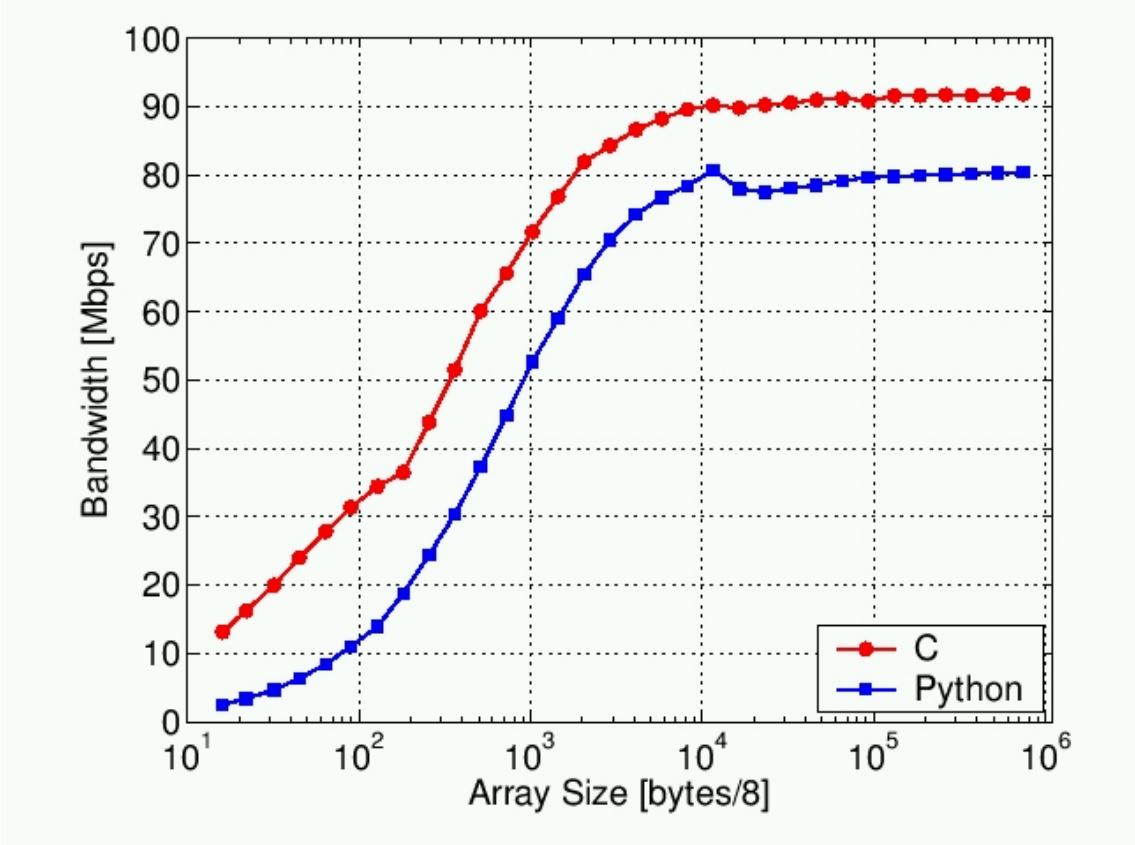
- **Efficiency: Some efficiency tests were run on the Beowulf class cluster Geronimo at CIMEC. Hardware consisted of ten computing nodes with Intel P4 2.4Ghz processors, 512KB cache size, 1024MB RAM DDR 333MHz and 3COM 3c509 (Vortex) NIC cards interconnected with an Encore ENH924-AUT+ 100Mbps Fast Ethernet switch. MPI for Python was compiled with MPICH 1.2.6 and Python 2.3, Numarray 1.1 was also used.**
- **The first test was a bi-directional blocking send and receive between pairs of processors. Messages were numeric arrays (NumArray ob jects) of double precision (64 bits) floating-point values. A basic implementation of this test using MPI for Python (translation to C or C++ is straightforward) is shown below.**

MPI for Python (cont.)

```
from mpi4py import mpi
import numarray as na
sbuff = na.array(shape=2**20,
                 type= na.Float64)
wt = mpi.Wtime()
if mpi.even:
   mpi.WORLD.Send(buffer,mpi.rank+1)
   rbuff = mpi.WORLD.Recv(mpi.rank+1)
else:
   rbuff = mpi.WORLD.Recv(mpi.rank-1)
   mpi.WORLD.Send(buffer,mpi.rank-1)
wt = mpi.Wtime() - wt
tp = mpi.WORLD.Gather(wt, root=0)
if mpi.zero: print tp
```

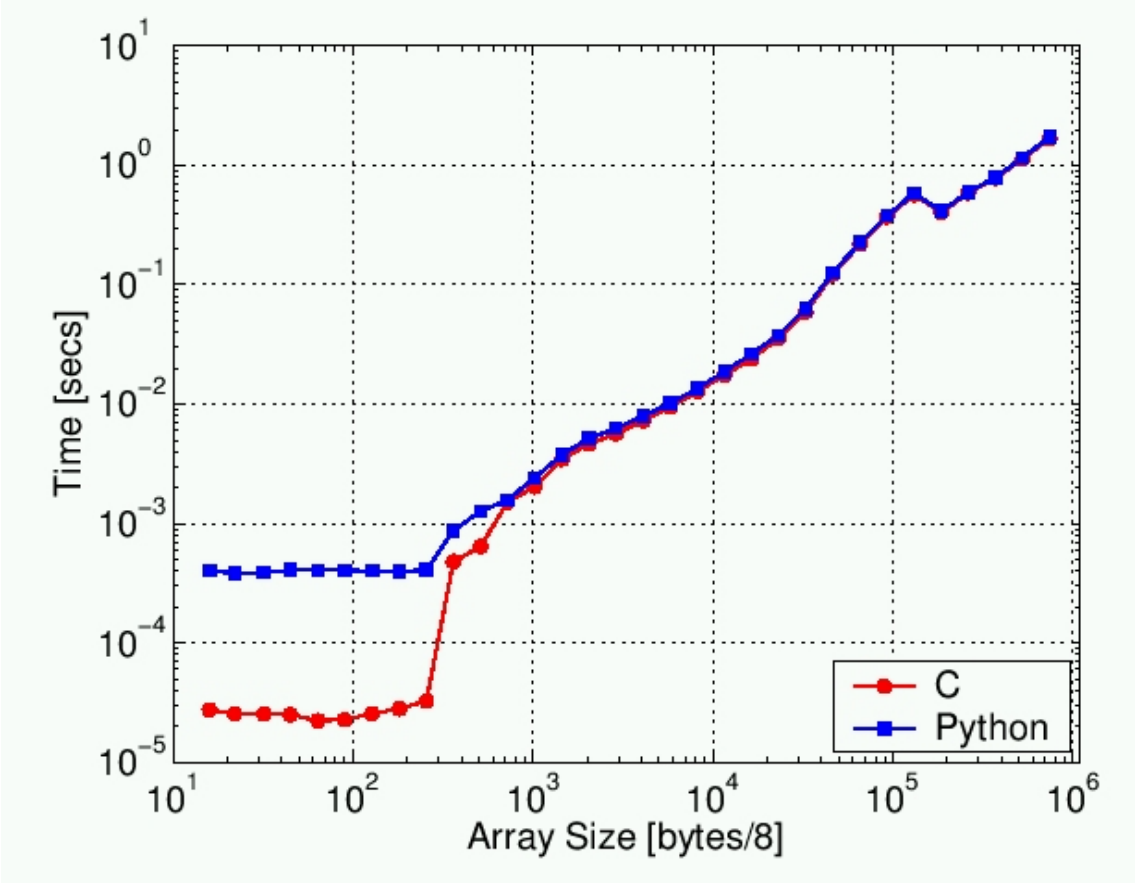
MPI for Python (cont.)

Maximum bandwidth in Python is about 85 % of maximum bandwidth in C. Clearly, the overhead introduced by object serialization degrades overall efficiency.

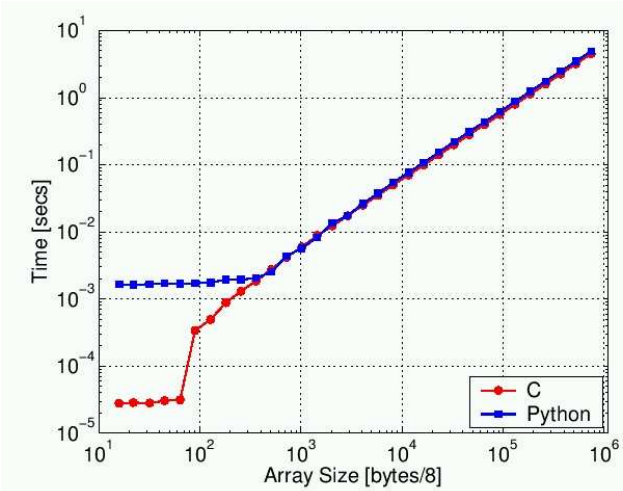


MPI for Python (cont.)

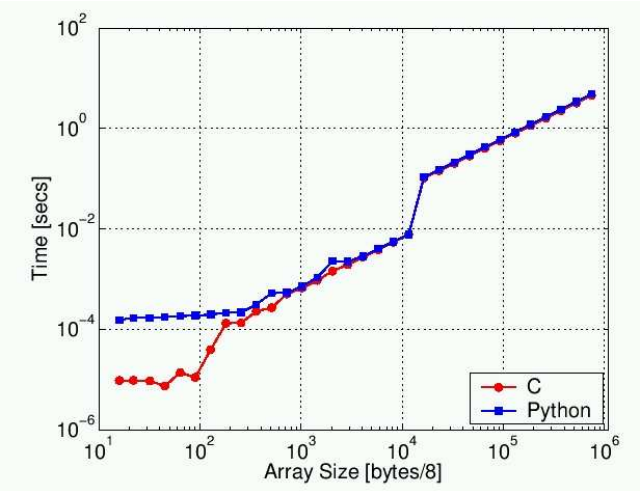
The second test consisted in wall-clock time measurements of some collective operations on ten uniprocessor nodes. Messages were again numeric arrays of double precision floating-point values. For array sizes greater than 103 (8KB), timings in Python are between 5 % (for Bcast) to 20 % (for Alltoall) greater than timings in C. (*Timing in Broadcast*)



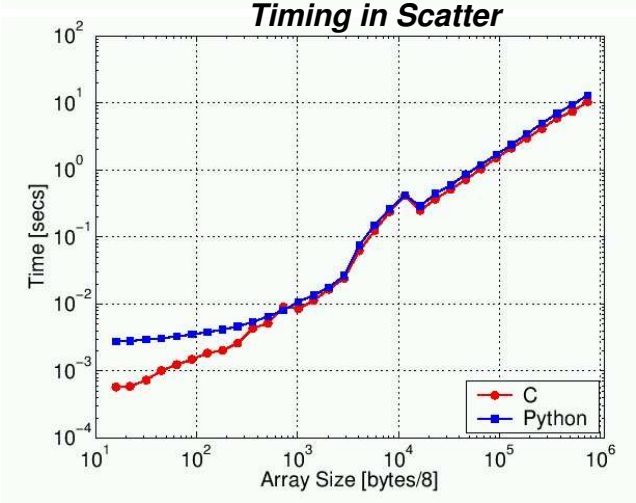
MPI for Python (cont.)



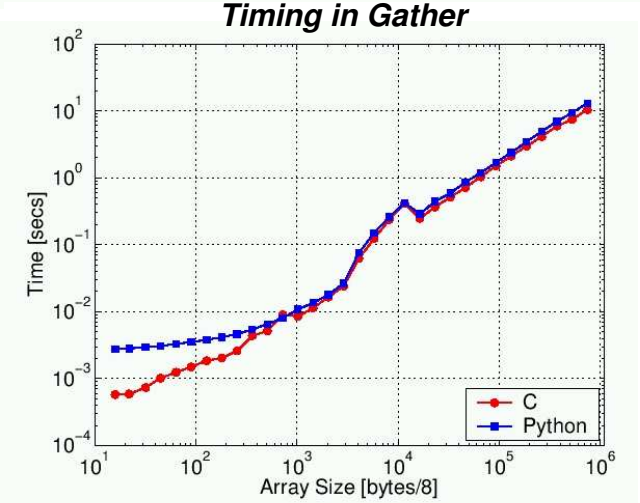
Timing in Scatter



Timing in Gather



Timing in Gather to All



Timing in All to All Gather/Scatter

MPI for Python (cont.)

Conclusions:

- **Like any scripting language, Python is not as efficient as compiled languages. However, it was conceived and carefully developed to be extensible in C (and consequently in C++).**
- **Python can be used as a glue language capable of connecting existing software components in a high-level, interactive, and productive environment.**
- **Running Python on parallel computers is a good starting point for decreasing the large software costs of using HPC systems.**
- **MPI for Python can be used for learning and teaching message-passing parallel programming taking advantage of Python's interactive nature.**

MPI for Python (cont.)

- **Efficiency tests have shown that performance degradation is not prohibitive, even for moderately sized objects. In fact, the overhead introduced by MPI for Python is far smaller than the normal one associated to the use of interpreted versus compiled languages.**
- **Examples of Mpi4py use in the JPDC paper or in the package (`http://www.cimec.org.ar/python`, maintainer `mailto:dalcinl@intec.unl.edu.ar`)**

- ***Future work:* Add some currently unsupported functionalities like non blocking communications and direct communication of numeric arrays.**
- **Develop Python packages providing access to very well known and widely used MPI based parallel libraries like PETSc and ParMETIS (almost done)**
- **Furthermore, the higher-level portions of the parallel multi-physics finite elements code PETSc-FEM [32,33] developed at CIMEC are planned to be implemented in Python in the next major rewrite.**

Proyectos

- **Balance dinámico de carga, a través de un algoritmo de tipo *"self-scheduling"*.**
- **Agregar un lenguaje de scripting (Guile? Perl? Python?)**
- ***"Functional programming"* (con Scheme? u otro?)**
- **Mejorar la OOP y el soporte a multifísica.**
- **Refinamiento adaptivo.**
- **Métodos multigrilla.**

Agradecimientos

- **Los resultados mostrados en este trabajo fueron obtenidos por un grupo de colaboradores en el CIMEC: Norberto Nigro, Jorge Rodrigo Paz, Lisandro Dalcín.**
- **Este trabajo ha recibido financiamiento de las siguientes instituciones Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET, Argentina, subsidios PIP 0198/98, PIP 02552/00, PIP 5271/05), Universidad Nacional del Litoral (UNL, Argentina, subsidios CAI+D 2000/43) y Agencia Nacional de Promoción Científica y Tecnológica (ANPCyT, Argentina, subsidios PICT 6973/99, PID-74/99, PICT Lambda 12-14573/2003, PME 209/2003).**

- **Se ha hecho uso intensivo de *Software Libre* (`http://www.gnu.org`) como GNU/Linux OS, MPI, PETSc, compilador GCC, Octave, Open-DX y muchos otros.**
- **Este material y más están disponibles en `http://www.cimec.org.ar/mstorti`.**