PyACTS: Una interface Python para las herramientas ACTS

Jose Penadés

Grupo de Computación de Altas Prestaciones y Paralelismo

Universidad de Alicante

Universitat d'Alacant
Universidad de Alicante

Vicente Galiano
Héctor Migallón
Universidad Miguel Hernández
Violeta Migallón
Universidad de Alicante
L.A. Drummond
National Berkeley Laboratory

PyACTS: Una interface Python para las herramientas ACTS

1. Motivación
2. Estructura de PyACTS
3. Herramientas auxiliares
4. Las interfaces PyBLACS, PyPBLAS y PyScaLAPACK
5. Aplicaciones
   - PyClimate
   - Modelado de la estructura interna de la Tierra.

1. Motivación

Necesidades:

- Uso de paralelismo para simular y resolver problemas científicos.
- Modelos, algoritmos y fenómenos son cada vez más complejos.
- Pero,... la programación paralela (y el uso de librerías paralelas) no es sencilla para usuarios no expertos.

Solución:

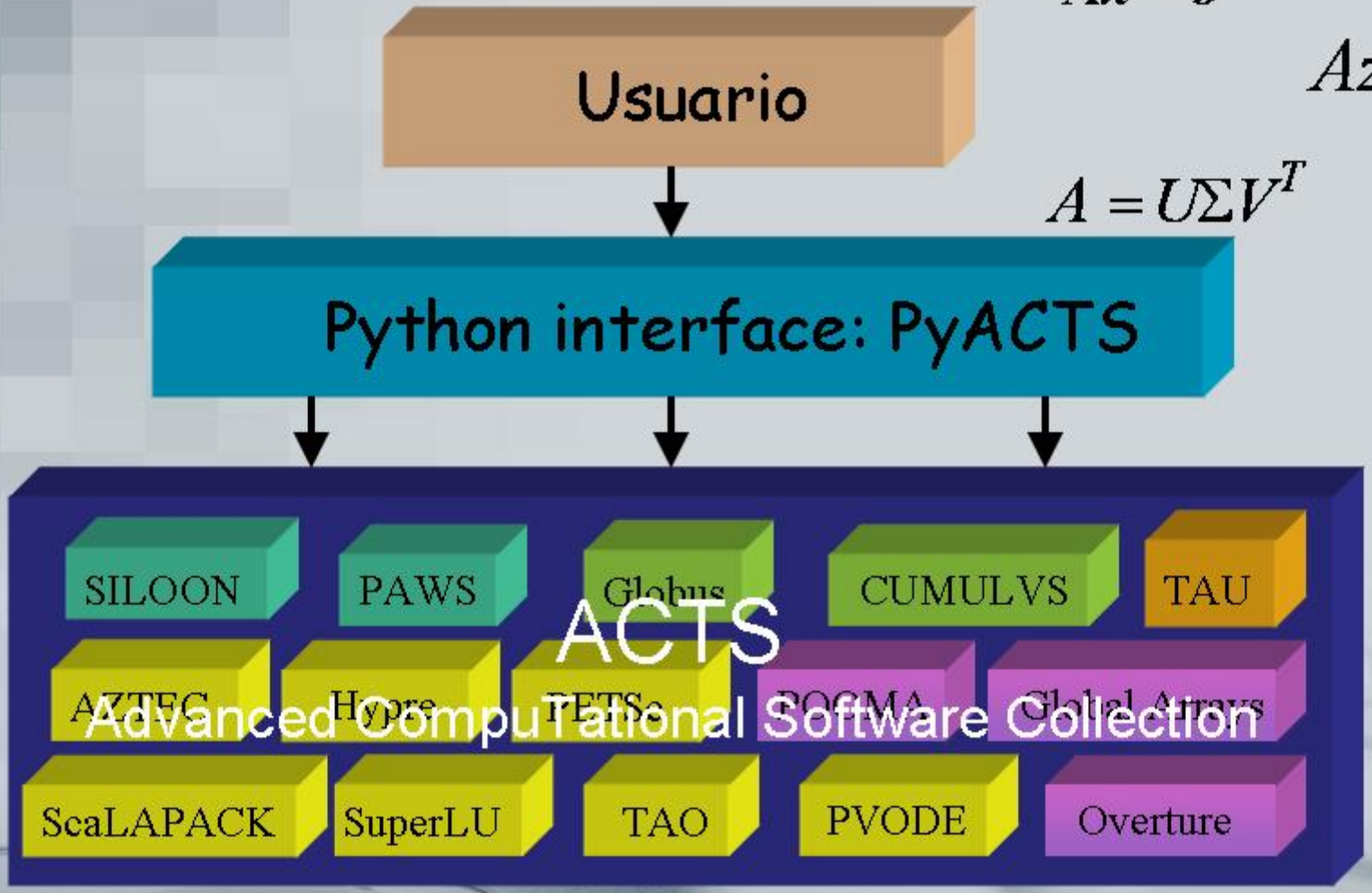
- Creación de interfaces en lenguajes de alto nivel.
- Uso transparente del entorno paralelo.
- Reconfiguración flexible.
- Interoperatibilidad.
- Elección del lenguaje: Python (interpretado, interactivo, orientado a objetos, libre, integrable, ...)

1. Motivación

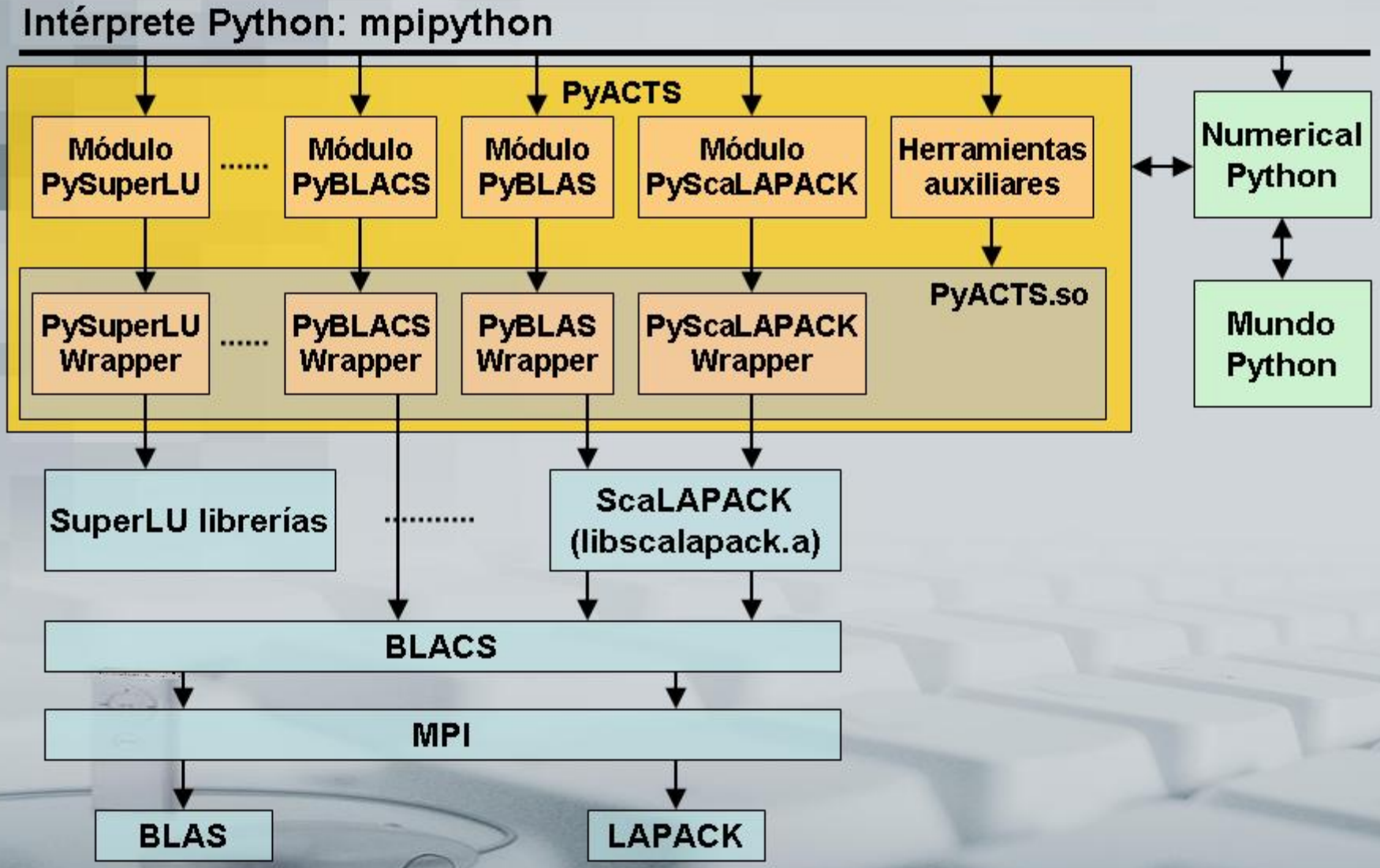
$$Ax = b$$

$$Az = \lambda z$$

$$A = U\Sigma V^T$$



2. Estructura de PyACTS



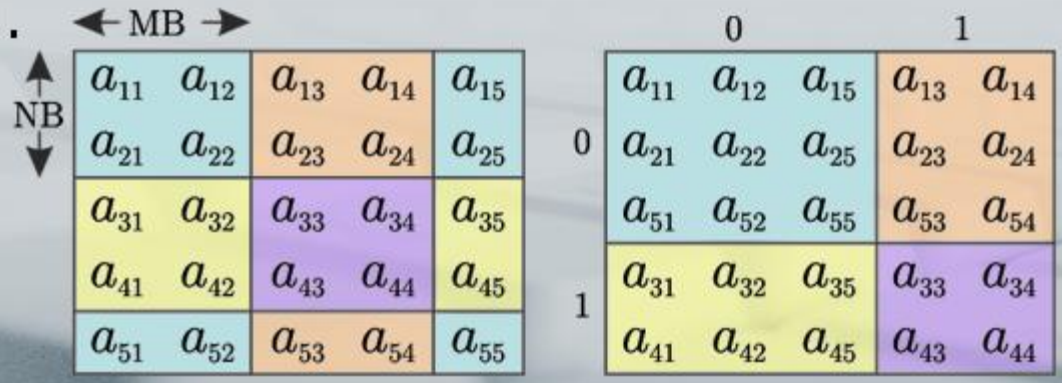
2. Estructura de PyACTS

En una implementación paralela que use ScaLAPACK, un usuarios debe:

- Conocer detalles del entorno paralelo.
- Inicializar la malla de procesos usando BLACS.

		npcol=2	
		p_c=0	p_c=1
nprow=2	p_r=0	0	1
	p_r=1	2	3

- Distribuir y/o recibir datos siguiendo la distribución cíclica por bloques 2D.



2. Estructura de PyACTS

Estas tareas pueden ser difíciles de implementar para usuarios no avanzados.

Sin embargo, el paralelismo no tiene porque ser complicado de implementar.

Ejemplo:

```
import PyACTS
import Numeric
PyACTS.gridinit()                          # Inicializamos la malla de procesos
ACTS_lib=PyACTS.ScaLAPACK_ID               # Librería ScaLAPACK
   ...                                     # El proceso master define
   ...                                     #   o lee tres matrices
   ...                                     #   A, B y C
A=PyACTS.Num2PyACTS(A,ACTS_lib)            # Distribuimos la matriz A
B=PyACTS.Num2PyACTS(B,ACTS_lib)            # Distribuimos la matriz B
C=PyACTS.Num2PyACTS(C,ACTS_lib)            # Distribuimos la matriz C
C=PyPBLAS.pvgemm(alpha,A,B,beta,C)         # Calculamos alpha·AB + beta·C
PyACTS.gridexit()
```

3. Herramientas auxiliares

Necesitamos un conjunto de rutinas que realicen y comprueben los pasos necesarios para la correcta ejecución de rutinas en niveles inferiores:

Primer grupo. Inicializar la malla de procesos y liberar recursos:

gridinit(nb=32, mb =32, nprow=4, npcol=1)

Segundo grupo. Validar argumentos y datos.

Tercer grupo. Servicios básicos:

- Obtener dimensiones globales: *m,n=getdims(x)*
- Obtener el descriptor: *readACTSdesc(ACTSArray)*
- Crear un descriptor: *ACTSmakedesc(m,n)*

3. Herramientas auxiliares

Cuarto grupo. Conversión de variables, distribución de datos y manejo del I/O:

- Distribución: *Num2PyACTS / PyACTS2Num*
- Generación: *Rand2PyACTS*
- Lectura ASCII: *Txt2PyACTS / PyACTS2Txt*
- Lectura netCDF: *PnetCDF2PyACTS / PyACTS2PnetCDF*

3. Herramientas auxiliares

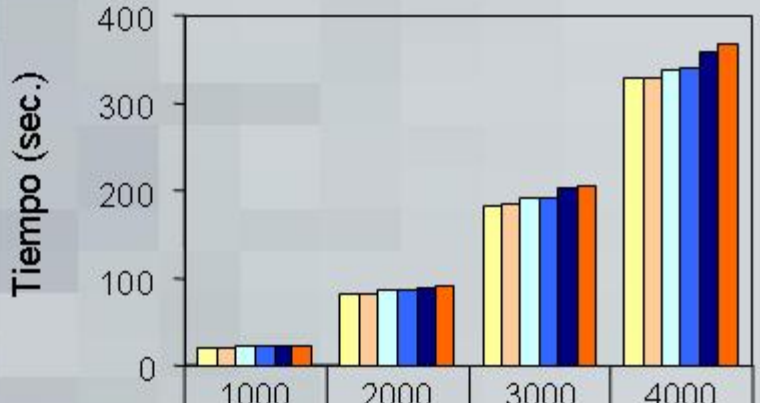
EJEMPLO

```
import PyACTS
import Numeric
PyACTS.gridinit(nb=2,mb=2)
n=8
ACTS_lib=PyACTS.ScaLAPACK_ID          # Librería ScaLAPACK
if PyACTS.iread==1:                   # Sólo el nodo master
  a=Numeric.reshape(range(n*n),[n,n]) # genera una matriz
else:                                 # para distribuirla
  a=None
a=PyACTS.Num2PyACTS(a,ACTS_lib)       # Convierte Numpy a PyACTS
    ......
    ......                            # Operaciones con a
    ......
datafile='./data_out.txt'
PyACTS.PyACTS2Txt(a,datafile)         # Escribir resultados en archivo
PyACTS.gridexit()
```

3. Herramientas auxiliares

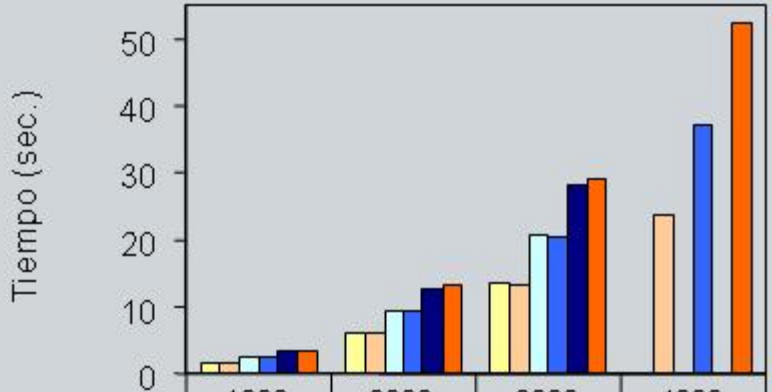
Comparativa tiempos de distribución y recolección

Plataforma: Seaborg (IBM SP RS/6000)



	1000	2000	3000	4000
Fortran 2X1	20.62	81.54	183.20	327.65
Python 2X1	21.28	81.80	183.99	327.70
Fortran 2X2	21.68	85.46	191.16	339.09
Python 2X2	22.25	85.25	191.11	339.50
Fortran 4X4	22.50	89.61	201.01	358.25
Python 4X4	22.63	90.73	204.58	367.04

Tamaño de la matriz (n x n)
Txt2PyACTS / PyACTS2Txt



	1000	2000	3000	4000
Fortran 2X1	1.55	6.16	13.41	
Python 2X1	1.47	6.01	13.12	23.68
Fortran 2X2	2.37	9.33	20.47	
Python 2X2	2.30	9.20	20.30	37.09
Fortran 4X4	3.24	12.86	28.18	
Python 4X4	3.23	13.17	29.15	52.37

Tamaño de la matriz (n x n)
Num2PyACTS / PyACTS2Num

4. Los interfaces PyBLACS, PyPBLAS y PyScaLAPACK

Características

- Uso transparente del entorno paralelo
- No es necesarios indicar el tipo de dato con el que se trabaja (DOUBLE, REAL, ...).
- Simplificación del número de parámetros en cada rutina.

$\alpha \cdot A \cdot B + \beta \cdot C$ (FORTRAN)

PvGEMM(~~TRANSA, TRANSB~~, ~~M, N~~, ALPHA,
A, ~~IA, JA~~, ~~DESCA~~,
B, ~~IB, JB~~, ~~DESCB~~,
BETA, C, ~~IC, JC~~, ~~DESCC~~)

$\alpha \cdot A \cdot B + \beta \cdot C$ (Python)

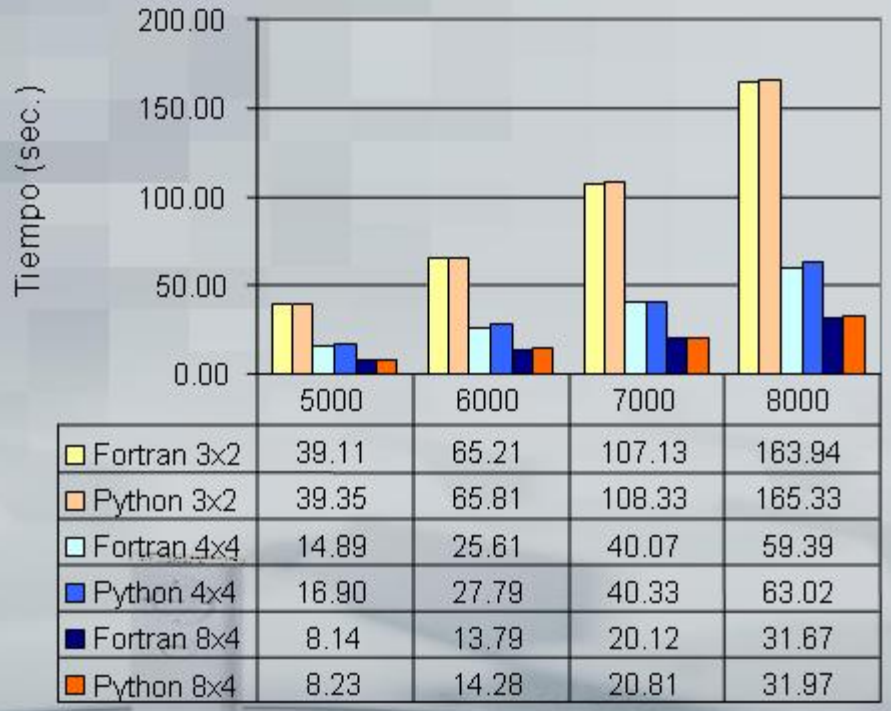
PvGEMM(ALPHA, A, B, BETA, C)

4. Los interfaces PyBLACS, PyPBLAS y PyScaLAPACK

Características

No hay penalización de tiempo frente a su uso desde otros lenguajes compilados como Fortran o C.

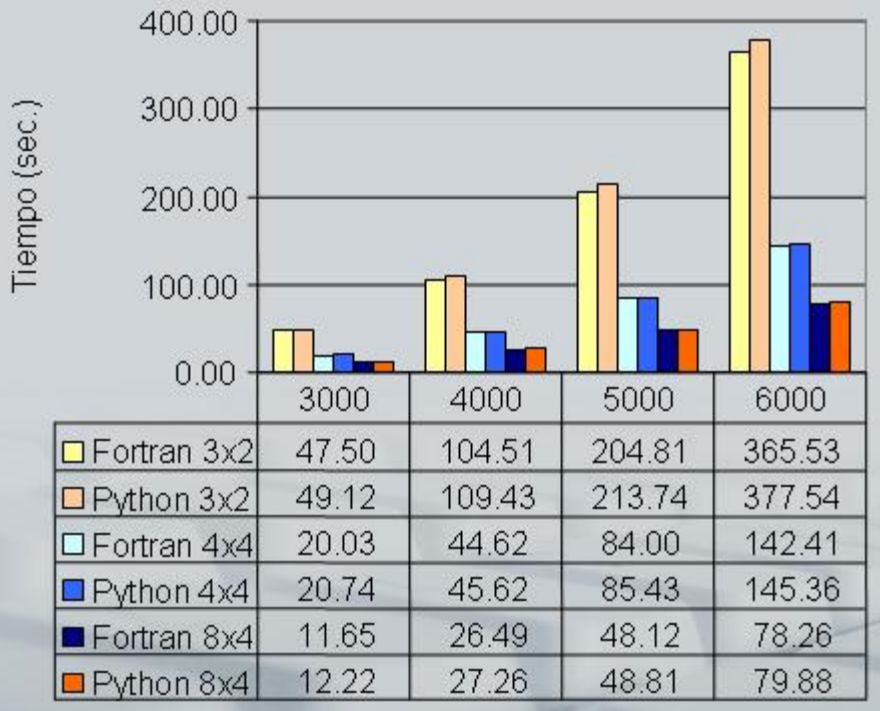
Plataforma: Jacquard (320-dual Opteron Cluster, Linux)



	5000	6000	7000	8000
Fortran 3x2	39.11	65.21	107.13	163.94
Python 3x2	39.35	65.81	108.33	165.33
Fortran 4x4	14.89	25.61	40.07	59.39
Python 4x4	16.90	27.79	40.33	63.02
Fortran 8x4	8.14	13.79	20.12	31.67
Python 8x4	8.23	14.28	20.81	31.97

Tamaño de la matriz

PDGEMM



	3000	4000	5000	6000
Fortran 3x2	47.50	104.51	204.81	365.53
Python 3x2	49.12	109.43	213.74	377.54
Fortran 4x4	20.03	44.62	84.00	142.41
Python 4x4	20.74	45.62	85.43	145.36
Fortran 8x4	11.65	26.49	48.12	78.26
Python 8x4	12.22	27.26	48.81	79.88

Tamaño de la matriz

PDGESVD

5. Aplicaciones

PyClimate

PyClimate es un paquete Python diseñado para realizar tareas comunes en el análisis de la variabilidad climática. Posee funciones para:

- Manejar operaciones de I/O usando formato netCDF
- Análisis EOF
- Análisis SVD
- Filtros digitales multivariantes (Kolmogorov-Zurbenko, Lanczos)
- Estimación de funciones de densidad de probabilidad
- Acceso a la librería DCDFLIB.C desde Python

APROXIMACIÓN SECUENCIAL

5. Aplicaciones

PyClimate

PyClimate hace uso de interfaces Python a la librería LAPACK.

Con PyScaLAPACK podemos paralelizar algunas de las aplicaciones de PyClimate.

Ejemplo basado en un estudio metereológico a partir del análisis EOF y SVD.

Éste es un ejemplo en el que se usa PyACTS para paralelizar una aplicación secuencial.

ANÁLISIS EOF

El propósito original del análisis de Funciones Ortogonales Empíricas (EOF) consiste en eliminar la información redundante en una muestra de datos con la mínima pérdida de variabilidad.

Los EOF's pueden ser obtenidos calculando la descomposición SVD de la matriz de datos, una vez eliminada la media de cada columna en cada una de ellas.

Hemos usado PyScaLAPACK para la descomposición SVD, y PyPBLAS para los cálculos adicionales.

ANÁLISIS EOF (Resultados)

Datos: Temperatura del aire (nivel 1000mb) medida en una malla global de 2.5º (latitud) x 2.5º (longitud), con un total de 144x73 puntos:

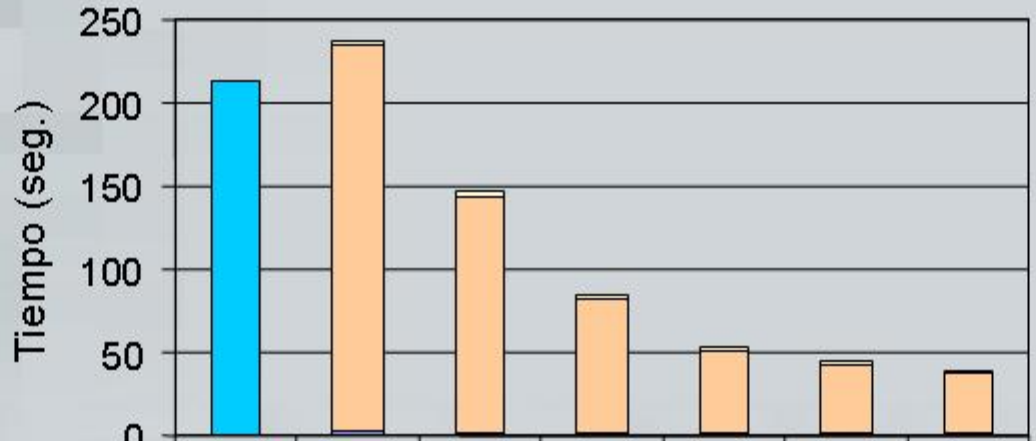
- Media mensual desde 1948: 703 medidas (*air.month.1948*)
- Medida diaria 1998-2005: 2922 medidas (*air.day.1998*)

Fuente: Climate Diagnostics Center (*www.cdc.noaa.gov*).

Formato: netCDF.

ANÁLISIS EOF (Resultados)

Datos: Media mensual desde 1948: 703 medidas (*air.month.1948*)
Plataforma: Linux cluster (6 Intel CPU, 2GHZ) Gigabit ethernet.



	Seq.	p=1	p=2	p=3	p=4	p=5	p=6
Recolección		1.9	2.4	2.1	2.0	2.0	2.0
Análisis EOF	213.6	233.0	142.3	81.5	49.6	41.4	36.0
Distribución		1.8	1.6	1.4	1.4	1.3	1.6

Número de procesadores

ANÁLISIS EOF (Resultados)

Datos: Medida diaria 1998-2005: 2922 medidas (*air.day.1998*)
Plataforma: Seaborg (IBM SP RS/6000).

	Seq.	p=1	p=2	p=4	p=8	p=16
Recolección				13.8	13.8	13.8
Análisis EOF				1251.7	503.6	205.8
Distribución				6.6	5.2	4.7

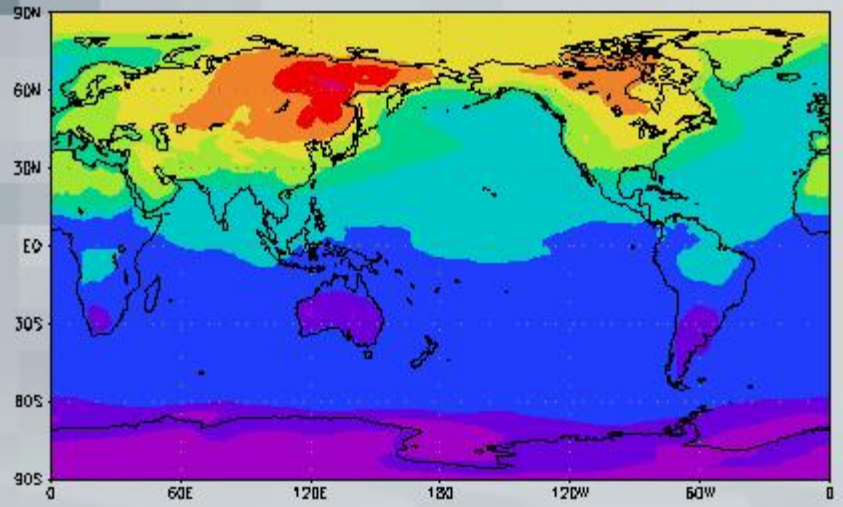
Número de procesadores

5. Aplicaciones

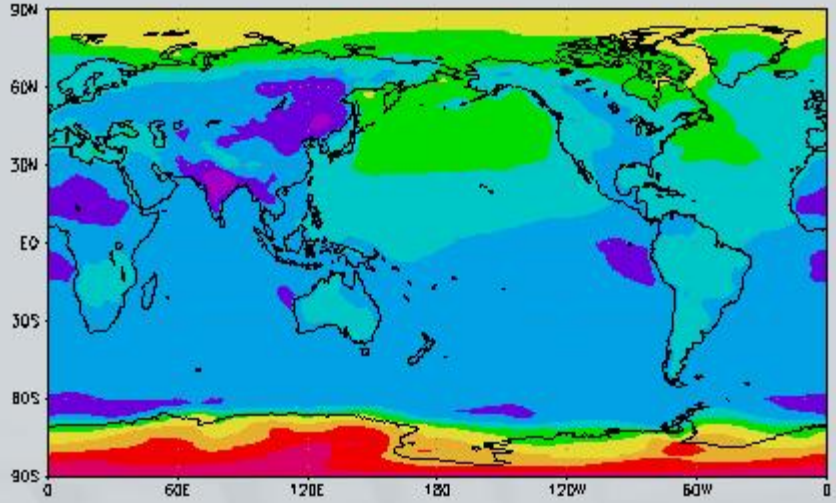
PyClimate

ANÁLISIS EOF (Resultados)

Datos: Medida diaria 1998-2005: 2922 medidas (*air.day.1998*)



Primer EOF. Varianza: 71.32%



Segundo EOF. Varianza: 5.55%

ANÁLISIS EOF (Conclusiones)

1. El script paralelo mantiene prácticamente la misma estructura que el script secuencial.
2. Se ha paralelizado una aplicación secuencial, con una reducción considerable de los tiempos de ejecución.
3. Se consigue ejecutar casos que desde una aproximación secuencial serían inviables, incrementando el tamaño de los conjuntos de datos que pueden ser analizados.

5. Aplicaciones

Geofísica

Modelos físicos para analizar la estructura interna de la Tierra

Implementación inicial: Fortran usando ScaLAPACK

Método: Usa una versión por bloques del algoritmo de Lanczos.

Hemos usado PyACTS para conseguir una versión más sencilla del código.

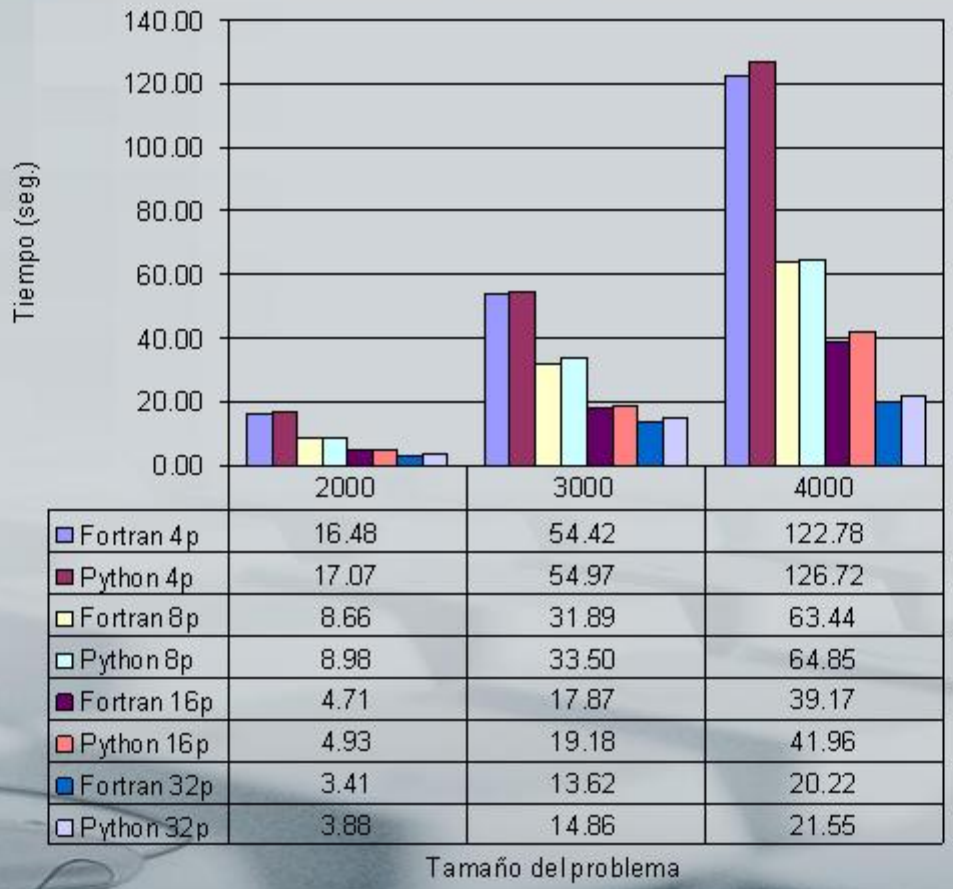
El código Fortran requiere unas 250 líneas de código, mientras que el código Python sólo necesita 75.

5. Aplicaciones

Geofísica

Modelos físicos para analizar la estructura interna de la Tierra (Resultados)

Plataforma: Jacquard (320-dual Opteron Cluster, Linux)



	2000	3000	4000
Fortran 4p	16.48	54.42	122.78
Python 4p	17.07	54.97	126.72
Fortran 8p	8.66	31.89	63.44
Python 8p	8.98	33.50	64.85
Fortran 16p	4.71	17.87	39.17
Python 16p	4.93	19.18	41.96
Fortran 32p	3.41	13.62	20.22
Python 32p	3.88	14.86	21.55

Modelos físicos para analizar la estructura interna de la Tierra (Conclusiones)

1. Uso de PyACTS en una aplicación real sin una penalización en el rendimiento.
2. Sencillez de uso: 75 líneas de código frente a las 250 de la implementación Fortran.

Accesos desde 01/01/2007

Esta siendo utilizado por la comunidad científica:

Spain	374
United States	266
France	107
Germany	62
China	27
Italy	25
United Kingdom	25
Japan	20
Russia	18
Poland	18

Referencias

• L.A. Drummond, V. Galiano, V. Migallón, and J. Penadés.
Improving ease of use in BLACS and PBLAS with Python.
Proceedings of Parallel Computing 2005 (ParCo 2005), NIC Series, volume 33, John von Neumann Institute for Computing, Germany, 2005.

• L.A. Drummond, V. Galiano, O. Marques, V. Migallón, and J. Penadés.
PyACTS: A high-level framework for fast development of high performance applications.
Lecture Notes in Computer Science, 4395, 417-425, 2007.

• L.A. Drummond, V. Galiano, V. Migallón, and J. Penadés.
High-level user interfaces for the DOE ACTS Collection.
Lecture Notes in Computer Science, 4699, 251-259, 2007.

• L.A. Drummond, V. Galiano, V. Migallón, and J. Penadés.
PyACTS: A Python based interface to ACTS tools and parallel scientific applications.
International Journal of Parallel Programming, to appear.

PyACTS: Una interface Python para las herramientas ACTS

Jose Penadés

Grupo de Computación de Altas Prestaciones y Paralelismo

Universidad de Alicante

Universitat d'Alacant
Universidad de Alicante

Vicente Galiano
Héctor Migallón
Universidad Miguel Hernández
Violeta Migallón
Universidad de Alicante
L.A. Drummond
National Berkeley Laboratory