Parallel Access to netCDF files in High Performance Aplications from High-Level Frameworks

V. Galiano

joint work with H. Migallón, V.Migallón and J. Penadés

Dpto. Física y Arquitectura de Computadores
Universidad Miguel Hernández
Elche (Alicante), Spain

Harrachov, Czech Republic
August 24, 2007

Outline

Outline

Motivation

Scientific and engineering applications usually ...

- Require knowledges about an specific programming language.
- Use libraries developed by others.
- Are platform dependent.
- Computational requirements grow with size of the problem.

Motivation

Scientific and engineering applications usually . . .

- Require knowledges about an specific programming language.
- Use libraries developed by others.
- Are platform dependent.
- Computational requirements grow with size of the problem.

Motivation

Scientific and engineering applications usually . . .

- Require knowledges about an specific programming language.
- Use libraries developed by others.
- Are platform dependent.
- Computational requirements grow with size of the problem.

High **P**erformance **C**omputing Systems *HPC*

- More complexity in libraries
- Need the execution in a multiprocessor system with . . .
  - Shared memory
  - Distributed memory
- Require Data distribution
- Syncronize Parallel execution

Example of computational requirements and motivation I

PyClimate

- Sequential application programmed in Python.
- Designed to accomplish some usual tasks during the analysis of climate variability.
- Open source and free distribution.

Example of computational requirements and motivation I

PyClimate 1.2

PyClimate

- Sequential application programmed in Python.
- Designed to accomplish some usual tasks during the analysis of climate variability.
- Open source and free distribution.

Utilities:

- Empirical Orthogonal Functions (EOFs) Analisys.
- Canonical Correlation Analysis (CCA).
- SVD of coupled datasets.
- Simple multivariate statistical tools
- Digital filters.
- Input/Output functions.

Example of computational requirements and motivation II

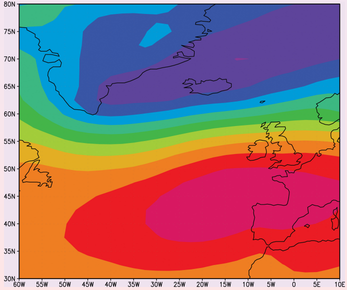
EOFs Analysis

- Delete redundant information with minimun lost of variability.
- Efficient method for data compression.
- Useful in high dimensionality spaces.
- We can get EOFs analysis based on the Singular Value Decomposition *SVD*.

EOFs Analysis

- Delete redundant information with minimun lost of variability.
- Efficient method for data compression.
- Useful in high dimensionality spaces.
- We can get EOFs analysis based on the Singular Value Decomposition *SVD*.

Analisys of a netCDF file containing sea level pressure in *NAO* region

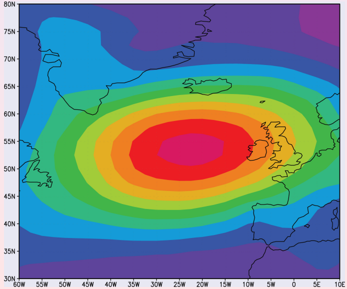


1st EOF accounting for
55,81% of the variance

EOFs Analysis

- Delete redundant information with minimun lost of variability.
- Efficient method for data compression.
- Useful in high dimensionality spaces.
- We can get EOFs analysis based on the Singular Value Decomposition *SVD*.

Analisys of a netCDF file containing sea level pressure in *NAO* region



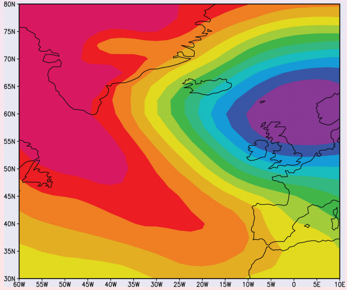
2nd EOF accounting for
20,38% of the variance

Example of computational requirements and motivation II

EOFs Analysis

- Delete redundant information with minimun lost of variability.
- Efficient method for data compression.
- Useful in high dimensionality spaces.
- We can get EOFs analysis based on the Singular Value Decomposition *SVD*.

Analisys of a netCDF file containing sea level pressure in *NAO* region



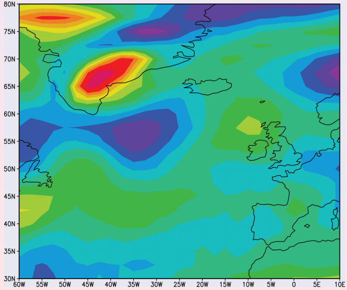
3rd EOF accounting for

el 11,62% of the variance

EOFs Analysis

- Delete redundant information with minimun lost of variability.
- Efficient method for data compression.
- Useful in high dimensionality spaces.
- We can get EOFs analysis based on the Singular Value Decomposition *SVD*.

Analisys of a netCDF file containing sea level pressure in *NAO* region



20th accounting for

el 0,03% of the variance

¿Scalability?

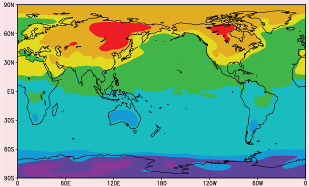
- NetCDF files obtained from Climate Diagnostics Center (`www.cdc.noaa.gov`).
- Extending data grid 144×73.
- Including Data grids at different elevations $17 \times 144 \times 73$.
- Increasing time axis.
  - `air.mon.ltm.nc`:
  - `air.day.ltm.nc`:
  - `air.mon.mean.nc`:
  - `air.1988to2005.nc`

EOFs Analisys with PyClimate

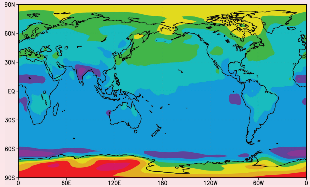
HARRACHOV 2007
V. Galiano
Introduction
Motivation
Python
PyACTS
PyPnetCDF
Aplications
Conclusions

¿Scalability?

- NetCDF files obtained from Climate Diagnostics Center (`www.cdc.noaa.gov`).
- Extending data grid 144×73.
- Including Data grids at different elevations $17 \times 144 \times 73$.
- Increasing time axis.
  - `air.mon.ltm.nc`: $\rightarrow 17 \times 12 \times 144 \times 73$ 8,5MB
  - `air.day.ltm.nc`:
  - `air.mon.mean.nc`:
  - `air.1988to2005.nc`



1ª EOF (90,64%)

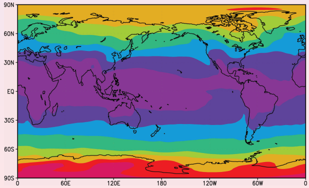


2ª EOF (7,19%)

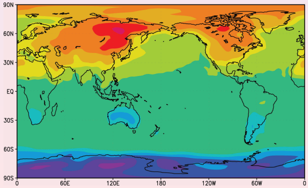
EOFs Analisys with PyClimate

¿Scalability?

- NetCDF files obtained from Climate Diagnostics Center (`www.cdc.noaa.gov`).
- Extending data grid 144×73.
- Including Data grids at different elevations $17 \times 144 \times 73$.
- Increasing time axis.
  - `air.mon.ltm.nc`:
  - `air.day.ltm.nc`: $\rightarrow 17 \times 365 \times 144 \times 73$ 1,3GB
  - `air.mon.mean.nc`:
  - `air.1988to2005.nc`



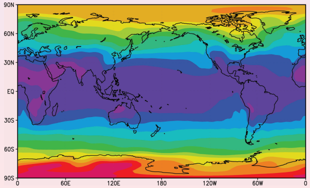
1ª EOF (99,72%)



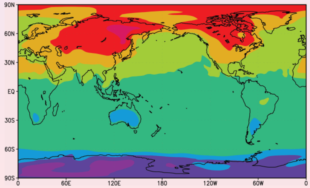
2ª EOF (0,253%)

¿Scalability?

- NetCDF files obtained from Climate Diagnostics Center (`www.cdc.noaa.gov`).
- Extending data grid 144×73.
- Including Data grids at different elevations $17 \times 144 \times 73$.
- Increasing time axis.
  - `air.mon.ltm.nc`:
  - `air.day.ltm.nc`:
  - `air.mon.mean.nc`: $\rightarrow 17 \times 703 \times 144 \times 73$ 2,5GB
  - `air.1988to2005.nc`



1ª EOF (99,60%)

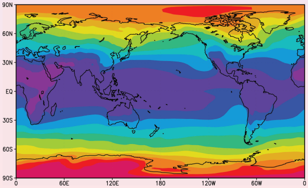


2ª EOF (0,34%)

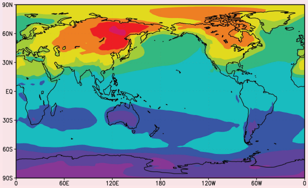
EOFs Analisys with PyClimate

¿Scalability?

- NetCDF files obtained from Climate Diagnostics Center (`www.cdc.noaa.gov`).
- Extending data grid 144×73.
- Including Data grids at different elevations $17 \times 144 \times 73$.
- Increasing time axis.
  - `air.mon.ltm.nc`:
  - `air.day.ltm.nc`:
  - `air.mon.mean.nc`:
  - `air.1988to2005.nc` $\rightarrow 17 \times 2922 \times 144 \times 73$ 4,17GB



1ª EOF (99,87%)



2ª EOF (0,08%)

¿Scalability?

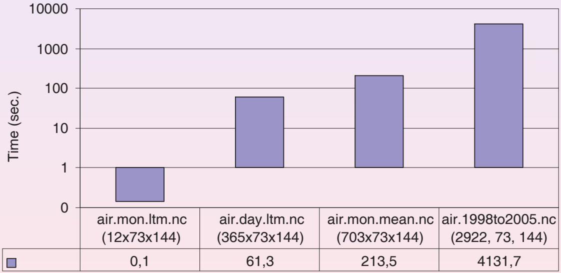
- Increasing execution times.
- Limited memory resources.

	air.mon.ltm.nc (12x73x144)	air.day.ltm.nc (365x73x144)	air.mon.mean.nc (703x73x144)	air.1998to2005.nc (2922, 73, 144)
■	0,1	61,3	213,5	4131,7

EOFs Analisys with PyClimate

¿Scalability?

- Increasing execution times.
- Limited memory resources.



⇓

We need an scalable solution to obtain EOFs Analisys

Índice

Python Introduction

Python is

- Free Distribution.
- Scripting language: No compile or link steps.
- No type declarations.
- High-level data-types and operations.
- Extending and embedding in C as system bond.
- Run in either batch mode or interactive mode.

Execution Modes

- Batch Mode.
- Interactive mode

Execution Modes

- Batch Mode.

```
vgaliano@node0:~$ python hello.py
Hello World
```

- Interactive mode

Execution Modes

- Batch Mode.

```
vgaliano@node0:~$ python hello.py
Hello World
```

- Interactive mode

```
vgaliano@node0:~$ python
Python 2.4.3 (#4, May 12 2006, 19:00:23)
[GCC 3.3.5 (Debian 1:3.3.5-13)] on linux2
Type "help", "copyright", "credits" or "license" for
more information.
>>> print "Hello World"
Hello World
```

Main used packages: Numeric Python

Numeric Python

- Adds a fast multidimensional array facility to Python.
- `array` object : homogeneous collections of potentially large numbers of numbers.
- Similiar arrays management like Matlab, Fortran, Basis and others.
- Operations performed *elementwise* on the arguments of the operation
- Same data types between C/Fortran y Python.
- Contiguous arrays performs better times (`PyArray_ContiguousFromObject()`).

Main used packages: Numeric Python

Numeric Python

- Adds a fast multidimensional array facility to Python.
- `array` object : homogeneous collections of potentially large numbers of numbers.
- Similiar arrays management like Matlab, Fortran, Basis and others.
- Operations performed *elementwise* on the arguments of the operation
- Same data types between C/Fortran y Python.
- Contiguous arrays performs better times (`PyArray_ContiguousFromObject()`).

```
>>> from Numeric import *
>>> vector1 = array([1,2,3,4,5])
>>> print vector1
[1 2 3 4 5]
```

Main used packages: Numeric Python

Numeric Python

- Adds a fast multidimensional array facility to Python.
- `array` object : homogeneous collections of potentially large numbers of numbers.
- Similiar arrays management like Matlab, Fortran, Basis and others.
- Operations performed *elementwise* on the arguments of the operation
- Same data types between C/Fortran y Python.
- Contiguous arrays performs better times (`PyArray_ContiguousFromObject()`).

```
>>> a = arrayrange(9)
>>> a.shape = (3,3)
>>> print a
[[0 1 2]
[3 4 5]
[6 7 8]]
>>> print a[0]
[0 1 2]
>>> print a[1,0]
3
```

Main used packages: Numeric Python

Numeric Python

- Adds a fast multidimensional array facility to Python.
- `array` object : homogeneous collections of potentially large numbers of numbers.
- Similiar arrays management like Matlab, Fortran, Basis and others.
- Operations performed *elementwise* on the arguments of the operation
- Same data types between C/Fortran y Python.
- Contiguous arrays performs better times (`PyArray_ContiguousFromObject()`).

```
>>> print a
[1 2 3]
>>> print a * 3
[3 6 9]
>>> print a + 3
[4 5 6]
>>> print sin(a)
[ 0.84147098 0.90929743 0.14112001]
```

Main used packages: Numeric Python

Numeric Python

- Adds a fast multidimensional array facility to Python.
- `array` object : homogeneous collections of potentially large numbers of numbers.
- Similiar arrays management like Matlab, Fortran, Basis and others.
- Operations performed *elementwise* on the arguments of the operation
- Same data types between C/Fortran y Python.
- Contiguous arrays performs better times (`PyArray_ContiguousFromObject()`).

```
>>> print a
[1 2 3]
>>> print a * 3
[3 6 9]
>>> print a + 3
[4 5 6]
>>> print sin(a)
[ 0.84147098 0.90929743 0.14112001]
```

Main used packages: Numeric Python

Numeric Python

- Adds a fast multidimensional array facility to Python.
- `array` object : homogeneous collections of potentially large numbers of numbers.
- Similiar arrays management like Matlab, Fortran, Basis and others.
- Operations performed *elementwise* on the arguments of the operation
- Same data types between C/Fortran y Python.
- Contiguous arrays performs better times (`PyArray_ContiguousFromObject()`).

```
>>> a=reshape(range(9),[3,3])
>>> a
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
>>> b=a[:,0]
>>> b
array([0, 3, 6])
>>> print a.iscontiguous(), b.iscontiguous()
1 0
```

Main used packages: Scientific Python y pyMPI

Scientific Python

- Collection of Python modules that are useful for scientific computing.
- `ScientificPython.MPI`.
- `ScientificPython.IO.NetCDF`.

Main used packages: Scientific Python y pyMPI

Scientific Python

- Collection of Python modules that are useful for scientific computing.
- `ScientificPython.MPI`.
- `ScientificPython.IO.NetCDF`.

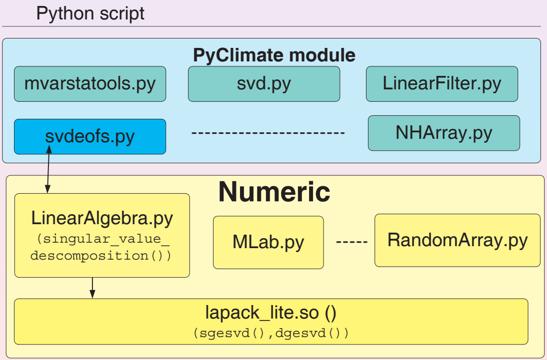
pyMPI

- Integrating the Message Passing Interface (MPI) into the Python interpreter.
- Asynchronous interactive execution.

PyClimate structure

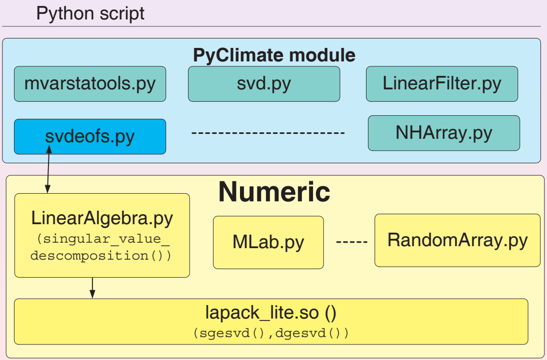
Software libraries used in EOFs Analysis

- NetCDF access: ScientificPython.
- Numeric Modules: Numeric Python (actually Numarray).
- C libraries: `lapack_lite.so`.



PyClimate structure

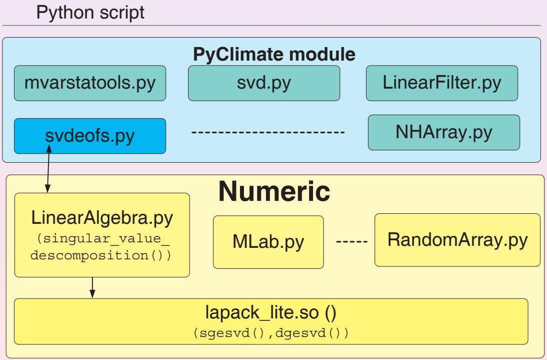
Parallelization of EOFs Analysis



PyClimate structure

Parallelization of EOFs Analysis

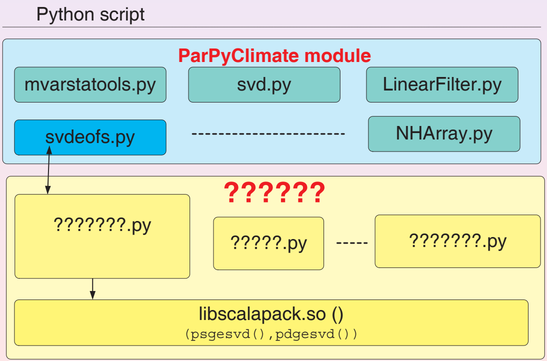
- Messages Passing Interface (PVM, MPI, . . .)
- `Xgesvd` $\in$ LAPACK $\Longrightarrow$ `pXgesvd` $\in$ `ScaLAPACK`



PyClimate structure

Parallelization of EOFs Analysis

- Messages Passing Interface (PVM, MPI, . . .)
- `Xgesvd` $\in$ LAPACK $\Longrightarrow$ `pXgesvd` $\in$ `ScaLAPACK`

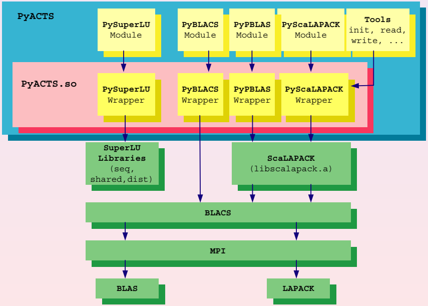


Outline

PyACTS Package

High Level Interfaces and Auxiliary routines to the ACTS Collection:

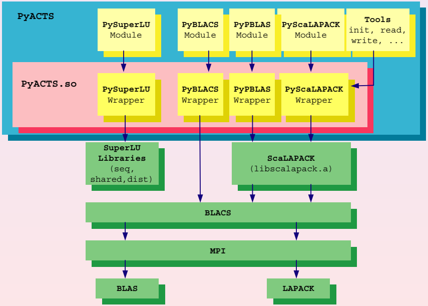
- PyBLACS.
- PyPBLAS.
- PyScaLAPACK.



PyACTS Package

High Level Interfaces and Auxiliary routines to the ACTS Collection:

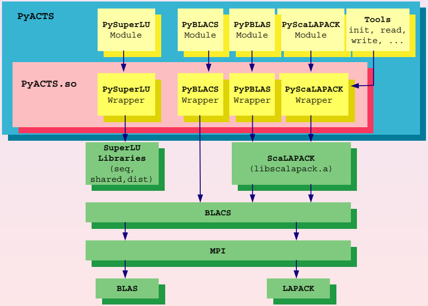
- PyBLACS.
- PyPBLAS.
- PyScaLAPACK.



PyACTS Package

High Level Interfaces and Auxiliary routines to the ACTS Collection:

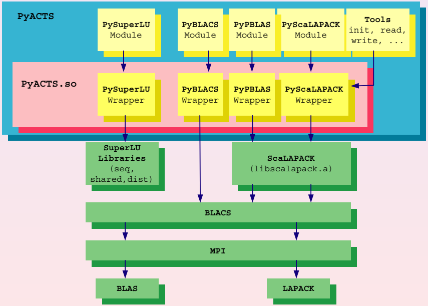
- PyBLACS.
- PyPBLAS.
- PyScaLAPACK.



PyACTS Package

High Level Interfaces and Auxiliary routines to the ACTS Collection:

- PyBLACS.
- PyPBLAS.
- PyScaLAPACK.

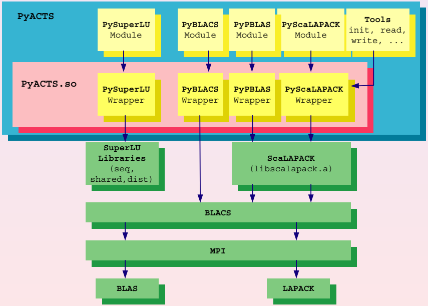


PyACTS Package

High Level Interfaces and Auxiliary routines to the ACTS Collection:

- PyBLACS.
- PyPBLAS.
- PyScaLAPACK.

- Support functionality:
  - Basic Services
  - I/O Services
  - Errors and Exceptions



Plattforms used in tests

Cluster-umh | Seaborg

6 Intel Pentium 4, 2 GHz with 1 Gbyte RAM connected with a Switch Gigabit Ethernet and with operative system Linux (Debian distribution). Node0 shares its hard disk with NFS.

Plattforms used in tests

Cluster-umh | Seaborg

IBM SP RS/6000 with 6080 processors. Each processor has a peak performance of 1.5 GFlops and the processors are distributed among 380 compute nodes with 16 processors per node. Disk Access with **G**lobal **P**arallel **F**ile **S**ystem

Communications about PyACTS

- L. A. Drummond, V. Galiano, V. Migallón, and J. Penadés. *High Level User Interfaces for High Performance Libraries in Linear Algebra: PyBLACS and PyPBLAS*. In *12th International Linear Algebra Society Conference*, University of Regina, Regina, Saskatchewan, Canada, June 2005.
- L. A. Drummond, V. Galiano, V. Migallón, and J. Penadés. Improving ease of use in BLACS and PBLAS with Python. Proceedings of the International Conference ParCo. September 2005.
- L. A. Drummond, V. Galiano, V. Migallón, and J. Penadés. PyACTS: A High-Level Framework for Fast Development of High Performance Applications. In *Proceedings from Seventh International Meeting on High Performance Computing for Computational Science - VECPAR'06*, pages 373–378, Rio de Janeiro, Brazil, July 2006.
- L. A. Drummond, V. Galiano, J. Penadés, and V. Migallón. An introduction to PyACTS. In *SIAM Conference on Computational Science and Engineering*. SIAM Activity Group on Computational Science and Engineering, Orlando, Florida, February 2005.
- L. A. Drummond, V. Galiano, J. Penadés, and V. Migallón. High Level User Interfaces for Numerical Linear Algebra Libraries: PyScaLAPACK. In *Proceedings of the Fifth International Conference on Engineering Computational Technology*, pages 1–10. Civil-Comp Ltd., September 2006.

Communications about PyACTS

- L. A. Drummond, V. Galiano, J. Penadés, and V. Migallón. PyACTS: A High-Level Framework for Fast Development of High Performance Applications. In *Proceeding of the SIAM Conference on Parallel Processing for Scientific Computing*. SIAM Activity Group on Supercomputing, San Francisco, California, February 2006.
- L. A. Drummond, V. Galiano, J. Penadés, and V. Migallón. High-level User Interfaces for the DOE ACTS Collection. In *PARA06: Proceeding of the Workshop on state-of-the-art in scientific and parallel computing*. Umea University, Sweden, June 2006.

PyACTS Publications

- L. A. Drummond, V. Galiano, V. Migallón, and J. Penadés. PyACTS: A High-Level Framework for Fast Development of High Performance Applications. In *Lectures Notes in Computer Science*, 4395: 417–425, 2007.
- L. A. Drummond, V. Galiano, J. Penadés, and V. Migallón. High-level User Interfaces for the DOE ACTS Collection. To appear in *Lecture Notes in Computer Science*.
- L. A. Drummond, V. Galiano, V. Migallón, and J. Penadés. Improving ease of use in BLACS and PBLAS with Python. In G.R. Joubert, W.E. Nagel, F.J. Peters, O. Plata, P. Tirado, and E. Zapata, editors, *Parallel Computing: Current & Future Issues of High-End Computing*, volume 33. NIC series, September 2006. ISBN 3-00-017352-8.
- L. A. Drummond, V. Galiano, J. Penadés, and V. Migallón. High Level User Interfaces for Numerical Linear Algebra Libraries: PyScaLAPACK. Submitted to *International Journal of Computers and Structures (CAS)*.

PyScaLAPACK Module

- Transparent and simplified access to ScaLAPACK from Python.

PyScaLAPACK Module

- Transparent and simplified access to ScaLAPACK from Python.
- Same names than in ScaLAPACK.

PyScaLAPACK Module

- Transparent and simplified access to ScaLAPACK from Python.
- Same names than in ScaLAPACK.
- Interaction with PyBLACS and PyPBLAS.

PyScaLAPACK Module

- Transparent and simplified access to ScaLAPACK from Python.
- Same names than in ScaLAPACK.
- Interaction with PyBLACS and PyPBLAS.
- PyScaLAPACK is classified in two groups:

PyScaLAPACK Module

- Transparent and simplified access to ScaLAPACK from Python.
- Same names than in ScaLAPACK.
- Interaction with PyBLACS and PyPBLAS.
- PyScaLAPACK is classified in two groups:
  - Driver Routines
    1. Linear Equations
    2. Linear Least Squares Problems
    3. Standard Eigenvalue and Singular Value Problems
    4. Generalized Symmetric Definite Eigenproblems (GSEP)

PyScaLAPACK Module

- Transparent and simplified access to ScaLAPACK from Python.
- Same names than in ScaLAPACK.
- Interaction with PyBLACS and PyPBLAS.
- PyScaLAPACK is classified in two groups:
  - Driver Routines
    1. Linear Equations
    2. Linear Least Squares Problems
    3. Standard Eigenvalue and Singular Value Problems
    4. Generalized Symmetric Definite Eigenproblems (GSEP)
  - Computational Routines
    1. Linear Equations
    2. Orthogonal Factorizations and Linear Least Squares Problems
    3. Generalized Orthogonal Factorizations
    4. Symmetric Eigenproblems
    5. Nonsymmetric Eigenproblems
    6. Singular Value Decomposition
    7. Generalized Symmetric Definite Eigenproblems

Driver Routines for Standard Eigenvalue and Singular Value Problems

Standard Eigenvalue and Singular Value Problems

- Simmetric Matrix.
  - Simple: `w,z,info= pvsyev(a[,jobz='N',uplo='U'])`
  - Advanced: `m,nz,w,z,ifail,iclustr,gap,info=pvsyevx(a [,jobz='N',range='A',uplo='U',orfac=0,rcond=0, vl=None,vu=None,il=None,iu=None, abstol=0])`
- General Matrix.
  - Simple: `s,u,vt,info= pvgesvd(a[,jobu='N',jobvt='N'])`

Singular value decomposition Example with pvgesvd

$$A = U\Sigma V^T, \ A \in \mathbb{R}^{m\times n}$$

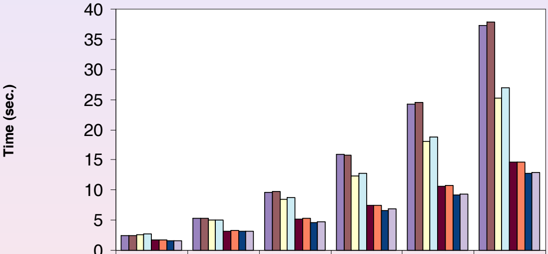
```
from PyACTS import *
import PyACTS.PyScaLAPACK as PySLK
from Numeric import *
n=8
PyACTS.gridinit()
if PyACTS.iread==1:
    a=8*identity(n,Float)
    print "a=",a
else:
    a=None
ACTS_lib=1
a=Num2PyACTS(a,ACTS_lib)
s,u,vt,info=PySLK.pvgesvd(a,'V','V')
u_num=PyACTS2Num(u)
vt_num=PyACTS2Num(vt)
if PyACTS.iread==1:
    print "s'=",transpose(s)
    print "u=",u_num
    print "vt=",vt_num
PyACTS.gridexit()
```

```
$ mpirun -np 4 mpipython pvgesvd.py
a= [[ 8.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  8.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  8.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  8.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  8.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  8.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  8.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  8.]]
s'= [ 8.  8.  8.  8.  8.  8.  8.  8.]
u= [[ 1.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  1.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  1.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  1.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  1.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  1.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  1.]]
vt= [[ 1.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  1.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  1.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  1.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  1.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  1.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  1.]]
Info: 0
```

SVD Execution times

Comparative Fortran vs Python: `pvgesvd` vs `PDGESVD`

Cluster-umh | Seaborg



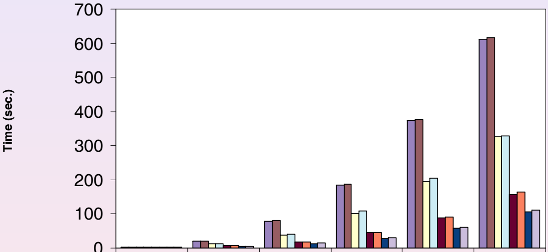
	200	400	600	800	1000	1200
FORTRAN 1X1	2,50	5,38	9,61	15,95	24,17	37,26
PYTHON 1X1	2,46	5,26	9,80	15,78	24,47	37,90
FORTRAN 2X1	2,59	5,04	8,51	12,36	18,05	25,20
PYTHON 2X1	2,67	5,05	8,71	12,71	18,76	26,94
FORTRAN 2X2	1,65	3,09	5,16	7,44	10,59	14,56
PYTHON 2X2	1,77	3,24	5,28	7,51	10,73	14,60
FORTRAN 3X2	1,55	3,12	4,53	6,63	9,14	12,79
PYTHON 3X2	1,57	3,21	4,75	6,82	9,38	12,90

Matrix size

SVD Execution times

Comparative Fortran vs Python: `pvgesvd` vs `PDGESVD`

Cluster-umh | Seaborg



	500	1000	1500	2000	2500	3000
FORTRAN 1X1	2,27	20,05	79,06	184,42	373,37	612,16
PYTHON 1X1	2,42	20,49	80,79	186,85	377,11	616,25
FORTRAN 2X1	1,98	11,69	38,80	102,33	194,60	325,91
PYTHON 2X1	1,95	13,43	40,65	109,44	203,58	328,24
FORTRAN 2X2	1,39	6,86	17,91	44,56	89,53	156,17
PYTHON 2X2	1,46	6,53	17,77	46,64	90,64	164,33
FORTRAN 3X2	1,34	4,77	13,76	28,77	57,48	106,95
PYTHON 3X2	1,34	5,00	13,97	29,23	59,73	111,87

Matrix size

Outline

Remembering scalability problems in EOFs Analysis

¿Scalability?

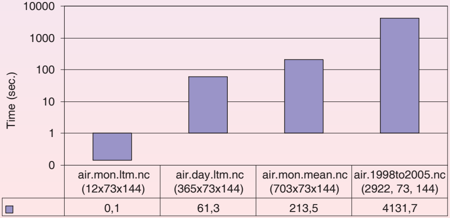
- NetCDF files obtained from Climate Diagnostics Center (`www.cdc.noaa.gov`).
- Extending data grid 144×73.
- Including Data grids at different elevations $17 \times 144 \times 73$.
- Increasing time axis.
  - `air.mon.ltm.nc`→ $17 \times 12 \times 144 \times 73$ 8,5MB
  - `air.day.ltm.nc`→ $17 \times 365 \times 144 \times 73$ 1,3GB
  - `air.mon.mean.nc`→ $17 \times 703 \times 144 \times 73$ 2,5GB
  - `air.1988to2005.nc`→ $17 \times 2922 \times 144 \times 73$ 4,17GB

	air.mon.ltm.nc (12x73x144)	air.day.ltm.nc (365x73x144)	air.mon.mean.nc (703x73x144)	air.1998to2005.nc (2922, 73, 144)
■	0,1	61,3	213,5	4131,7

Remembering scalability problems in EOFs Analysis

¿Scalability?

- NetCDF files obtained from Climate Diagnostics Center (`www.cdc.noaa.gov`).
- Extending data grid 144×73.
- Including Data grids at different elevations $17 \times 144 \times 73$.
- Increasing time axis.
  - `air.mon.ltm.nc`$\rightarrow 17 \times 12 \times 144 \times 73$ 8,5MB
  - `air.day.ltm.nc`$\rightarrow 17 \times 365 \times 144 \times 73$ 1,3GB
  - `air.mon.mean.nc`$\rightarrow 17 \times 703 \times 144 \times 73$ 2,5GB
  - `air.1988to2005.nc`$\rightarrow 17 \times 2922 \times 144 \times 73$ 4,17GB



An scalable solution also requires a parallel access to disk and data distribution

Introduction to netCDF

netCDF: network Common Data Format

- Popular package for storing data files in scientific applications.
- NetCDF consists of both an API and a portable file format.
- NetCDF file format guarantees data portability and self defined.
- API libraries for C/C++/F77/F90.
- Support from Unidata.
- Many organizations rely on the netCDF data access standard for data storage.
  - Easy to learn.
  - Widely used in climate community .

Data structure and file format

Data structure

- Dimensions.
- Variables.
- Attributes.

```
ejemplo_netCDF {
dimensions:
     lat = 5, lon = 10;
     level = 4, time = unlimited;
variables:
     float temp(time,level,lat,lon);
     temp:long_name = "Temperature";
     temp:units = "º Celsius";
     float rh(time,lat,lon);
          rh:long_name = "Relative Hum.";
          rh:valid_range = 0.0, 1.0;
// Global Attributes:
     :source = "Output Model";
data:
temp = 3.2, ... , 15.1,
            ...
       1.6, ... , 2.4;
rh = .5,.2,.4,.2,.3,.2,.4,.5,.6,.7,
            ...
      0,.1,.2,.4,.4,.4,.4,.7,.9,.9;
}
```

Data structure and file format

Data structure

- Dimensions.
- Variables.
- Attributes.

```
ejemplo_netCDF {
dimensions:
     lat = 5, lon = 10;
     level = 4, time = unlimited;
variables:
     float temp(time,level,lat,lon);
     temp:long_name = "Temperature";
     temp:units = "o Celsius";
     float rh(time,lat,lon);
          rh:long_name = "Relative Hum.";
          rh:valid_range = 0.0, 1.0;
// Global Attributes:
     :source = "Output Model";
data:
temp = 3.2, ... , 15.1,
            ...
       1.6, ... , 2.4;
rh = .5,.2,.4,.2,.3,.2,.4,.5,.6,.7,
            ...
      0,.1,.2,.4,.4,.4,.4,.7,.9,.9;
}
```

Data structure

- Dimensions.
- Variables.
- Attributes.

```
ejemplo_netCDF {
dimensions:
     lat = 5, lon = 10;
     level = 4, time = unlimited;
variables:
     float temp(time,level,lat,lon);
     temp:long_name = "Temperature";
     temp:units = "º Celsius";
     float rh(time,lat,lon);
          rh:long_name = "Relative Hum.";
          rh:valid_range = 0.0, 1.0;
// Global Attributes:
     :source = "Output Model";
data:
temp = 3.2, ... , 15.1,
            ...
       1.6, ... , 2.4;
rh = .5,.2,.4,.2,.3,.2,.4,.5,.6,.7,
            ...
      0,.1,.2,.4,.4,.4,.4,.7,.9,.9;
}
```

NetCDF access from Python

ScientificPython Package

- `Scientific.IO.NetCDF.`
- NetCDF metadata is managed with:
  - `NetCDFFile`
  - `NetCDFVariable`

NetCDF access from Python

ScientificPython Package

- `Scientific.IO.NetCDF`.
- NetCDF metadata is managed with:
  - `NetCDFFile`
  - `NetCDFVariable`

```
import sys, string
from Numeric import *
from Scientific.IO.NetCDF import NetCDFFile
file = NetCDFFile('test.nc', 'w')
file.title = "Example file"
file.version = 42
file.createDimension('xyz', 3)
file.createDimension('n', 20)
file.createDimension('t', None)
foo = file.createVariable('foo',
                 Float, ('n', 'xyz'))
foo.units = "arbitrary"
foo[:,:] = 1.
foo[0:3,:] = [42., 42., 42.]
foo[:,1] = 4.
foo[0,0] = 27.
file.close()
file2 = NetCDFFile('test.nc', 'r')
print file2.variables.keys()
print file2.dimensions.keys()
for varname in file2.variables.keys():
  var1 = file2.variables[varname]
  print varname,":",var1.shape
  foo = file2.variables['foo']
  data1 = var1.getValue()
  print "Datos:",data1
file2.close()
```

NetCDF access from Python

ScientificPython Package

- `Scientific.IO.NetCDF.`
- NetCDF metadata is managed with:
  - `NetCDFFile`
  - `NetCDFVariable`

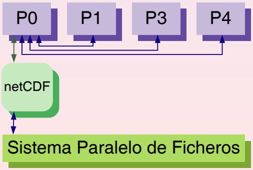
```
$ python read_netcdf.py test.nc
['foo']
['xyz', 't', 'n']
foo : (20, 3)
Datos: [[ 27.   4.  42.]
 [ 42.   4.  42.]
 [ 42.   4.  42.]
 [  1.   4.   1.]
 [  1.   4.   1.]
 [  1.   4.   1.]
 [  1.   4.   1.]
 [  1.   4.   1.]
 [  1.   4.   1.]
 [  1.   4.   1.]
 [  1.   4.   1.]
 ...
 [  1.   4.   1.]]
```

```
import sys, string
from Numeric import *
from Scientific.IO.NetCDF import NetCDFFile
file = NetCDFFile('test.nc', 'w')
file.title = "Example file"
file.version = 42
file.createDimension('xyz', 3)
file.createDimension('n', 20)
file.createDimension('t', None)
foo = file.createVariable('foo',
                  Float, ('n', 'xyz'))
foo.units = "arbitrary"
foo[:,:] = 1.
foo[0:3,:] = [42., 42., 42.]
foo[:,1] = 4.
foo[0,0] = 27.
file.close()
file2 = NetCDFFile('test.nc', 'r')
print file2.variables.keys()
print file2.dimensions.keys()
for varname in file2.variables.keys():
  var1 = file2.variables[varname]
  print varname,":",var1.shape
  foo = file2.variables['foo']
  data1 = var1.getValue()
  print "Datos:",data1
file2.close()
```

netCDF files management in parallel architectures

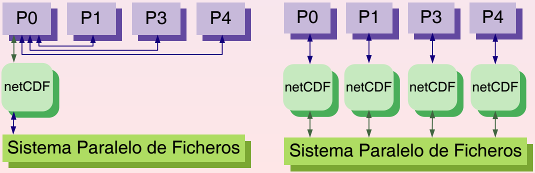
Parallel access to netCDF files

netCDF files management in parallel architectures



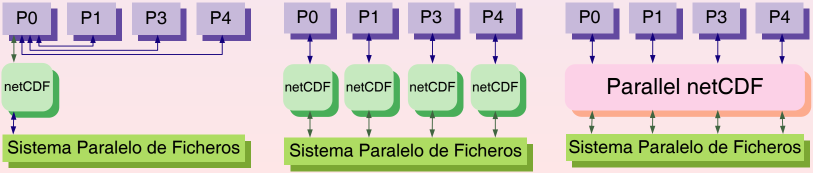
Parallel access to netCDF files

netCDF files management in parallel architectures



Parallel access to netCDF files

netCDF files management in parallel architectures

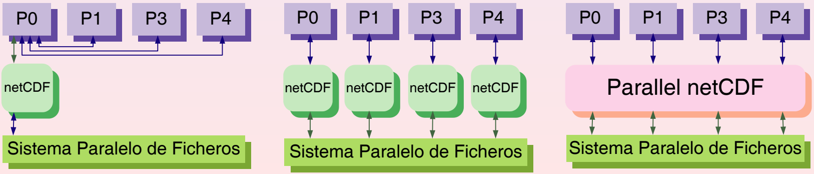


Parallel access to netCDF files

PnetCDF library

- Parallel interface to netCDF files.
- Call MPI-IO routines.
- Similar sintaxis than netCDF.
- Very similar to the sequential version netCDF.
- Interfaces for Fortran (`nfmpi`) y C (`ncmpi`).

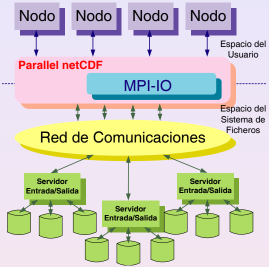
netCDF files management in parallel architectures



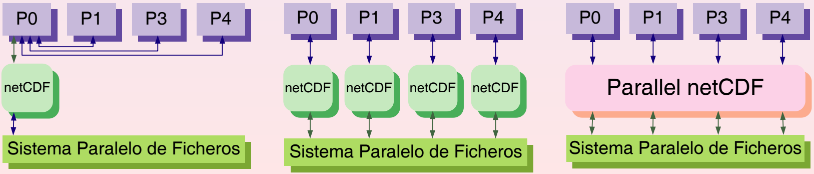
Parallel access to netCDF files

PnetCDF library

- Parallel interface to netCDF files.
- Call MPI-IO routines.
- Similar sintaxis than netCDF.
- Very similar to the sequential version netCDF.
- Interfaces for Fortran (`nfmpi`) y C (`ncmpi`).



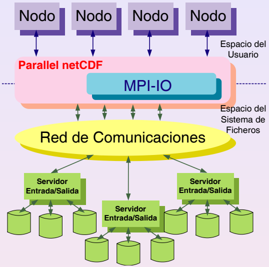
netCDF files management in parallel architectures



Parallel access to netCDF files

PnetCDF library

- Parallel interface to netCDF files.
- Call MPI-IO routines.
- Similar sintaxis than netCDF.
- Very similar to the sequential version netCDF.
- Interfaces for Fortran (`nfmpi`) y C (`ncmpi`).



Write	***Read***
`ncmpi_create(mpi_comm,filename,0, mpi_info,&file_id)`	`ncmpi_open(mpi_comm,filename,0, mpi_info, &file_id)`
`ncmpi_def_var(file_id, ...)`	`ncmpi_inq(file_id, ... );`
`ncmpi_enddef(file_id)`	`ncmpi_enddef(file_id)`
`ncmpi_put_vara_all(file_id,var_id,start[], count[],buffer,bufcount,mpi_datatype)`	`ncmpi_get_vars_all(file_id,var_id, buffer,bufcount,start[],count[], stride[],mpi_datatype)`
`ncmpi_close(file_id)`	

Motivation

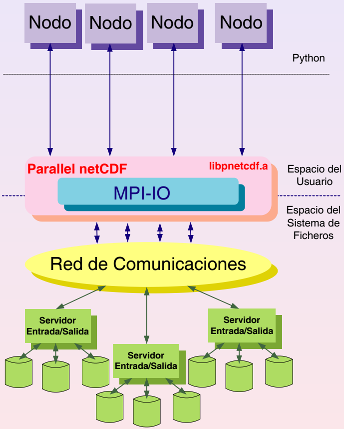
- NetCDF access from Python, only sequential .
- Limited in parallel aplications with Python
- ... but PnetCDF library is ...
  - Free distribution and Open Source.
  - Interfaces for C/C++/F77/F90.
- Create an interface for access to `libpnetcdf.a` from Python.
- Python Module hide parallelism and simplify synchronizing tasks.
- Functionality and sintaxis similar to ScientificPython.
- Object oriented programming:
  - `PNetCDFFile`
  - `NetCDFVariable`
- Integration with PyACTS.

Motivation

- NetCDF access from Python, only sequential .
- Limited in parallel aplications with Python
- ... but PnetCDF library is ...
  - Free distribution and Open Source.
  - Interfaces for C/C++/F77/F90.
- Create an interface for access to `libpnetcdf.a` from Python.
- Python Module hide parallelism and simplify synchronizing tasks.
- Functionality and sintaxis similar to ScientificPython.
- Object oriented programming:
  - `PNetCDFFile`
  - `NetCDFVariable`
- Integration with PyACTS.

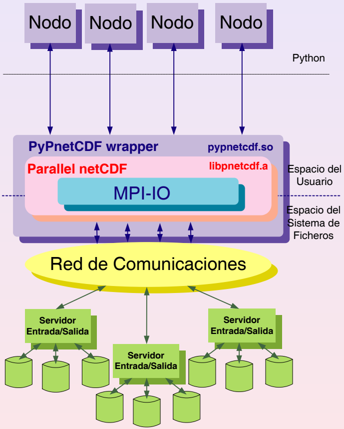
Motivation

- NetCDF access from Python, only sequential .
- Limited in parallel aplications with Python
- ... but PnetCDF library is ...
  - Free distribution and Open Source.
  - Interfaces for C/C++/F77/F90.
- Create an interface for access to `libpnetcdf.a` from Python.
- Python Module hide parallelism and simplify synchronizing tasks.
- Functionality and sintaxis similar to ScientificPython.
- Object oriented programming:
  - `PNetCDFFile`
  - `NetCDFVariable`
- Integration with PyACTS.



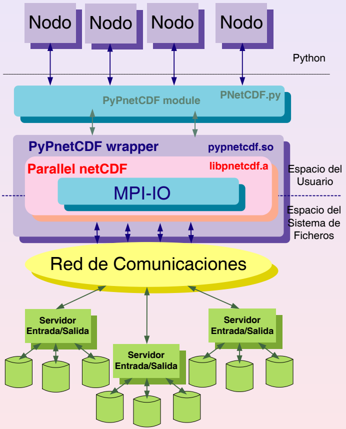
Motivation

- NetCDF access from Python, only sequential .
- Limited in parallel aplications with Python
- ... but PnetCDF library is ...
  - Free distribution and Open Source.
  - Interfaces for C/C++/F77/F90.
- Create an interface for access to `libpnetcdf.a` from Python.
- Python Module hide parallelism and simplify synchronizing tasks.
- Functionality and sintaxis similar to ScientificPython.
- Object oriented programming:
  - `PNetCDFFile`
  - `NetCDFVariable`
- Integration with PyACTS.



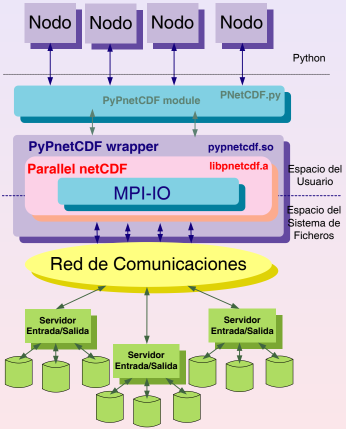
Motivation

- NetCDF access from Python, only sequential .
- Limited in parallel aplications with Python
- ... but PnetCDF library is ...
  - Free distribution and Open Source.
  - Interfaces for C/C++/F77/F90.
- Create an interface for access to `libpnetcdf.a` from Python.
- Python Module hide parallelism and simplify synchronizing tasks.
- Functionality and sintaxis similar to ScientificPython.
- Object oriented programming:
  - `PNetCDFFile`
  - `NetCDFVariable`
- Integration with PyACTS.



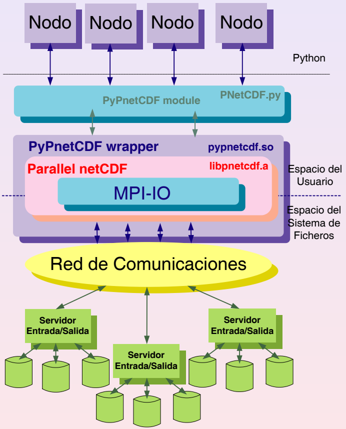
Motivation

- NetCDF access from Python, only sequential .
- Limited in parallel aplications with Python
- ... but PnetCDF library is ...
  - Free distribution and Open Source.
  - Interfaces for C/C++/F77/F90.
- Create an interface for access to `libpnetcdf.a` from Python.
- Python Module hide parallelism and simplify synchronizing tasks.
- Functionality and sintaxis similar to ScientificPython.
- Object oriented programming:
  - `PNetCDFFile`
  - `NetCDFVariable`
- Integration with PyACTS.



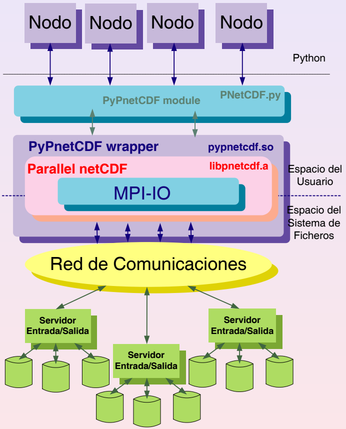
Motivation

- NetCDF access from Python, only sequential .
- Limited in parallel aplications with Python
- ... but PnetCDF library is ...
  - Free distribution and Open Source.
  - Interfaces for C/C++/F77/F90.
- Create an interface for access to `libpnetcdf.a` from Python.
- Python Module hide parallelism and simplify synchronizing tasks.
- Functionality and sintaxis similar to ScientificPython.
- Object oriented programming:
  - `PNetCDFFile`
  - `NetCDFVariable`
- Integration with PyACTS.



Motivation

- NetCDF access from Python, only sequential .
- Limited in parallel aplications with Python
- ... but PnetCDF library is ...
  - Free distribution and Open Source.
  - Interfaces for C/C++/F77/F90.
- Create an interface for access to `libpnetcdf.a` from Python.
- Python Module hide parallelism and simplify synchronizing tasks.
- Functionality and sintaxis similar to ScientificPython.
- Object oriented programming:
  - `PNetCDFFile`
  - `NetCDFVariable`
- Integration with PyACTS.



Parallel access from Python using PyPnetCDF

```
from Numeric import *
from PyPNetCDF.PNetCDF import *
import PyACTS

file = PNetCDFFile('test.nc', 'w')
file.title = "Just some useless junk"
file.version = 42

file.createDimension('xyz', 3)
file.createDimension('n', 20)
# Dimension ilimitada
file.createDimension('t', None)
dfoo=('n','xyz')
foo=file.createVariable('foo',Float,dfoo)
foo.units = "arbitrary"

foo[:,:] = 1.
foo[0:3,:] = [42., 42., 42.]
foo[:,1] = 4.
foo.data[0,0] = PyACTS.iam
file.enddef()
foo.setValue()
file.close()
```

```
file2 = PNetCDFFile('test.nc', 'r')
print "*"*10," Proceso ",PyACTS.iam,"/",
      PyACTS.nprocs,"*"*10
print ncfile1.variables.keys()
print ncfile1.dimensions.keys()
for varname in file2.variables.keys():
   var1 = file2.variables[varname]
   print varname,":",var1.shape,";",
         var1.units
   foo = file2.variables['foo']
   data1 = var1.getValue()
   print "Datos:",data1
file2.close()
```

Parallel access from Python using PyPnetCDF

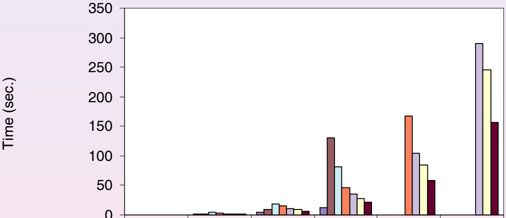

```
$mpirun -np 4 mpipython read_pnetcdf.py ./test.nc
```

```
 **********  Proceso  0 / 4 **********
['foo']
['xyz', 't', 'n']
foo : (20, 3) ; arbitrary
Datos: [[ 0.   4.  42.]
 [ 42.   4.  42.]
 [ 42.   4.  42.]
 [  1.   4.   1.]
 [  1.   4.   1.]]
**********  Proceso  3 / 4 **********
['foo']
['xyz', 't', 'n']
foo : (20, 3) ; arbitrary
Datos: [[ 3.   4.   1.]
 [  1.   4.   1.]
 [  1.   4.   1.]
 [  1.   4.   1.]
 [  1.   4.   1.]]
```

```
**********  Proceso  2 / 4 **********
['foo']
['xyz', 't', 'n']
foo : (20, 3) ; arbitrary
Datos: [[ 2.   4.   1.]
 [  1.   4.   1.]
 [  1.   4.   1.]
 [  1.   4.   1.]
 [  1.   4.   1.]]
**********  Proceso  1 /4 **********
['foo']
['xyz', 't', 'n']
foo : (20, 3) ; arbitrary
Datos: [[ 1.   4.   1.]
 [  1.   4.   1.]
 [  1.   4.   1.]
 [  1.   4.   1.]
 [  1.   4.   1.]]
```

PyPnetCDF Scalability

Reading Times in Cluster-umh

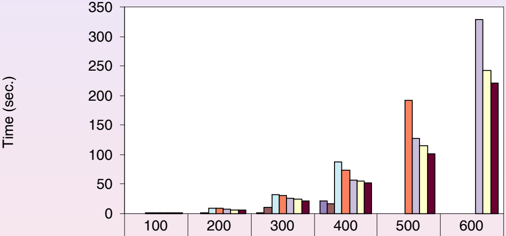


	100	200	300	400	500	600
ScientificPython	0,18	1,37	4,65	11,67		
PyPnetCDF 1p	0,26	1,95	8,89	130,21		
PyPnetCDF 2p	0,14	4,31	17,98	82,12		
PyPnetCDF 3p	0,29	3,30	15,11	45,93	167,11	
PyPnetCDF 4p	0,07	2,22	11,22	34,96	104,40	289,87
PyPnetCDF 5p	0,06	2,25	8,56	27,66	83,82	245,78
PyPnetCDF 6p	0,12	1,61	6,90	21,38	58,57	156,83

Matrix size (NxNxN)

PyPnetCDF Scalability

Writing Times in Cluster-umh



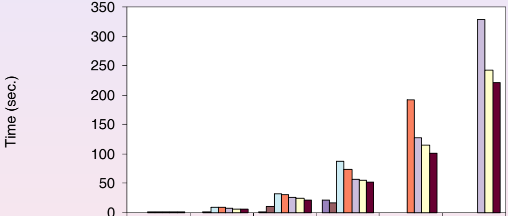
	100	200	300	400	500	600
ScientificPython	0,03	0,25	0,93	21,61		
PyPnetCDF 1p	0,27	2,08	10,20	16,28		
PyPnetCDF 2p	1,36	9,38	32,12	87,71		
PyPnetCDF 3p	1,28	9,50	31,35	73,68	191,96	
PyPnetCDF 4p	1,25	8,43	25,34	57,54	127,05	328,43
PyPnetCDF 5p	1,18	6,32	23,87	54,84	114,56	242,10
PyPnetCDF 6p	1,09	6,61	21,34	51,74	101,94	220,99

Matrix size (NxNxN)

PyPnetCDF Scalability

PyPnetCDF scales with the number of processors

Writing Times in Cluster-umh



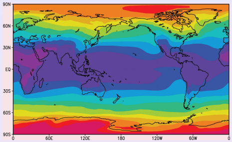
	100	200	300	400	500	600
ScientificPython	0,03	0,25	0,93	21,61		
PyPnetCDF 1p	0,27	2,08	10,20	16,28		
PyPnetCDF 2p	1,36	9,38	32,12	87,71		
PyPnetCDF 3p	1,28	9,50	31,35	73,68	191,96	
PyPnetCDF 4p	1,25	8,43	25,34	57,54	127,05	328,43
PyPnetCDF 5p	1,18	6,32	23,87	54,84	114,56	242,10
PyPnetCDF 6p	1,09	6,61	21,34	51,74	101,94	220,99

Matrix size (NxNxN)

Outline

EOFs Analysis with PyClimate

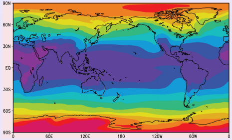
- Delete redundant information with minimun lost of variability.

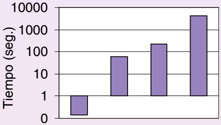


Remembering EOFs Analysis

EOFs Analysis with PyClimate

- Delete redundant information with minimun lost of variability.
- Scalability problem with huge data volume.

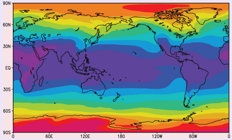


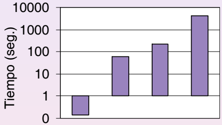


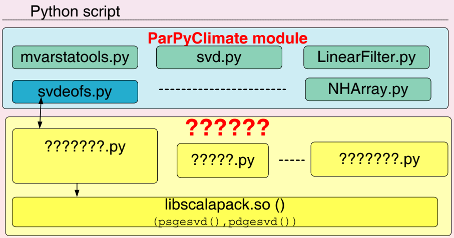
Remembering EOFs Analysis

EOFs Analysis with PyClimate

- Delete redundant information with minimun lost of variability.
- Scalability problem with huge data volume.
- We didn't have appropriate Python tools.



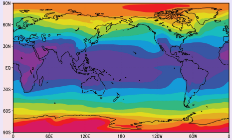


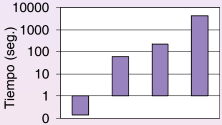


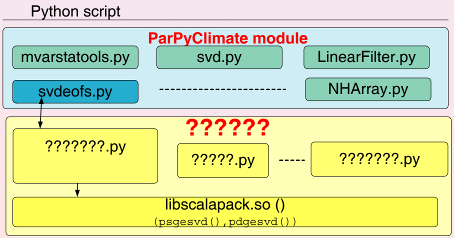
Remembering EOFs Analysis

EOFs Analysis with PyClimate

- Delete redundant information with minimun lost of variability.
- Scalability problem with huge data volume.
- We didn't have appropriate Python tools.
- But with PyACTS and PyPnetCDF we can build and scalable EOFs analisys.





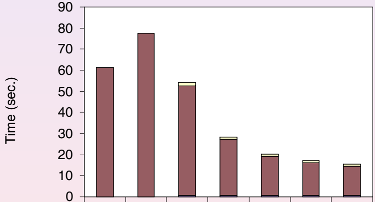


Scalability with *ParPyClimate*

air.day.ltm.nc | air.mon.mean.nc | air.1988to2005.nc

$17 \times 365 \times 144 \times 73$, 1.3GB

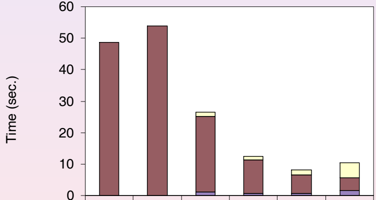
Cluster



	Sec.	Par.1	Par.2	Par.3	Par.4	Par.5	Par.6
Writing		0,00	1,62	1,06	1,01	1,03	1,03
svdeofs	61,31	77,47	51,96	26,48	18,36	15,35	13,56
Reading		0,00	0,79	0,73	0,70	0,68	0,79

Number of processors

Seaborg



	Sec.	Par.1	Par.2	Par.4	Par.8	Par.16
Writing		0,00	1,38	1,11	1,73	4,71
svdeofs	48,61	53,97	23,95	10,69	5,75	4,08
Reading		0,00	1,16	0,65	0,73	1,56

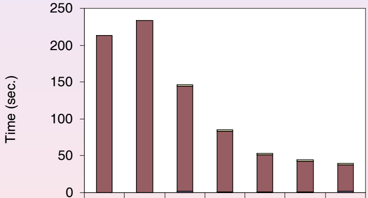
Number of processors

Scalability with *ParPyClimate*

`air.day.ltm.nc` | `air.mon.mean.nc` | `air.1988to2005.nc`

$17 \times 703 \times 144 \times 73$, 2.5GB

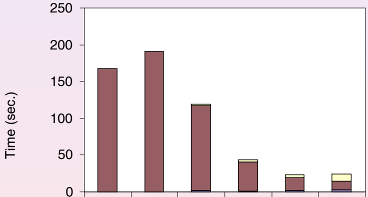
Cluster



	Sec.	Par.1	Par.2	Par.3	Par.4	Par.5	Par.6
Writing		0,03	2,40	2,06	2,05	2,04	2,04
svdeofs	213,5	233,0	142,3	81,46	49,63	41,36	35,97
Reading		0,05	1,60	1,42	1,37	1,32	1,55

Number of processors

Seaborg



	Sec.	Par.1	Par.2	Par.4	Par.8	Par.16
Recolección		0,08	2,17	2,27	3,53	9,40
svdeofs	167,64	190,43	115,53	39,61	17,87	11,39
Distribución		0,07	1,86	1,32	1,49	3,08

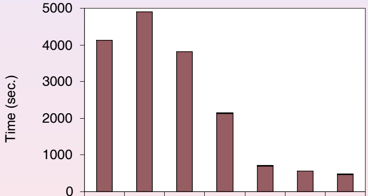
Number of processors

Scalability with *ParPyClimate*

`air.day.ltm.nc` `air.mon.mean.nc` `air.1988to2005.nc`

17 $\times$ 2922 $\times$ 144 $\times$ 73, 4.17GB

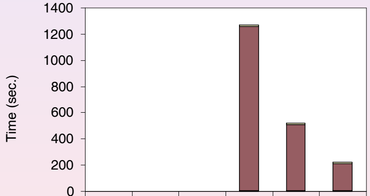
Cluster



	Sec.	Par.1	Par.2	Par.3	Par.4	Par.5	Par.6
Writing		0,12	10,49	10,74	10,45	10,54	10,58
svdeofs	4131,	4912,	3805,	2123,	693,4	552,6	458,5
Reading		0,10	8,51	8,13	6,06	6,83	6,47

Number of processors

Seaborg



	Sec.	Par.1	Par.2	Par.4	Par.8	Par.16
Writing				13,84	13,79	13,82
svdeofs				1251,7	503,59	205,76
Reading				6,61	5,23	4,67

Number of processors

Outline

Conclusions

Goals

- We can manage huge datasets in a parallel execution with a intuitive language.

Conclusions

Goals

- We can manage huge datasets in a parallel execution with a intuitive language.
- PyPnetCDF is integrated with PyACTS.

Conclusions

Goals

- We can manage huge datasets in a parallel execution with a intuitive language.
- PyPnetCDF is integrated with PyACTS.
- Simplified framework that let us manage huge data volume in a simple way.

Conclusions

Goals

- We can manage huge datasets in a parallel execution with a intuitive language.
- PyPnetCDF is integrated with PyACTS.
- Simplified framework that let us manage huge data volume in a simple way.
- Hide parallelism and data distribution if the users that don't want to know about it.

Conclusions

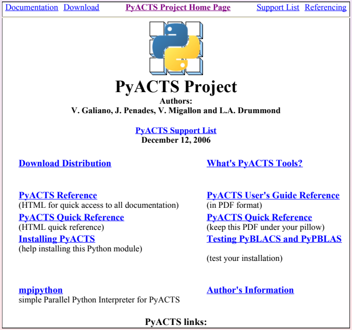
Goals

- We can manage huge datasets in a parallel execution with a intuitive language.
- PyPnetCDF is integrated with PyACTS.
- Simplified framework that let us manage huge data volume in a simple way.
- Hide parallelism and data distribution if the users that don't want to know about it.
- New applications, not only in climate area.

PyACTS and PyPnetCDF Distribution

- Web site.
- Mail List and mail support.
- Backup in sourceforge.net
- Documentation and quickguide in PDF & HTML.
- Referenced from Python and Unidata.

```
http://www.pyacts.org
http://www.pyacts.org/pypnetcdf
```



Documentation Download PyACTS Project Home Page Support List Referencing

PyACTS Project

Authors:
V. Galiano, J. Penades, V. Migallon and L.A. Drummond

PyACTS Support List
December 12, 2006

Download Distribution

What's PyACTS Tools?

PyACTS Reference
(HTML for quick access to all documentation)

PyACTS User's Guide Reference
(in PDF format)

PyACTS Quick Reference
(HTML quick reference)

PyACTS Quick Reference
(keep this PDF under your pillow)

Installing PyACTS
(help installing this Python module)

Testing PyBLACS and PyPBLAS
(test your installation)

mpipython
simple Parallel Python Interpreter for PyACTS

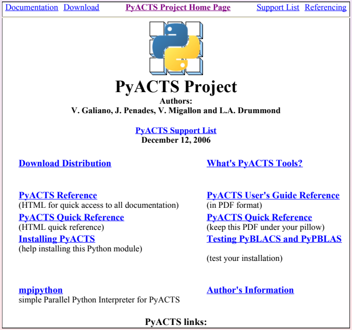
Author's Information

PyACTS links:

PyACTS and PyPnetCDF Distribution

- Web site.
- Mail List and mail support.
- Backup in sourceforge.net
- Documentation and quickguide in PDF & HTML.
- Referenced from Python and Unidata.

```
http://www.pyacts.org
http://www.pyacts.org/pypnetcdf
```



Documentation Download PyACTS Project Home Page Support List Referencing

PyACTS Project

Authors:
V. Galiano, J. Penades, V. Migallon and L.A. Drummond

PyACTS Support List
December 12, 2006

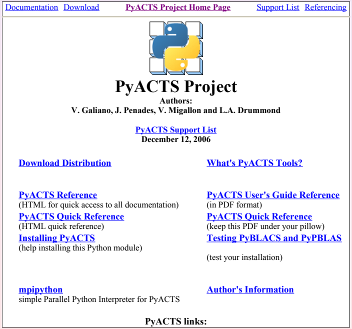
Download Distribution	**What's PyACTS Tools?**
PyACTS Reference (HTML for quick access to all documentation)	**PyACTS User's Guide Reference** (in PDF format)
PyACTS Quick Reference (HTML quick reference)	**PyACTS Quick Reference** (keep this PDF under your pillow)
Installing PyACTS (help installing this Python module)	**Testing PyBLACS and PyPBLAS** (test your installation)
mpipython simple Parallel Python Interpreter for PyACTS	**Author's Information**

PyACTS links:

PyACTS and PyPnetCDF Distribution

- Web site.
- Mail List and mail support.
- Backup in sourceforge.net
- Documentation and quickguide in PDF & HTML.
- Referenced from Python and Unidata.

http://sourceforge.net/projects/pyacts
http://sourceforge.net/projects/pypnetcdf



Documentation Download PyACTS Project Home Page Support List Referencing

PyACTS Project

Authors:
V. Galiano, J. Penades, V. Migallon and L.A. Drummond

PyACTS Support List
December 12, 2006

Download Distribution

What's PyACTS Tools?

PyACTS Reference
(HTML for quick access to all documentation)
PyACTS Quick Reference
(HTML quick reference)
Installing PyACTS
(help installing this Python module)

PyACTS User's Guide Reference
(in PDF format)
PyACTS Quick Reference
(keep this PDF under your pillow)
Testing PyBLACS and PyPBLAS
(test your installation)

mpipython
simple Parallel Python Interpreter for PyACTS

Author's Information

PyACTS links:

PyACTS and PyPnetCDF Distribution

- Web site.
- Mail List and mail support.
- Backup in sourceforge.net
- Documentation and quickguide in PDF & HTML.
- Referenced from Python and Unidata.

Manual de Referencia de PyACTS:
PyBLACS, PyPBLAS
y PyScaLAPACK

Release 1.0

Vicente Galiano
Violeta Migallón
Jose Penadés
Tony Drummond

PyACTS and PyPnetCDF Distribution

- Web site.
- Mail List and mail support.
- Backup in sourceforge.net
- Documentation and quickguide in PDF & HTML.
- Referenced from Python and Unidata.

Manual de Referencia de PyACTS:
PyBLACS, PyPBLAS
y PyScaLAPACK

Release 1.0

Vicente Galiano
Violeta Migallón
Jose Penadés
Tony Drummond

And of course, ...

DÊKUJI

DANKE

THANKS

GRACIAS