PYTABLES & Family
Analyzing and Sharing HDF5 Data with Python

Francesc Altet
Cáribos Coop. V.

Who are we?

- Cárbabos is the company committed to the PYTABLES suite development and deployment.

- We have years of experience in designing software solutions for handling extremely large datasets.

- What we provide:
  - Commercial support for the PYTABLES suite.
  - PYTABLES-based applications.
  - Consulting services for managing complex data environments.
Who are we?

- Cárabos is the company committed to the PYTABLES suite development and deployment.

- We have years of experience in designing software solutions for handling extremely large datasets.

What we provide:

- Commercial support for the PYTABLES suite.
- PYTABLES-based applications.
- Consulting services for managing complex data environments.
Who are we?

- Cárbabos is the company committed to the PYTABLES suite development and deployment.

- We have years of experience in designing software solutions for handling extremely large datasets.

- What we provide:
  - Commercial support for the PYTABLES suite.
  - PYTABLES-based applications.
  - Consulting services for managing complex data environments.
Outline

1. An Introduction to the Python Language
   - Facts about Python
   - Scientific Packages
   - An example

2. PYTABLES
   - Overview
   - Usage Examples
   - PYTABLES Pro

3. CSTABLES
   - Overview
   - Design goals
   - Examples of Use
Outline

1. An Introduction to the Python Language
   - Facts about Python
   - Scientific Packages
   - An example

2. PYTABLES
   - Overview
   - Usage Examples
   - PYTABLES Pro

3. CSTABLES
   - Overview
   - Design goals
   - Examples of Use
Main Features

- Interpreted $\Rightarrow$ Allows interactivity
- Flexible data structures $\Rightarrow$ Malleability
- Minimalistic grammar $\Rightarrow$ Easy to learn
- Very complete library (*Batteries included*) $\Rightarrow$ Boosts productivity
Main Features

- Interpreted ⇒ Allows interactivity
- Flexible data structures ⇒ Malleability
- Minimalistic grammar ⇒ Easy to learn
- Very complete library (Batteries included) ⇒ Boosts productivity
Main Features

- Interpreted ⇒ Allows interactivity
- Flexible data structures ⇒ Malleability
- Minimalistic grammar ⇒ Easy to learn
- Very complete library (*Batteries included*) ⇒ Boosts productivity
Main Features

- Interpreted ⇒ Allows interactivity
- Flexible data structures ⇒ Malleability
- Minimalistic grammar ⇒ Easy to learn
- Very complete library (*Batteries included*) ⇒ Boosts productivity
Very Suitable for Scientific/Engineer Fields

- Interactivity has always been very appreciated for increasing productivity.
- Being interpreted does not necessarily mean being inefficient. Python has solutions for linking C & Fortran libraries in an easy way.
- Programming is normally considered a necessary evil. Rich expressivity of Python normally reduces the amount of code to solve real problems.
- Many efficient scientific libraries have been developed for Python: Numeric, numarray, SciPy, Scientific-Python, matplotlib...
Very Suitable for Scientific/Engineer Fields

- Interactivity has always been very appreciated for increasing productivity.
- Being interpreted does not necessarily mean being inefficient. Python has solutions for linking C & Fortran libraries in an easy way.
- Programming is normally considered a necessary evil. Rich expressivity of Python normally reduces the amount of code to solve real problems.
- Many efficient scientific libraries have been developed for Python: Numeric, numarray, SciPy, Scientific-Python, matplotlib...
Very Suitable for Scientific/Engineer Fields

- Interactivity has always been very appreciated for increasing productivity.
- Being interpreted does not necessarily mean being inefficient. Python has solutions for linking C & Fortran libraries in an easy way.
- Programming is normally considered a *necessary evil*. Rich expressivity of Python normally reduces the amount of code to solve real problems.
- Many efficient scientific libraries have been developed for Python: Numeric, numarray, SciPy, Scientific-Python, matplotlib...
Very Suitable for Scientific/Engineer Fields

- Interactivity has always been very appreciated for increasing productivity.
- Being interpreted does not necessarily mean being inefficient. Python has solutions for linking C & Fortran libraries in an easy way.
- Programming is normally considered a *necessary evil*. Rich expressivity of Python normally reduces the amount of code to solve real problems.
- Many efficient scientific libraries have been developed for Python: Numeric, numarray, SciPy, Scientific-Python, matplotlib...
Outline

1. An Introduction to the Python Language
   - Facts about Python
   - Scientific Packages
   - An example

2. PYTABLES
   - Overview
   - Usage Examples
   - PYTABLES Pro

3. CSTABLES
   - Overview
   - Design goals
   - Examples of Use
Basic Matrix Handling

**Numeric**
- Very mature but lacking some features.
- Still has a huge user base.

**scipy.core**
- Called to substitute both Numeric & numarray. When finished, it is supposed to have all the advantages of Numeric & numarray together.

**numarray**
- Created to overcome some of the limitations of Numeric.
- Better handling of large arrays, heterogeneous data...
Basic Matrix Handling

**Numeric**
- Very mature but lacking some features.
- Still has a huge user base.

**scipy.core**
- Called to substitute both Numeric & numarray. When finished, it is supposed to have all the advantages of Numeric & numarray together.

**numarray**
- Created to overcome some of the limitations of Numeric.
- Better handling of large arrays, heterogeneous data...
Basic Matrix Handling

**Numeric**
- Very mature but lacking some features.
- Still has a huge user base.

**scipy.core**
- Called to substitute both Numeric & numarray. When finished, it is supposed to have all the advantages of Numeric & numarray together.

**numarray**
- Created to overcome some of the limitations of Numeric.
- Better handling of large arrays, heterogeneous data...
Numerical Supplements

SciPy
- Optimization, integration, special functions
- Signal and image processing
- Genetic algorithms, ODE solvers...

Scientific Python
- Quaternions, automatic derivatives, (linear) interpolation, polynomials, ...
- Overlaps somewhat SciPy, but it has...
- A nice netCDF module for I/O
Numerical Supplements

SciPy
- Optimization, integration, special functions
- Signal and image processing
- Genetic algorithms, ODE solvers...

Scientific Python
- Quaternions, automatic derivatives, (linear) interpolation, polynomials, ...
- Overlaps somewhat SciPy, but it has...
- A nice netCDF module for I/O
Plotting (2D)

**matplotlib**
- Great interactivity.
- Produces publication quality figures.

[Graph showing Gaussian colored noise with impulse response and probability distribution]
Plotting (2D)

PyQwt

- Fast plotting.
- Nice to be embedded in Qt apps.
MayaVi

- 3D-oriented scientific data visualizer.
- Uses the Visualization Toolkit (VTK).

MayaVi example

Francesc Altet

PYTABLES & Family
Outline

1. An Introduction to the Python Language
   - Facts about Python
   - Scientific Packages
   - An example

2. PYTABLES
   - Overview
   - Usage Examples
   - PYTABLES Pro

3. CSTABLES
   - Overview
   - Design goals
   - Examples of Use
Evaluate $E_n(x) = \int_1^\infty \frac{e^{-xt}}{t^n} dt$ for each value of $n$

```python
from scipy import *
from scipy.integrate import quad, Inf

def integrand(t,n,x):
    return exp(-x*t) / t**n

def expint(n,x):
    return quad(integrand, 1, Inf, args=(n, x))[0]

v_expint = vectorize(expint)
print v_expint(3,arange(1.0,4.0,0.5))
```

The output

```
[ 0.10969197, 0.05673949, 0.03013338, 0.01629537,
  0.00893065, 0.00494538,]
```
### Programming as a Necessary Evil

The plotting code (matplotlib)

```python
xrng = arange(1.0, 4.0, 0.1)
plot(xrng, v_expint(1, xrng))
plot(xrng, v_expint(2, xrng))
plot(xrng, v_expint(3, xrng))
legend(('n=1', 'n=2', 'n=3'))
title('Exponential Integral')
```

---

The output

![Exponential Integral graph](image)
Resources

- Introductory material:
  - docs.python.org/tut/tut.html
  - www.python.org/doc/Intros.html


- Martelli & Ascher, "Python Cookbook": more specialized, useful recipes for particular problems.

- Hans P. Langtangen, "Python Scripting for Computational Science": teaches computational scientists and engineers how to write small Python scripts efficiently.
Outline

1. An Introduction to the Python Language
   - Facts about Python
   - Scientific Packages
   - An example

2. PyTables
   - Overview
   - Usage Examples
   - PyTables Pro

3. CSTables
   - Overview
   - Design goals
   - Examples of Use
Simply stated: a database for Python based on HDF5.

- Designed to deal with extremely large datasets.
- Provides an easy-to-use interface.
- Supports many of the features of HDF5 and others that are not present in it (e.g. indexation).
- It is Open Source software (BSD license).
Why HDF5?

I started looking for different backends for saving large datasets, but HDF5 was the final winner.

- Thought out for managing very large datasets in an efficient way.
- Let you organize datasets hierchically.
- Very flexible and well tested in scientific environments.
- Good maintenance and improvement rate.

- It is Open Source software.
I started looking for different backends for saving large datasets, but HDF5 was the final winner.

- Thought out for managing very large datasets in an efficient way.
- Let you organize datasets hierarchically.
- Very flexible and well tested in scientific environments.
- Good maintenance and improvement rate.
- It is Open Source software.
What Does *Extremely Large* Exactly Mean?

The Hitch Hiker’s Guide to the Galaxy offers this definition of the word “Infinite”:

**Infinite**: Bigger than the biggest thing ever and then some. Much bigger than that in fact, really amazingly immense, a totally stunning size, real “wow, that’s big”. Infinity is just so big that, by comparison, bigness looks really titchy. Gigantic multiplied by colossal multiplied by staggeringly huge is the sort of concept we’re trying to get across here.

**Disclaimer**

Agreed, ’Extremely Large’ may not exactly mean ’Infinite’, although it is pretty close to this definition.
What Does *Extremely Large* Exactly Mean?

The *Hitch Hiker’s Guide to the Galaxy* offers this definition of the word “Infinite”:

**Infinite:** Bigger than the biggest thing ever and then some. Much bigger than that in fact, really amazingly immense, a totally stunning size, real “wow, that’s big”. Infinity is just so big that, by comparison, bigness looks really titchy. Gigantic multiplied by colossal multiplied by staggeringly huge is the sort of concept we’re trying to get across here.

**Disclaimer**

Agreed, ’*Extremely Large*’ may not exactly mean ’*Infinite*’, although it is pretty close to this definition.
What Does *Extremely Large* Exactly Mean?

The *Hitch Hiker’s Guide to the Galaxy* offers this definition of the word “Infinite”:

**Infinite**: Bigger than the biggest thing ever and then some. Much bigger than that in fact, really amazingly immense, a totally stunning size, real “wow, that’s big”. Infinity is just so big that, by comparison, bigness looks really titchy. Gigantic multiplied by colossal multiplied by staggeringly huge is the sort of concept we’re trying to get across here.

**Disclaimer**

Agreed, ’*Extremely Large*’ may not exactly mean ’*Infinite*’, although it is pretty close to this definition.
Which *ELD* Features Does *PYTABLES* Support?

- A good base library (HDF5).
- Support for 64-bit in all data addressing.
  - Need to overcome a Python slicing limitation: only 32-bit addresses are supported.
- Buffered I/O.
- A LRU cache system for an efficient reuse of objects.
- Fast indexing and searching capabilities for tables.
Which *ELD* Features Does *PYTABLES* Support?

- A good base library (HDF5).
- Support for 64-bit in all data addressing.
  - Need to overcome a Python slicing limitation: only 32-bit addresses are supported.
- Buffered I/O.
- A LRU cache system for an efficient reuse of objects.
- Fast indexing and searching capabilities for tables.
Which *ELD* Features Does *PYTABLES* Support?

- A good base library (HDF5).
- Support for 64-bit in all data addressing.
  - Need to overcome a Python slicing limitation: only 32-bit addresses are supported.
- Buffered I/O.
- A LRU cache system for an efficient reuse of objects.
- Fast indexing and searching capabilities for tables.
Which *ELD* Features Does *PyTables* Support?

- A good base library (HDF5).
- Support for 64-bit in all data addressing.
  - Need to overcome a Python slicing limitation: only 32-bit addresses are supported.
- Buffered I/O.
- A LRU cache system for an efficient reuse of objects.
- Fast indexing and searching capabilities for tables.
Which *ELD* Features Does **PYTABLES** Support?

- A good base library (HDF5).
- Support for 64-bit in all data addressing.
  - Need to overcome a Python slicing limitation: only 32-bit addresses are supported.
- Buffered I/O.
- A LRU cache system for an efficient reuse of objects.
- Fast indexing and searching capabilities for tables.
Why Buffered I/O?

Because of speed (what else?):

![Graph showing insert time for table over number of rows]
Why a Cache System?

1.- To achieve better open file times:

PyTables Access Tree Performance (using MR)

- pytables 1.1.1 hdf5 1.6.4
- pytables 1.1.1 hdf5 1.7.51
- pytables 1.2 hdf5 1.6.4
- pytables 1.2 hdf5 1.7.51
Why a Cache System?

2.- To achieve a conservative usage of the memory:

![PyTables Access Tree Memory Usage (using MR) chart]

Why a Cache System? 2.- To achieve a conservative usage of the memory:

![PyTables Access Tree Memory Usage (using MR) chart]
Why Fast Indexing?

It is not trifling in terms of time when you have to index tables with more than one billion of rows:

![Graph showing creation index time for Float64 column]
Ease of Use

**Natural naming**

```python
# access to file:/group1/table
table = file.root.group1.table
```

**Support for generalized slicing**

```python
# step means a stride in the slice
table[start:stop:step]
```

**Support for iterators**

```python
# get the values in col1 that satisfy the
# (1.3 < col3 <= 2.) condition in table
col3 = table.cols.col3
[r['col1'] for r in table.where(1.3 < col3 <= 2.)]
```
Ease of Use

Natural naming

```python
# access to file:/group1/table
table = file.root.group1.table
```

Support for generalized slicing

```python
# step means a stride in the slice
table[start:stop:step]
```

Support for iterators

```python
# get the values in col11 that satisfy the
# (1.3 < col13 <= 2.) condition in table
col13 = table.cols.col13
[r['col11'] for r in table.where(1.3 < col13 <= 2.)]
```
Ease of Use

Natural naming

```python
# access to file:/group1/table
table = file.root.group1.table
```

Support for generalized slicing

```python
# step means a stride in the slice
table[start:stop:step]
```

Support for iterators

```python
# get the values in col1 that satisfy the
# (1.3 < col3 <= 2.) condition in table
col3 = table.cols.col3
[r['col1'] for r in table.where(1.3 < col3 <= 2.)]
```
Outline

1. An Introduction to the Python Language
   - Facts about Python
   - Scientific Packages
   - An example

2. PyTables
   - Overview
   - Usage Examples
   - PyTables Pro

3. CsTables
   - Overview
   - Design goals
   - Examples of Use
Supported Objects in PYTABLES

- **Table**: Lets you deal with heterogeneous datasets. Chunked. Enlargeable. Support for nested types.
- **Array**: Provides quick and dirty array handling. Not chunked. Non enlargeable.
- **CArray**: Provides compressed array support. Chunked. Not enlargeable.
- **EArray**: Most general array support. Chunked. Enlargeable.
- **VLArray**: Collections of homogeneous data with a variable number of entries. Chunked. Enlargeable.
- **Group**: The *structural* component.
What’s New in PYTABLES 1.2?

- Added a new cache for the object tree.
  - Allows a contained use of memory (even with huge trees).
  - Almost instantaneous opening of files (good news for interactive use!).

- New NetCDF module (contributed by Jeff Whitaker).
  - Provides API emulation for `Scientific.IO.NetCDF`.
  - `netCDF-3 ⇔ HDF5` conversions.
  - `tables.NetCDF` datasets can be shared over the internet with the OPeNDAP protocol.
  - Plans to write data natively in netCDF-4 format in the future.
What’s New in PYTABLES 1.2?

- Added a new cache for the object tree.
  - Allows a contained use of memory (even with huge trees).
  - Almost instantaneous opening of files (good news for interactive use!).

- New NetCDF module (contributed by Jeff Whitaker).
  - Provides API emulation for Scientific.IO.NetCDF.
  - netCDF-3 ⇔ HDF5 conversions.
  - tables.NetCDF datasets can be shared over the internet with the OPeNDAP protocol.
  - Plans to write data natively in netCDF-4 format in the future.
Some Metrics on PYTABLES 1.2

- **Core Library**
  - ~7.5 thousand LOC (Python)
  - ~2.8 thousand LOC (Pyrex)
  - ~4.0 thousand LOC (C)

- **Test Units**
  - ~3.1 thousand test units (Python)
  - ~23 thousand LOC (Python)

- **Documentation**
  - ~5.0 thousand lines of on-line doc strings
  - ~160 pages of Users’ Guide in PDF (and HTML)
Outline

1. An Introduction to the Python Language
   - Facts about Python
   - Scientific Packages
   - An example

2. PyTables
   - Overview
   - Usage Examples
   - PyTables Pro

3. CSTables
   - Overview
   - Design goals
   - Examples of Use
What is **PYTABLES PRO**?

It’s just a regular **PYTABLES** but with enhanced features:

- **Improved search speed**: Selections in tables with > 1 billion rows will be typically done in less than 1 second.

- **Complex queries**: Supports an unlimited combination of conditions.

- **Query optimizer**: Queries are analyzed, reordered and classified to get an optimum response time.

- **Support for complex indices**: Expressions like `col1 + col2*col3 + col4**3` can be indexed and used for selections afterwards.
Query time for Float64 column (not in cache)

- **Postgres 8**
- **PyTables Pro**
- **PyTables Std**

Time (s) vs. Number of rows

Index Selection Speed
Index Selection Speed: The Goal

Query time for Float64 column (not in cache)

- Postgres 8
- PyTables Pro (Simulation)
Current Status for PYTABLES PRO

- Improving the query response time.
- Remains to be done:
  - Complex queries.
  - Query optimizer.
- Date of release: 2nd quarter 2006 (tentative).
Outline

1. An Introduction to the Python Language
   - Facts about Python
   - Scientific Packages
   - An example

2. PyTables
   - Overview
   - Usage Examples
   - PyTables Pro

3. CStables
   - Overview
   - Design goals
   - Examples of Use
CSTABLES: PyTables Goes Client-Server

HDF5 Files

PyTables Layer

CSTables Server

Internet
TCP/IP

CSTables client
ViTables GUI
Outline

1. An Introduction to the Python Language
   - Facts about Python
   - Scientific Packages
   - An example

2. PYTABLES
   - Overview
   - Usage Examples
   - PYTABLES Pro

3. CSTABLES
   - Overview
   - Design goals
   - Examples of Use
Design Goals

- **Concurrency**: Allows multiple access to the same file at the same time.
  - **Locking system**: Avoids corruption of data.

- **High Throughput**: Data is transmitted in large blocks. High-speed compressor/decompressor used.

- **Client Cache**: Metadata is kept in the client-side when an object is first accessed.

- **Full PyTables API compatibility**: Existing PyTables programs can be re-used to access remote files without changing virtually anything.
Concurrency Issues

**Question**: CSTABLES does not provide threading or asynchronous features yet. So, how does it deal with several requests at a time?

**Answer**: Large data read and write requests are split into small chunks. This considerably improves server response time.

In addition, CSTABLES provides a *lock mechanism* that allows applications to explicitly put a lock on a node or on an entire subtree.

- Different locking access modes: **READ**, **WRITE** and **ALL**
Concurrency Issues

**Question:** CSTABLES does not provide threading or asynchronous features yet. So, how does it deal with several requests at a time?

**Answer:** Large data read and write requests are split into small chunks. This considerably improves server response time.

In addition, CSTABLES provides a *lock mechanism* that allows applications to explicitly put a lock on a node or on an entire subtree.

- Different locking access modes: READ, WRITE and ALL
Concurrency Issues

**Question:** CSTABLES does not provide threading or asynchronous features yet. So, how does it deal with several requests at a time?

**Answer:** Large data read and write requests are split into small chunks. This considerably improves server response time.

In addition, CSTABLES provides a *lock mechanism* that allows applications to explicitly put a lock on a node or on an entire subtree.

- Different locking access modes: READ, WRITE and ALL
CSTABLES caches some of the metadata of the PyTables object tree.

- When a client makes a change to the metadata, this change is *pushed* to the other clients’ caches.

- Easy to control which attributes should be cached and which should not.
  - You should try to cache primarily read-only attributes.

Caveat emptor:

Caching attributes that are updated very often might generate more traffic than attributes that are not cached at all!
Client Cache More in Depth

- CSTABLES caches some of the metadata of the PyTables object tree.

- When a client makes a change to the metadata, this change is *pushed* to the other clients' caches.

- Easy to control which attributes should be cached and which should not.
  - You should try to cache primarily read-only attributes.

**Caveat emptor:**

Caching attributes that are updated very often might generate more traffic than attributes that are not cached at all!
CSTABLES caches some of the metadata of the PyTables object tree.

When a client makes a change to the metadata, this change is *pushed* to the other clients’ caches.

Easy to control which attributes should be cached and which should not.

- You should try to cache primarily read-only attributes.

*Caveat emptor:* Caching attributes that are updated very often might generate more traffic than attributes that are not cached at all!
CSTABLES caches some of the metadata of the PYTABLES object tree.

When a client makes a change to the metadata, this change is *pushed* to the other clients’ caches.

Easy to control which attributes should be cached and which should not.

- You should try to cache primarily read-only attributes.

**Caveat emptor:**

Caching attributes that are updated very often might generate more traffic than attributes that are not cached at all!
Outline

1. An Introduction to the Python Language
   - Facts about Python
   - Scientific Packages
   - An example

2. PYTABLES
   - Overview
   - Usage Examples
   - PYTABLES Pro

3. CSTABLES
   - Overview
   - Design goals
   - Examples of Use
Two Different APIs

**PYTABLES API (prefix with `csclient -ip=server_IP`)**

```python
import tables
fileh = tables.openFile("file.h5")
print fileh.root.table.cols.col1[:]
fileh.close()
```

**CSTABLES API (no need to be prefixed)**

```python
from cstables.client import client
c=client.Session()
app=c.connect("your_FQDN_server")
fileh=app.openFile("file.h5")
print fileh.root.table.cols.col1[:]
fileh.close()
```
Two Different APIs

**PYTABLES API (prefix with `csclient -ip=server_IP`)**

```python
import tables
fileh = tables.openFile("file.h5")
print fileh.root.table.cols.col1[:]
fileh.close()
```

**CSTABLES API (no need to be prefixed)**

```python
from cstables.client import client

c=client.Session()
app=c.connect("your_FQDN_server")
fileh=app.openFile("file.h5")
print fileh.root.table.cols.col1[:]
fileh.close()
```
The main design features are already implemented and working.

Beta available (caveat: only works against PYTABLES 1.0).

Focus now is on checking & debugging possible errors, improving the throughput and bettering the Users’ Guide.

Future directions: threading, asynchronous communications.
# The PyTables Suite Is Getting Shape

## PyTables
The basic layer for the other components.

## PyTables Pro
PyTables with a twist: Complex searches and ultra-fast selections in tables.

## CStables
The client-server PyTables. It supports as well generic HDF5 files.

## ViTables
A data viewer for PyTables (and HDF5) files.
The **PyTables** Suite Is Getting Shape

**PyTables**
The basic layer for the other components.

**PyTables Pro**
**PyTables** with a twist: Complex searches and ultra-fast selections in tables.

**CsTables**
The client-server **PyTables**. It supports as well generic HDF5 files.

**ViTables**
A data viewer for **PyTables** (and HDF5) files.
The **PyTables** Suite Is Getting Shape

**PyTables**
The basic layer for the other components.

**PyTables Pro**
PyTables with a twist: Complex searches and ultra-fast selections in tables.

**CSTables**
The client-server PyTables. It supports as well generic HDF5 files.

**Vitables**
A data viewer for PyTables (and HDF5) files.
The **PyTables** Suite Is Getting Shape

**PyTables**
The basic layer for the other components.

**PyTables Pro**
PyTables with a twist: Complex searches and ultra-fast selections in tables.

**CSTables**
The client-server PyTables. It supports as well generic HDF5 files.

**ViTables**
A data viewer for PyTables (and HDF5) files.
Python is a joy for scientific computing. Start using it now!

The PyTables suite is designed to work with HDF5 files in an interactive, efficient and, most importantly, easy way.

Outlook

Working hard to release CSTables in the first quarter of 2006. PyTables Pro will come later on (~ 2nd quarter of 2006).

It would be nice to produce a parallel version of PyTables (long term goal).
Thank You!

Thanks also to:

- Elena Pourmal, for inviting us to the Workshop.
- Frank Baker, for being kind.
- The HDF Group for pushing forward PYTABLES.