Finding Needles in a Huge *DataStack*

Introducing *PyTables Pro*

Francesc Altet & Ivan Vilata

Cárbos Cooperativa Valenciana

CERN, Geneva, Switzerland.
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.
Cáribos 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.
What is PyTables and why it exists?
- Introduction to PyTables
- Features more in depth
- A few words about HDF5

PyTables Pro
- Introduction
- Benchmarks
- Current status

The complete PyTables Suite
- CSTables: A Client-Server PyTables system
- ViTables: A data Viewer for PyTables files
- Overview
Outline

1. **What is PYTABLES and why it exists?**
   - Introduction to PYTABLES
   - Features more in depth
   - A few words about HDF5

2. **PYTABLES PRO**
   - Introduction
   - Benchmarks
   - Current status

3. **The complete PYTABLES SUITE**
   - CSTABLES: A Client-Server PYTABLES system
   - ViTABLES: A data Viewer for PYTABLES files
   - Overview

Francesc Altet & Ivan Vilata

Finding Needles in a Huge DataStack
Motivation

Many applications need to save and read very large amounts of data. Coping with this is a real challenge!

Most computers today can deal with such large datasets. However, we should ask for an interface that should be usable by human beings.

Requirements:

- Interactivity: data analysis is an iterative process.
- Need to re-read many times the data: efficiency.
- Easy categorization of data.
- Ability to keep data for long time.
Motivation

Many applications need to save and read very large amounts of data. Coping with this is a real challenge!

Most computers today can deal with such large datasets. However, we should ask for an interface that should be usable by *human beings*.

Requirements:

- **Interactivity**: data analysis is an iterative process.
- Need to re-read many times the data: **efficiency**.
- Easy **categorization** of data.
- Ability to keep data for **long time**.
Many applications need to save and read very large amounts of data. Coping with this is a real challenge!

Most computers today can deal with such large datasets. However, we should ask for an interface that should be usable by human beings.

Requirements:

- **Interactivity**: data analysis is an iterative process.
- Need to re-read many times the data: **efficiency**.
- Easy **categorization** of data.
- Ability to keep data for **long time**.
What Does PYTABLES Offer?

**Interactivity** You can take immediate action based on previous feedback.

**Efficiency** Improves your productivity (very important for interactive work).

**Hierarchical structure** It allows you to categorize your data into smaller, related chunks.

**Backward/Forward compatibility** Based on HDF5, a general purpose framework with a great commitment with backward/forward compatibility.
Outline

1. What is PYTABLES and why it exists?
   - Introduction to PYTABLES
   - Features more in depth
   - A few words about HDF5

2. PYTABLES PRO
   - Introduction
   - Benchmarks
   - Current status

3. The complete PYTABLES SUITE
   - CSTABLES: A Client-Server PYTABLES system
   - ViTABLES: A data Viewer for PYTABLES files
   - Overview
PYTABLES Highlights (I)

- Designed for efficiently dealing with extremely large amounts of data.
- High level of flexibility for structuring your data:
  - Datatypes: scalars (numerical & strings), records, enumerated, time...
  - Multidimensional cells
  - Nested records
  - Variable length arrays
- Support for the complete Numeric/numarray/NumPy family.
PYTABLES Highlights (II)

- Transparent data compression support (Zlib, LZO, Bzip2...).
- Support of full 64-bit addressing in files, even on 32-bit platforms.
- Can handle generic HDF5 files (most of them).
- Aware of little/big endian issues (data is portable).
- It’s Open Source (BSD license).
PYTABLES Highlights (II)

- Transparent data compression support (Zlib, LZO, Bzip2...).
- Support of full 64-bit addressing in files, even on 32-bit platforms.
- Can handle generic HDF5 files (most of them).
- Aware of little/big endian issues (data is portable).
- It’s Open Source (BSD license).
Ease of Use

Natural naming

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

Support for generalized slicing

# step means a stride in the slice
array[idx, start:stop, :, start:stop:step]

Support for iterators

# 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
array[idx, start:stop, :, 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

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

Support for generalized slicing

```
# step means a stride in the slice
array[idx, start:stop, :, start:stop:step]
```

Support for iterators

```
# 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. What is PYTABLES and why it exists?
   - Introduction to PYTABLES
   - Features more in depth
   - A few words about HDF5

2. PYTABLES PRO
   - Introduction
   - Benchmarks
   - Current status

3. The complete PYTABLES SUITE
   - CSTABLES: A Client-Server PYTABLES system
   - ViTABLES: A data Viewer for PYTABLES files
   - Overview
Why HDF5?

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

- It is Open Source software.
- Outstanding backward & forward compatibility between library versions.
Why HDF5?

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

- It is Open Source software.
- Outstanding backward & forward compatibility between library versions.
Outline

1. What is PYTABLES and why it exists?
   - Introduction to PYTABLES
   - Features more in depth
   - A few words about HDF5

2. PYTABLES PRO
   - Introduction
   - Benchmarks
   - Current status

3. The complete PYTABLES SUITE
   - CSTABLES: A Client-Server PYTABLES system
   - ViTABLES: A data Viewer for PYTABLES files
   - Overview
What is **PyTables** Pro?

Plainly stated, it’s an enhanced version of **PyTables**. It sports:

- **Complex queries**: Supports an unlimited combination of conditions.
- **Much improved search speed**: Selections in tables having up to 1 billion rows can be typically done in less than 1 second.
- **Customizable index quality**: The indexes can be created with an *optimization level* (specified as a number ranging from 0 to 9).
- **Support for complex indexes**: Expressions like “*col1 + col2*col3 + col4**3*” can be indexed and used for selections afterwards.
Outline

1. What is PYTABLES and why it exists?
   - Introduction to PYTABLES
   - Features more in depth
   - A few words about HDF5

2. PYTABLES PRO
   - Introduction
   - Benchmarks
   - Current status

3. The complete PYTABLES SUITE
   - CSTABLES: A Client-Server PYTABLES system
   - ViTABLES: A data Viewer for PYTABLES files
   - Overview

Francesc Altet & Ivan Vilata
Finding Needles in a Huge DataStack
Several benchmarks have been conducted in order to analyze how PYTABLES PRO performs in comparison with existing tools to save data persistently.

The benchmarks consist in writing and selecting table data that fulfills a series of conditions. Indexed queries have been included as well.

The effect of the quality of the index in queries has been analyzed.

Finally, the effect of compression in speed is also checked.
A simple table with 4 columns has been adopted for conducting the benchmarks:

Record Size: 24 bytes

class Record(tables.IsDescription):
    col1 = tables.IntCol()
    col2 = tables.IntCol(indexed=True)
    col3 = tables.FloatCol()
    col4 = tables.FloatCol(indexed=True)
The simple query (labeled as ‘simple’ in benchmarks):

**PyTables**

```python
condition = "(%s<=col) \& \& (col<=%s)" % (lim1, lim2)
result = [ row['col1'] for row in table.where(condition) ]
```

**Postgres**

```sql
condition = "(%s<=col) and (col<=%s)" % (lim1, lim2)
cursor.execute("select col1 from table where %s" % condition)
```
The complex query (labeled as 'complex1' in benchmarks):

**PyTables**

```python
condition = "(%s<=col) & (col<=%s) & \n(sqrt(col1+3.1*col2+col3*col4)>3)" % (lim1, lim2)
result = [ row['col1'] for row in
          table.where(condition) ]
```

**Postgres**

```python
condition = "(%s<=col) and (col<=%s) and \n(sqrt(col1+3.1*col2+col3*col4)>3)" % (lim1, lim2)
cursor.execute("select col1 from table where %s" %
               condition)
```
What is PYTABLES and why it exists?
PYTABLES PRO
The complete PYTABLES SUITE
Summary

Data Distribution For Filling Columns

Distribution of values for columns

Francesc Altet & Ivan Vilata
Finding Needles in a Huge DataStack
What is PyTables and why it exists?

PyTables Pro

The complete PyTables Suite

Summary

Introduction

Benchmarks

Current status

Benchmark Setup

- Opteron machine at 2 GHz (2 dual-cores)
- 4 GB of RAM
- SATA2 disk @ 7200 RPM
- SuSE GNU/Linux 10.0 (x86-64)
- PyTables Pro 1.0 alpha1
- HDF5 1.8.0 alpha3
- numarray 1.5.1
- Postgres 8.0.3
In-kernel selections are optimized in C-space and this is why they are much faster than selections in Python-space.

**Pythonic selection**

```python
r = [row['col1'] for row in table.iterrows()
    if row['col'] >= lim1 and row['col'] <= lim2]
```

**In-Kernel selection**

```python
condition = "(%s<=col) & (col<=%s)" % (lim1, lim2)
r = [row['col1'] for row in table.where(condition)]
```
What is PYTABLES and why it exists?

Introduction
Benchmarks
The complete PYTABLES SUITE
Summary

Keys for Interpreting the Next Plots

nhits  The number of rows resulting from a selection.
simple The selection has been done with the ’simple’ condition.
complex1 The selection has been done with the ’complex’ condition.
in cache The entire dataset fits perfectly in cache.
not in cache The dataset doesn’t fit in cache even if compression is used.
In-Kernel Selects (simple, dataset in cache)

Query time for int32 (not indexed)

- PyTables Pro 10Mrow simple
- PyTables Pro 10Mrow simple lzo
- PyTables Pro 10Mrow simple zlib
- Postgres 8.0 10Mrow simple
What is **PyTables** and why does it exist?

**PyTables Pro**

The complete **PyTables Suite**

**Summary**

**Introduction**

**Benchmarks**

**Current status**

In-kernel Selects (simple, dataset in cache)

---

**Query time for float64 (not indexed)**

<table>
<thead>
<tr>
<th>Number of hits</th>
<th>PyTables Pro 10Mrow simple</th>
<th>PyTables Pro 10Mrow simple lzo</th>
<th>PyTables Pro 10Mrow simple zlib</th>
<th>Postgres 8.0 10Mrow simple</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Time (s)**

- PyTables Pro 10Mrow simple
- PyTables Pro 10Mrow simple lzo
- PyTables Pro 10Mrow simple zlib
- Postgres 8.0 10Mrow simple

---

Finding Needles in a Huge DataStack
In-kernel Selects (complex, dataset in cache)

Query time for int32 (not indexed)

- **PyTables Pro 10Mrow complex1**
- **PyTables Pro 10Mrow complex1 lzo**
- **PyTables Pro 10Mrow complex1 zlib**
- **Postgres 8.0 10Mrow complex1**

Number of hits

0, 2, 4, 6, 8, 10, 12

Time (s)

Francesc Altet & Ivan Vilata

Finding Needles in a Huge DataStack
In-kernel Selects (complex, dataset in cache)

Query time for float64 (not indexed)

- PyTables Pro 10Mrow complex1
- PyTables Pro 10Mrow complex1 lzo
- PyTables Pro 10Mrow complex1 zlib
- Postgres 8.0 10Mrow complex1

Finding Needles in a Huge DataStack
Conclusions for In-Kernel Selects (dataset in cache)

- For simple, non-indexed queries, PYTABLES PRO can be between 1.1x and up to 8x faster than Postgres.
- For complex queries, PYTABLES PRO is faster by a 2x and up to 5x.
- Compression doesn’t affect too much to PYTABLES PRO performance:
  - LZO selects are slower by between a 1.1x and 1.5x (not too much!)
  - Zlib selects are slower by between a 2x and 3x
- Double precision floating point (Float64) and integers (Int32) perform very similarly on every case.
In-kernel selects (simple, dataset not in cache)

Query time for int32 (not indexed)

- PyTables Pro 1Grow simple
- PyTables Pro 1Grow simple lzo
- PyTables Pro 1Grow simple zlib
- Postgres 8.0 1Grow simple

Number of hits vs. Time (s)
In-kernel Selects (complex, dataset not in cache)

Query time for int32 (not indexed)

- **PyTables Pro 1Grow complex1**
- **PyTables Pro 1Grow complex1 lzo**
- **PyTables Pro 1Grow complex1 zlib**
- **Postgres 8.0 1Grow complex1**

<table>
<thead>
<tr>
<th>Number of hits</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td></td>
</tr>
<tr>
<td>200</td>
<td></td>
</tr>
<tr>
<td>400</td>
<td></td>
</tr>
<tr>
<td>800</td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td></td>
</tr>
<tr>
<td>4000</td>
<td></td>
</tr>
<tr>
<td>8000</td>
<td></td>
</tr>
<tr>
<td>10000</td>
<td></td>
</tr>
<tr>
<td>20000</td>
<td></td>
</tr>
<tr>
<td>40000</td>
<td></td>
</tr>
<tr>
<td>80000</td>
<td></td>
</tr>
<tr>
<td>100000</td>
<td></td>
</tr>
<tr>
<td>200000</td>
<td></td>
</tr>
<tr>
<td>400000</td>
<td></td>
</tr>
<tr>
<td>800000</td>
<td></td>
</tr>
<tr>
<td>1000000</td>
<td></td>
</tr>
</tbody>
</table>
Conclusions for In-Kernel Selects (dataset not in cache)

- For both simple and complex queries, the times are very similar (the bottleneck is in the disk access).
- PyTables PRO (without compression) is typically 2x faster than Postgres.
- Compression has a clear benefit on PyTables PRO performance:
  - LZO selects are faster by an additional 1.9x ~ 2.4x (!)
  - Zlib selects are faster by an additional 1.2x ~ 1.4x
Indexing is the more common way to accelerate searches.

Clever use of indexes can reduce look-up times from hours to milliseconds.

However, index computation often requires a large amount of time (and space!).

If you have huge datasets, you need to be very careful and select only the indexes that you are going to use.
What is PyTables and why it exists?

PyTables Pro
The complete PyTables Suite

Summary

Introduction

Benchmarks

Current status

Selectable Index Quality

PyTables Pro allows the user to choose the level of optimization of an index.

Low Quality Index Allows much faster index creation times but will result in degraded searching times (but may be perfectly useful in many cases!).

High Quality Index Allows better query speed at the expense of consuming more time for index creation.

In addition, if space is important, PyTables Pro will let you compress the indexes in order to minimize their size.

Francesc Altet & Ivan Vilata

Finding Needles in a Huge DataStack
number of rows  The number of rows for the different table sizes.

index not in cache  The corresponding part of the index is not loaded in memory.

index in cache  The appropriate part of the index for the query is already loaded.

PyTables Std  The regular PyTables library (i.e. not the PRO version).
Index Creation Speed (not compressed)
Conclusions for Index Creation Times

- For minimum index quality, PYTABLES PRO can index up to 20x times faster than Postgres.
- For maximum index quality, PYTABLES PRO can create indexes in the range of 0.5x and 1x of the time taken by Postgres.
- PYTABLES PRO never takes longer to index than Postgres.
Simple Query Speed
(index not in cache, not compressed)

![PyTables Pro O0 vs PyTables Pro O3 vs PyTables Pro O6 vs PyTables Pro O9 vs PostGres vs PyTables Std]

Francesc Altet & Ivan Vilata
Finding Needles in a Huge DataStack
Simple Query Speed
(index not in cache, compressed with LZO)

Query time for int32 (index not in cache)

<table>
<thead>
<tr>
<th>Number of rows</th>
<th>PyTables Pro lzo O0</th>
<th>PyTables Pro lzo O3</th>
<th>PyTables Pro lzo O6</th>
<th>PyTables Pro lzo O9</th>
<th>PostGres</th>
<th>PyTables Std lzo</th>
</tr>
</thead>
<tbody>
<tr>
<td>10^3</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>10^4</td>
<td>10^2</td>
<td>10^2</td>
<td>10^2</td>
<td>10^2</td>
<td>10^2</td>
<td>10^2</td>
</tr>
<tr>
<td>10^5</td>
<td>10^3</td>
<td>10^3</td>
<td>10^3</td>
<td>10^3</td>
<td>10^3</td>
<td>10^3</td>
</tr>
<tr>
<td>10^6</td>
<td>10^4</td>
<td>10^4</td>
<td>10^4</td>
<td>10^4</td>
<td>10^4</td>
<td>10^4</td>
</tr>
<tr>
<td>10^7</td>
<td>10^5</td>
<td>10^5</td>
<td>10^5</td>
<td>10^5</td>
<td>10^5</td>
<td>10^5</td>
</tr>
<tr>
<td>10^8</td>
<td>10^6</td>
<td>10^6</td>
<td>10^6</td>
<td>10^6</td>
<td>10^6</td>
<td>10^6</td>
</tr>
<tr>
<td>10^9</td>
<td>10^7</td>
<td>10^7</td>
<td>10^7</td>
<td>10^7</td>
<td>10^7</td>
<td>10^7</td>
</tr>
</tbody>
</table>
Simple Query Speed
(index not in cache, compressed with Zlib)
Conclusions for Indexed Queries (index not in cache)

- For lowest index quality, PYTABLES PRO can be up to 10x slower than Postgres, but still, doing a query in a table with 1 billion rows in less than 3 seconds can be useful.
- For maximum index quality, PYTABLES PRO times are similar to those of Postgres, and queries in tables with 1 billion rows can be typically done in less than 1 second (this was our primary goal).
- Using compressed indexes effectively accelerates the searches, specially for large tables where, in some cases, can be up to 2x faster than Postgres.
What is PyTables and why it exists?

PyTables Pro
The complete PyTables Suite
Summary

Simple Query Speed
(index in cache, not compressed)

Query time for int32 (index in cache)

- PyTables Pro O0
- PyTables Pro O3
- PyTables Pro O6
- PyTables Pro O9
- PostGres
- PyTables Std

Number of rows
10^-3
10^-2
10^-1
10^0
10^1

Time (s)
10^0
10^1
10^2
10^3

Francesc Altet & Ivan Vilata
Finding Needles in a Huge DataStack
What is PyTables and why it exists?

PyTables Pro
The complete PyTables Suite
Summary

Simple Query Speed
(index in cache, compressed with LZO)

Query time for int32 (index in cache)

Number of rows

Time (s)

PyTables Pro lzo O0
PyTables Pro lzo O3
PyTables Pro lzo O6
PyTables Pro lzo O9
PostGres
PyTables Std lzo

Francesc Altet & Ivan Vilata
Finding Needles in a Huge DataStack
Simple Query Speed
(index in cache, compressed with Zlib)

What is PyTables and why it exists?

The complete PyTables Suite

Introduction

Benchmarks

Current status

Summary

Francesc Altet & Ivan Vilata

Finding Needles in a Huge DataStack
Conclusions for Indexed Queries (index in cache)

- For lowest index quality, PYTABLES PRO can be more than 100x slower than Postgres, but still, doing a query in a table with 1 billion rows in less than 0.1 seconds may be enough for many situations.

- For maximum index quality, PYTABLES PRO times are behind 1 ms for all the range of tables up to 1 billion rows.

- Compression when index is in cache reduces performance between 1.1x and 3x, but the times are still very good.
Special case: Querying Huge Tables Exceeding the Cache Size.

- A typical case of use where PYTABLES PRO shines is doing complex look-ups in tables that are too large to fit in cache and selections that don’t get repeated in time.
- The next plot is made for the complex query in a table with 1 billion rows, a size that exceeds by a factor of 10x the amount of available memory in our platform.
Special Case: The Plot

Query time for int32 (index not in cache)

- **PyTables Pro 1Grow complex1**
- **PyTables Pro 1Grow complex1 lzo**
- **PyTables Pro 1Grow complex1 zlib**
- **Postgres 8.0 1Grow complex1**

Number of hits vs Time (s)
Conclusions For All Benchmarks

- When the dataset (or index) is in cache, compression can actually improve performance in both indexed and non-indexed searches (up to 8x faster).
- When the dataset (or index) is not in cache, compression (specially LZO) doesn’t hurt performance very much in both indexed and non-indexed searches.
- Selectable optimization of indexes can adapt better to your searching needs.
- Comparisons with Postgres show that PyTables Pro searching capabilities are very competitive, specially when dealing with very large datasets (up to 10x faster).
Outline

1. What is PYTABLES and why it exists?
   - Introduction to PYTABLES
   - Features more in depth
   - A few words about HDF5

2. PYTABLES PRO
   - Introduction
   - Benchmarks
   - Current status

3. The complete PYTABLES SUITE
   - CSTABLES: A Client-Server PYTABLES system
   - ViTABLES: A data Viewer for PYTABLES files
   - Overview

Finding Needles in a Huge DataStack
Improving the query response time (there is always room for this ;-) )

Remains to be done:
- Support for improved string searches.
- Complex indexes.
- Query optimizer.

Date of release: October 2006.
Outline

1. What is PYTABLES and why it exists?
   - Introduction to PYTABLES
   - Features more in depth
   - A few words about HDF5

2. PYTABLES PRO
   - Introduction
   - Benchmarks
   - Current status

3. The complete PYTABLES SUITE
   - CSTABLES: A Client-Server PYTABLES system
   - ViTABLES: A data Viewer for PYTABLES files
   - Overview
CSTABLES: PYTABLES Goes Client-Server

HDF5 Files +
PyTables Layer +
CSTables Server

Internet TCP/IP

CSTables client + ViTables GUI

Finding Needles in a Huge DataStack
CSTABLES Status & Availability

- 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 User’s Guide.
- Future directions: threading, asynchronous communications.
Outline

1. What is PyTables and why it exists?
   - Introduction to PyTables
   - Features more in depth
   - A few words about HDF5

2. PyTables Pro
   - Introduction
   - Benchmarks
   - Current status

3. The complete PyTables Suite
   - CSTables: A Client-Server PyTables system
   - ViTables: A data Viewer for PyTables files
   - Overview
ViTABLES Status & Availability

- First version (1.0) was out in March 2006
- Getting ready for releasing ViTABLES 1.1. The main improvements will be:
  - Higher speed in opening datasets.
  - Querying of tables has been improved.
  - General usability enhancements.
  - All known bugs have been fixed.
Outline

1. What is PYTABLES and why it exists?
   - Introduction to PYTABLES
   - Features more in depth
   - A few words about HDF5

2. PYTABLES PRO
   - Introduction
   - Benchmarks
   - Current status

3. The complete PYTABLES SUITE
   - CSTABLES: A Client-Server PYTABLES system
   - ViTABLES: A data Viewer for PYTABLES files
   - Overview

Francesc Altet & Ivan Vilata
Finding Needles in a Huge DataStack
The PyTables Suite is Getting Shape

**PyTables**
The free library for dealing with large amounts of data.

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

**CSTables**
The client-server PyTables.

**ViTables**
A data viewer for PyTables (and HDF5) files.
**Summary**

- **PyTables** is practical and powerful for dealing with extremely large amounts of data. Give it a try!
- You need extreme speed?: stay tuned with **PyTables Pro**.
- The **PyTables Suite** is designed to work with large data files in an interactive, efficient and pleasant way.

**Outlook**

- Working hard to release **PyTables Pro** (October 2006).
- **CSTables** will come later on (~ 2nd quarter of 2007).
**Summary**

- **PyTables** is practical and powerful for dealing with extremely large amounts of data. Give it a try!
- You need extreme speed?: stay tuned with **PyTables Pro**.
- The **PyTables Suite** is designed to work with large data files in an interactive, efficient and pleasant way.

**Outlook**

- Working hard to release **PyTables Pro** (October 2006).
- **CSTables** will come later on (~ 2nd quarter of 2007).
Thank You!

Thanks also to:

- **The PYTables community**, for using, testing, reporting bugs and providing solutions and extensions for the open source PYTables.
- **Wladimiro Díaz**, from University of València for providing the benchmarking platform.
- **Scott Prater**, for his guidance and common sense.
- **The HDF Group**, for pushing PYTables forward.
What is PyTables and why it exists?

PyTables is a high-performance library for managing large arrays of data, providing a scalable alternative to database technology.

The complete PyTables Suite

Summary

More Info

http://www.pytables.org
The site for the open source PyTables project.

http://hdf.ncsa.uiuc.edu/HDF5/
The library and format in which is based PyTables.

http://www.carabos.com
The company behind the PyTables Suite development.