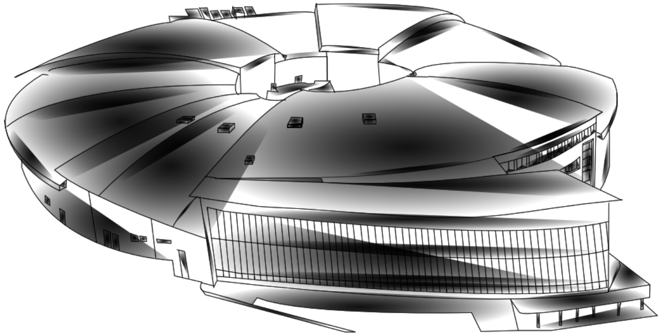
Taurus: Big & Small

from Particle Accelerators to Desktop Labs



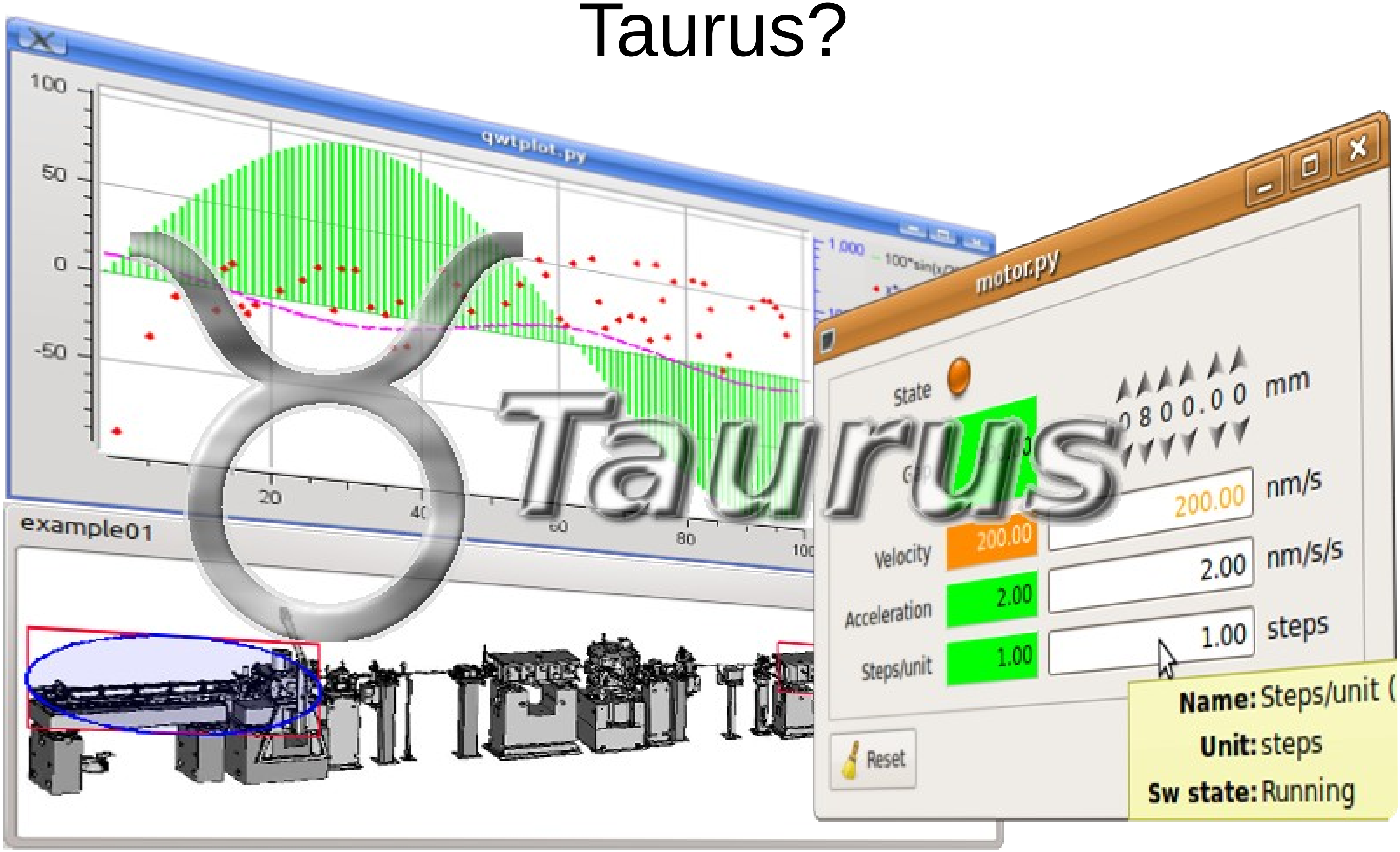
by: Carlos Pascual-Izarra

+ G. Cuní, C. Falcón-Torres, D. Fernández-Carreiras, Z. Reszela, M. Rosanes & O. Prades-Palacios

ICALEPCS2017

ALBA

Taurus?



State
Velocity 200.00 | 200.00 nm/s
Acceleration 2.00 | 2.00 nm/s/s
Steps/unit 1.00 | 1.00 steps
0800.00 mm

Name: Steps/unit (
Unit: steps
Sw state: Running

- **Taurus** is a framework for building control and data acquisition **CLIs** and **GUIs**
- It is based on **Python** and extends **PyQt**
- It supports plugins for various control systems (**Tango**, **EPICS**,...) or data sources (**HDF5**, **Python eval**,...)

MORE INFO https://taurus-scada.org

Taurus Community

ALBA

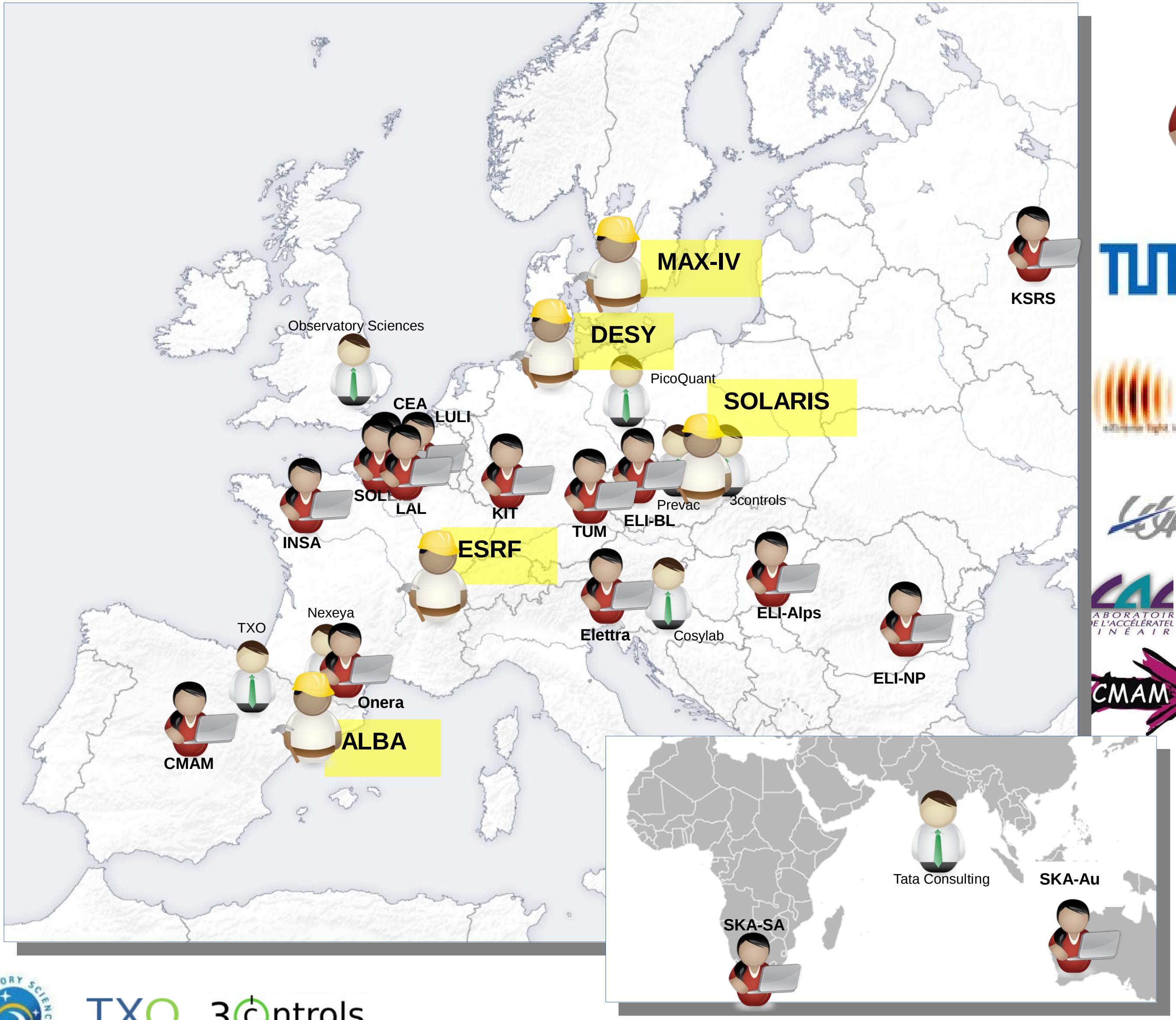
DESY

ESRF

MAX IV

SOLARIS
NATIONAL SYNCHROTRON RADIATION CENTRE

debian



TUM

KIT
Karlsruhe Institute of Technology

eli
extreme light infrastructure

cea

INSA
RENNES

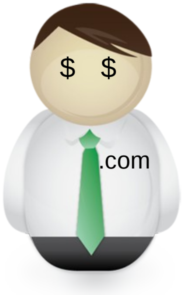
LAL
LABORATOIRE DE L'ACCÉLÉRATEUR LINÉAIRE

ONERA
THE FRENCH AEROSPACE LAB

CMAM

SKA
SQUARE KILOMETRE ARRAY

SOLEIL
SYNCHROTRON



NEXEYA

COSYLAB

OBSERVATORY SCIENCES

PicoQuant

TXO

PREVAC

3controls

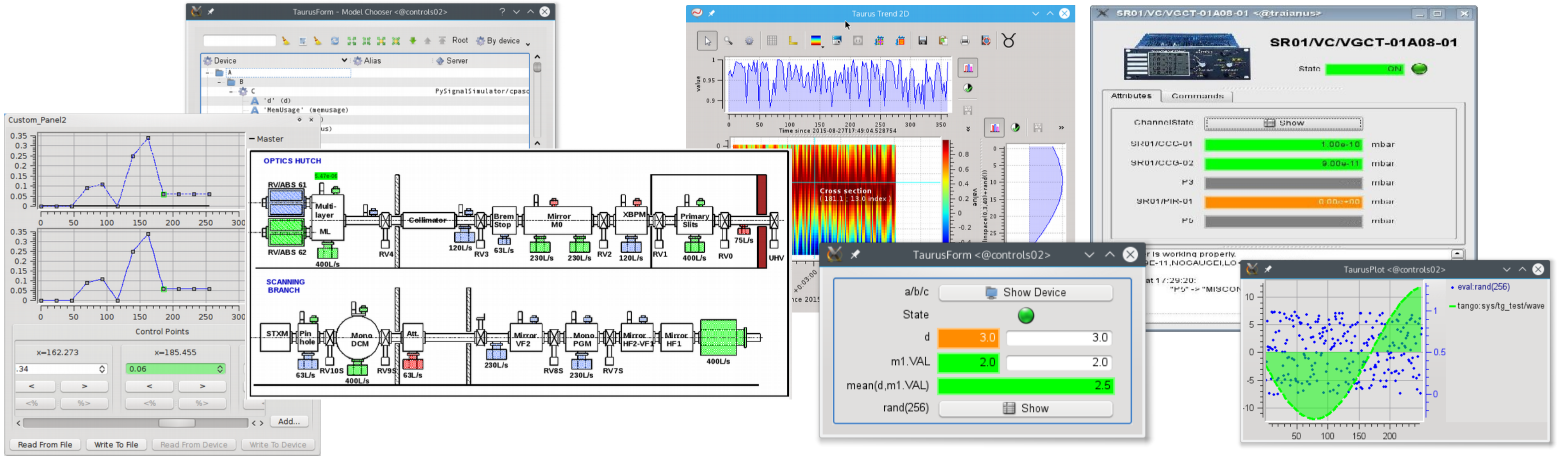
TATA
TATA CONSULTANCY SERVICES

MORE INFO

http://github.com/taurus-org/taurus

Structure of Taurus

Taurus Qt Widgets



TaurusForm <@controls02>

a/b/c Show Device
State ●
d 3.0 3.0
m1.VAL 2.0 2.0
mean(d,m1.VAL) 2.5
rand(256) Show

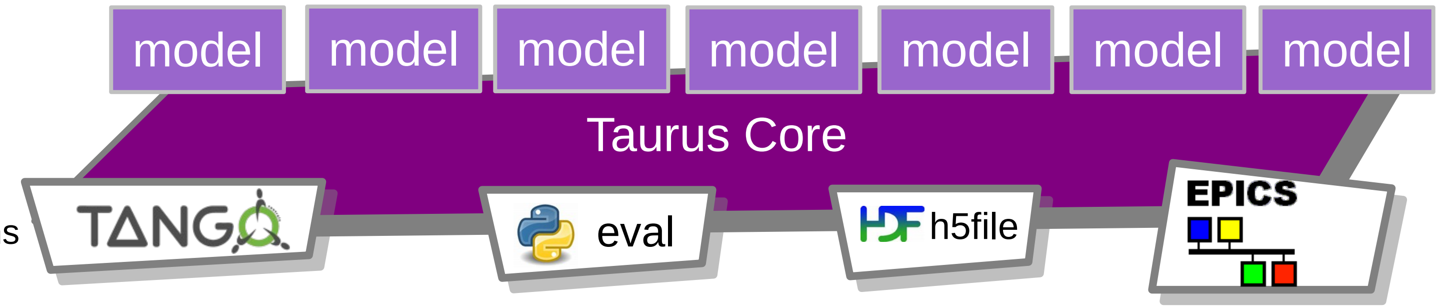
SR01/VC/VGCT-01A08-01

State ON
ChannelState Show
SR01/CCG-01 1.00e-10 mbar
SR01/CCG-02 9.00e-11 mbar
P3 mbar
SR01/PIR-01 0.00e+00 mbar
P5 mbar

Taurus Core

Model Objects

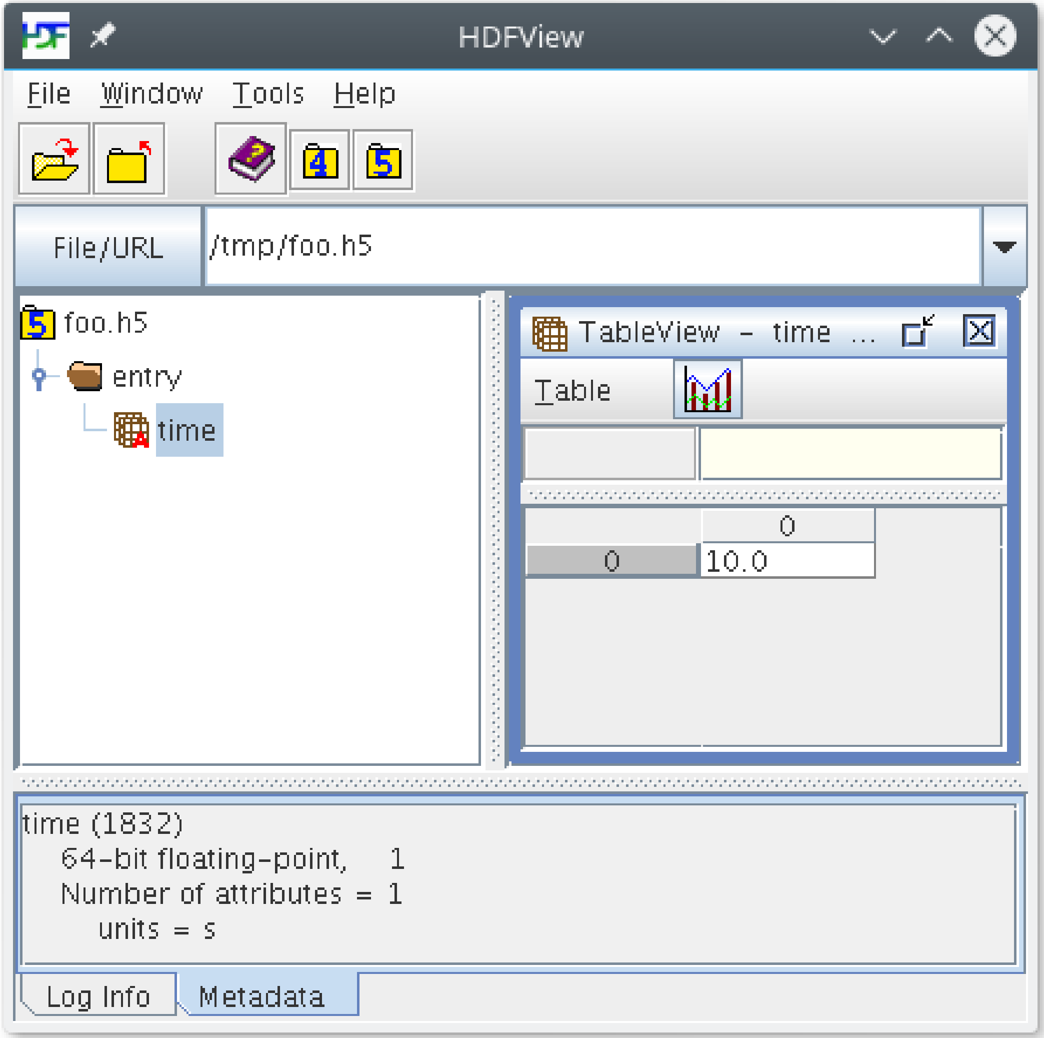
Scheme plugins



Data Sources

Dealing with new data sources

e.g., how to read data from a HDF5 file as an attribute?

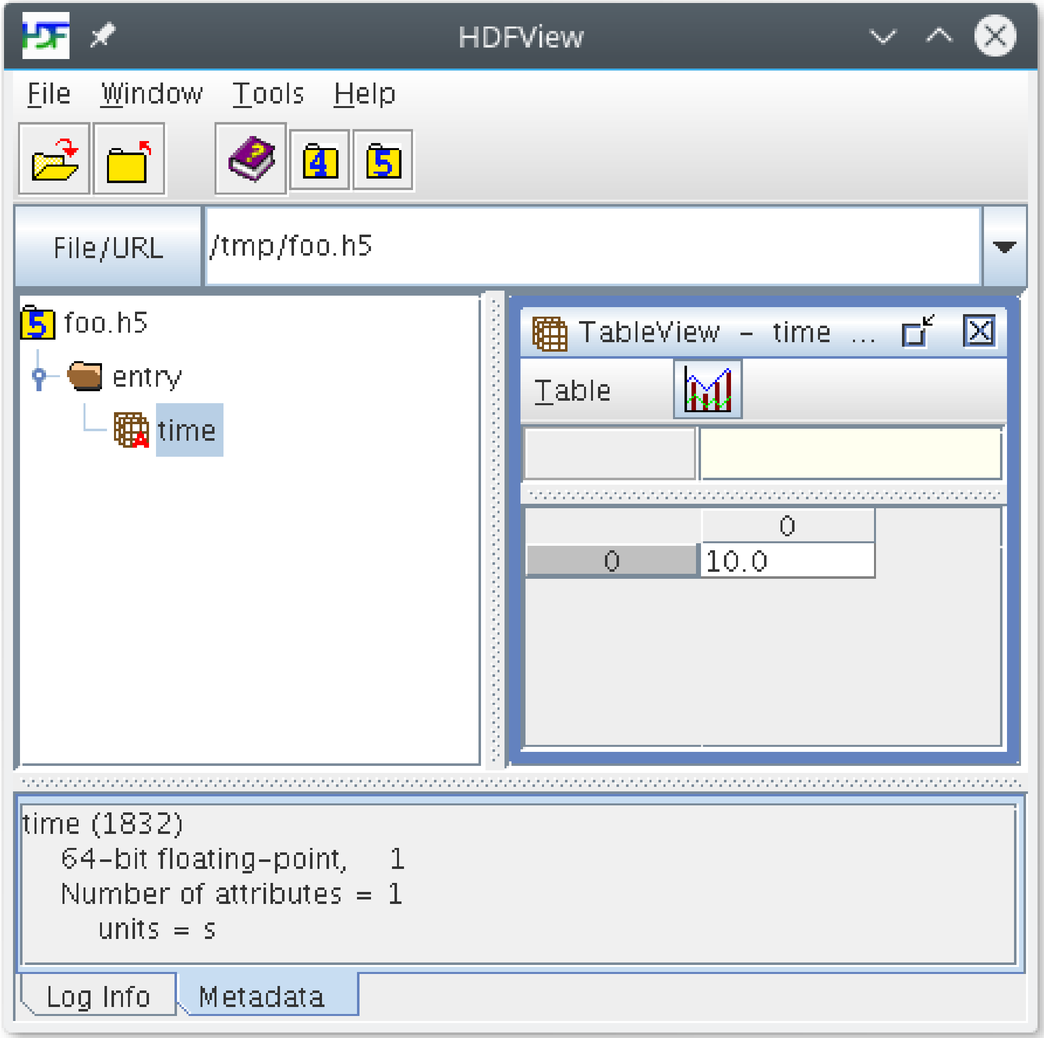


```
time (1832)
   64-bit floating-point,    1
   Number of attributes = 1
      units = s
```

Dealing with new data sources

e.g., how to read data from a HDF5 file as an attribute?

Approach 1: Distributed Control System



/tmp/foo.h5

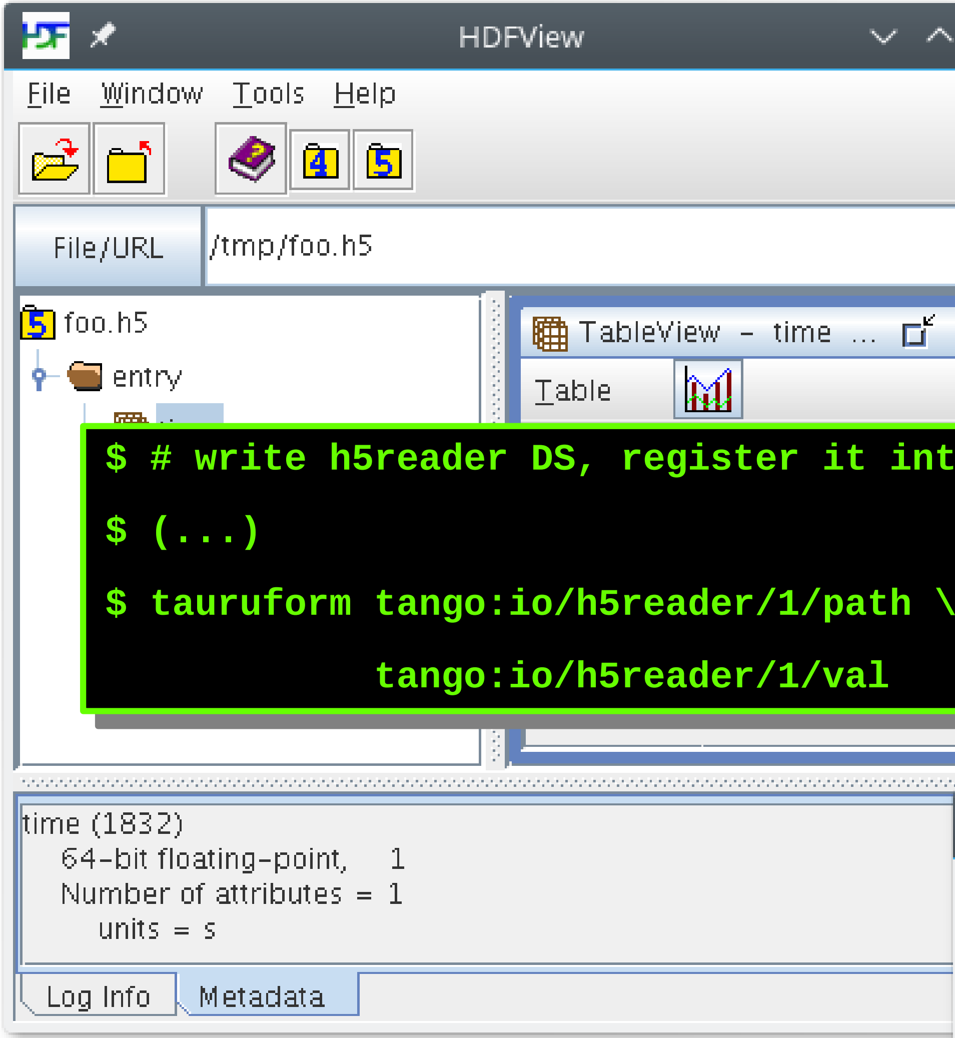
	0
0	10.0

```
time (1832)
   64-bit floating-point,    1
   Number of attributes = 1
      units = s
```

Dealing with new data sources

e.g., how to read data from a HDF5 file as an attribute?

Approach 1: Distributed Control System



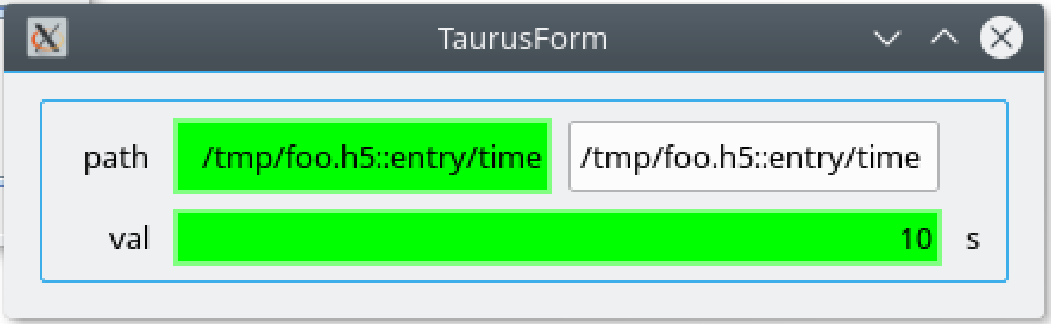
/tmp/foo.h5

foo.h5
entry

TableView – time …

```
time (1832)
   64-bit floating-point,    1
   Number of attributes = 1
      units = s
```

```
$ # write h5reader DS, register it into the DB and launch it
$ (...)
$ tauruform tango:io/h5reader/1/path \
            tango:io/h5reader/1/val
```

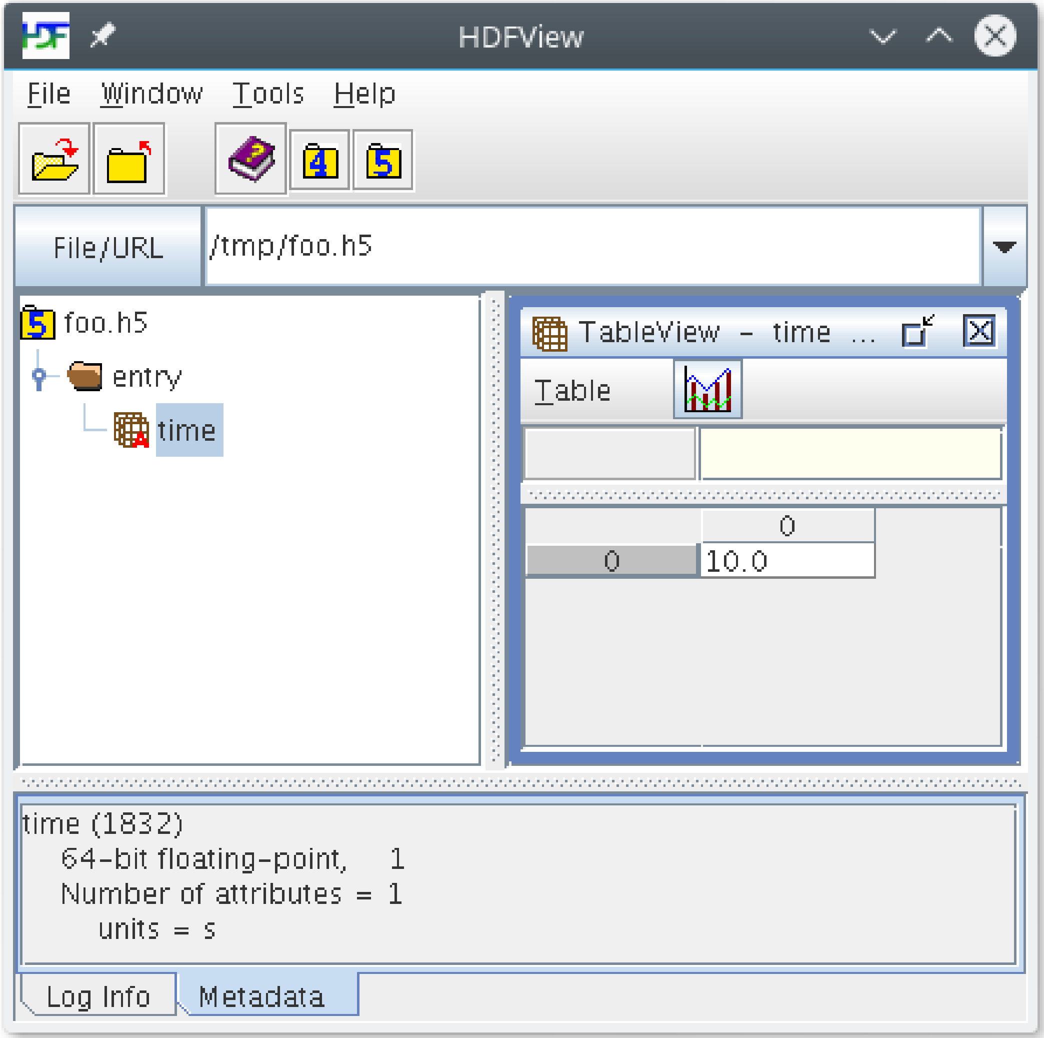


path	/tmp/foo.h5::entry/time	/tmp/foo.h5::entry/time	
val	10		s

Dealing with new data sources

e.g., how to read data from a HDF5 file as an attribute?

Approach 2: Evaluation scheme

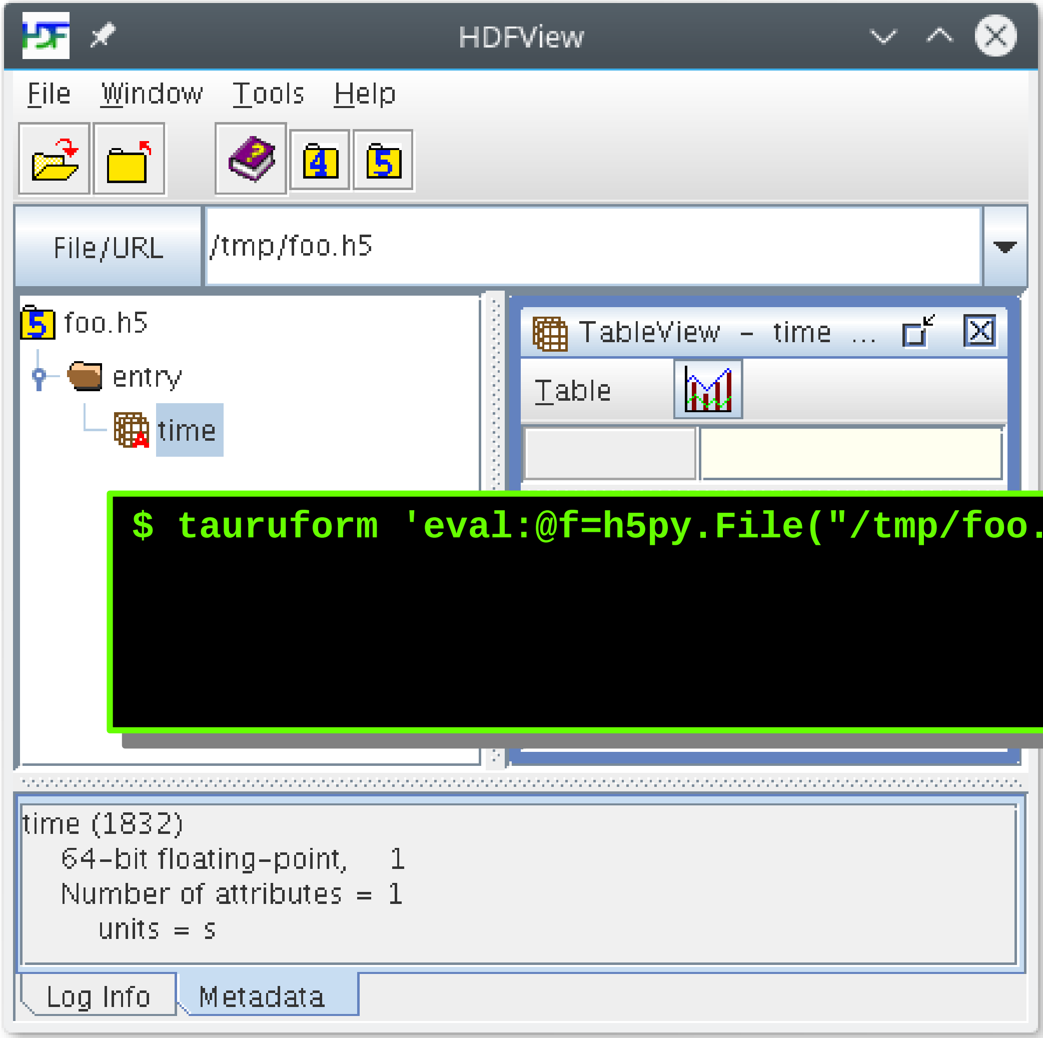


```
time (1832)
   64-bit floating-point,    1
   Number of attributes = 1
      units = s
```

Dealing with new data sources

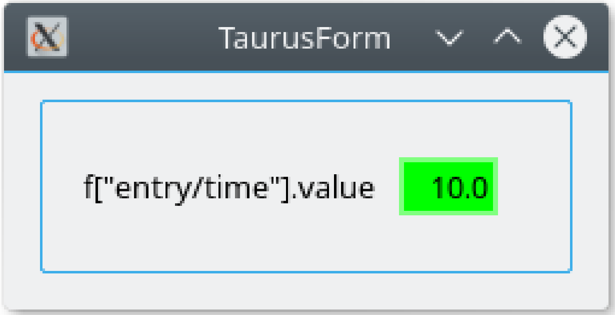
e.g., how to read data from a HDF5 file as an attribute?

Approach 2: Evaluation scheme



```
time (1832)
   64-bit floating-point,    1
   Number of attributes = 1
      units = s
```

```
$ tauruform 'eval:@f=h5py.File("/tmp/foo.h5")/f["entry/time"].value'
```

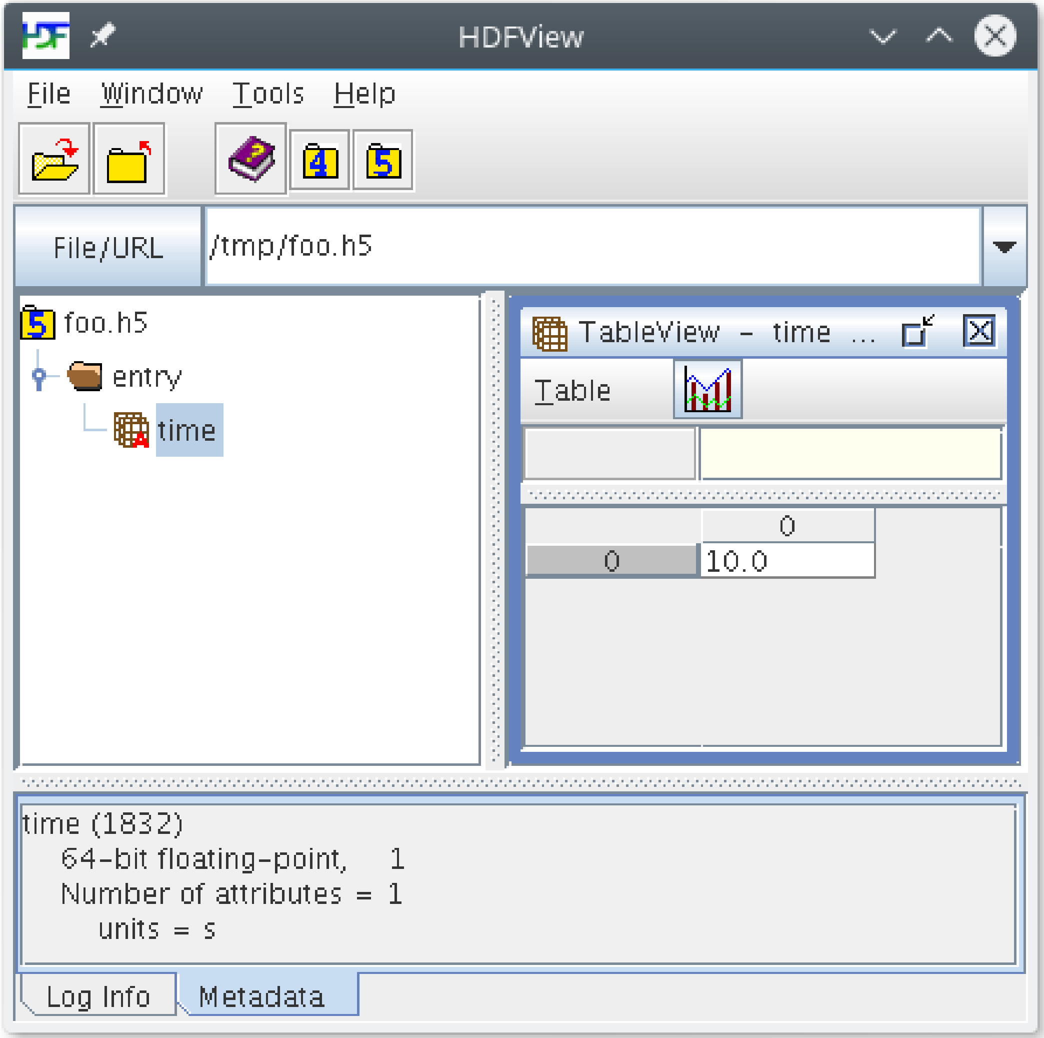


f["entry/time"].value 10.0

Dealing with new data sources

e.g., how to read data from a HDF5 file as an attribute?

Approach 3: Custom scheme



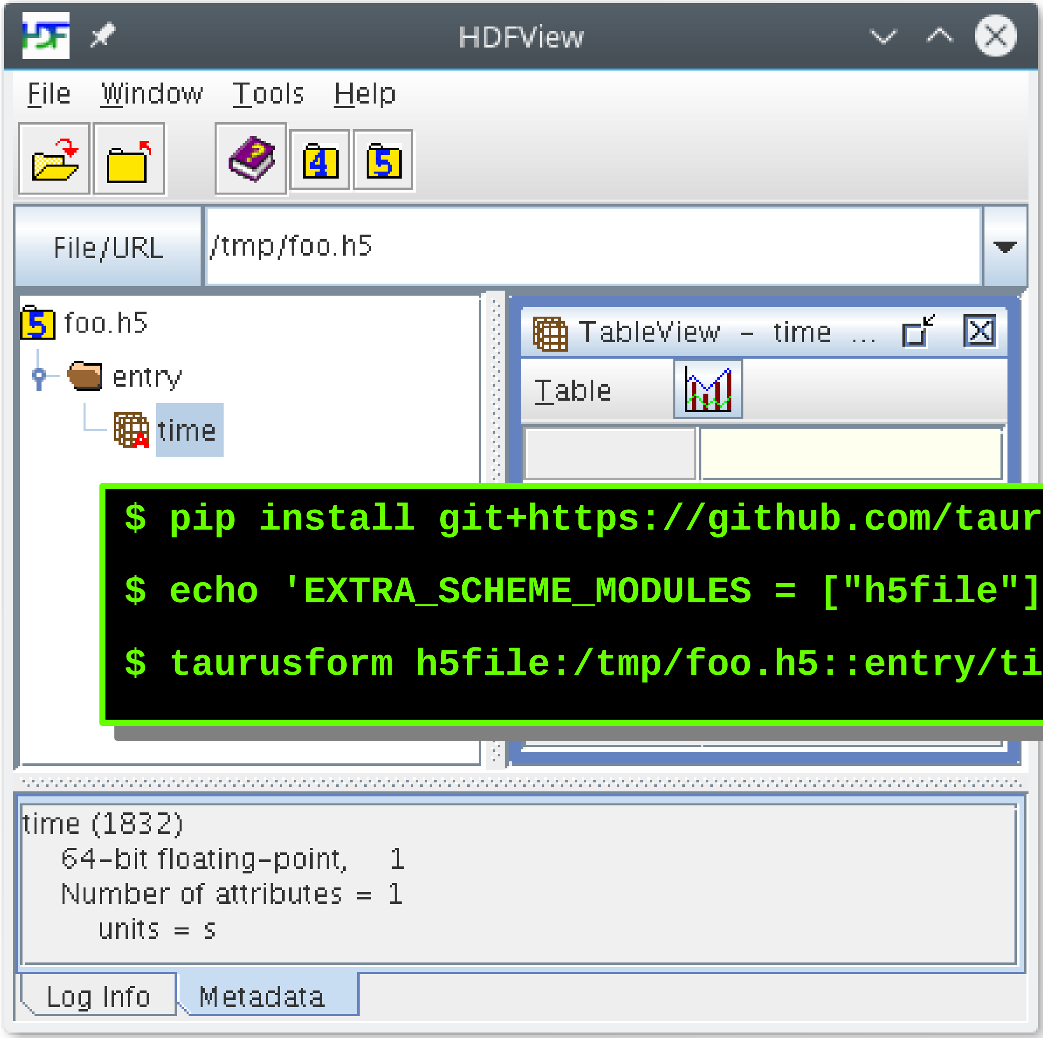
	0
0	10.0

```
time (1832)
   64-bit floating-point,    1
   Number of attributes = 1
      units = s
```

Dealing with new data sources

e.g., how to read data from a HDF5 file as an attribute?

Approach 3: Custom scheme

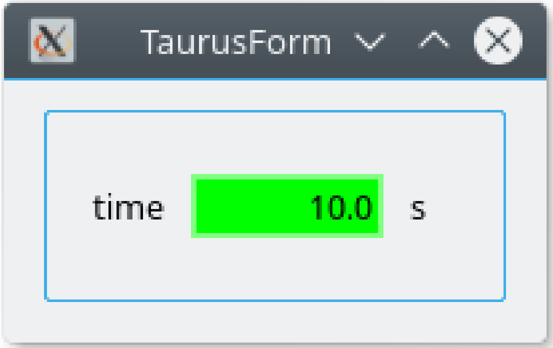


File/URL /tmp/foo.h5

```
foo.h5
  entry
    time
```

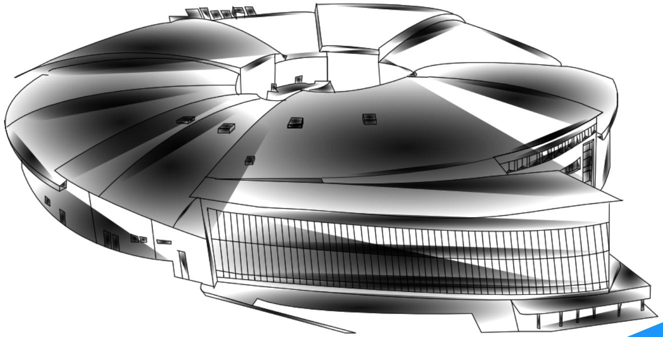
```
time (1832)
   64-bit floating-point,    1
   Number of attributes = 1
      units = s
```

```
$ pip install git+https://github.com/taurus-org/h5file-scheme.git
$ echo 'EXTRA_SCHEME_MODULES = ["h5file"]' >> taurus/tauruscustomsettings.py
$ taurusform h5file:/tmp/foo.h5::entry/time
```



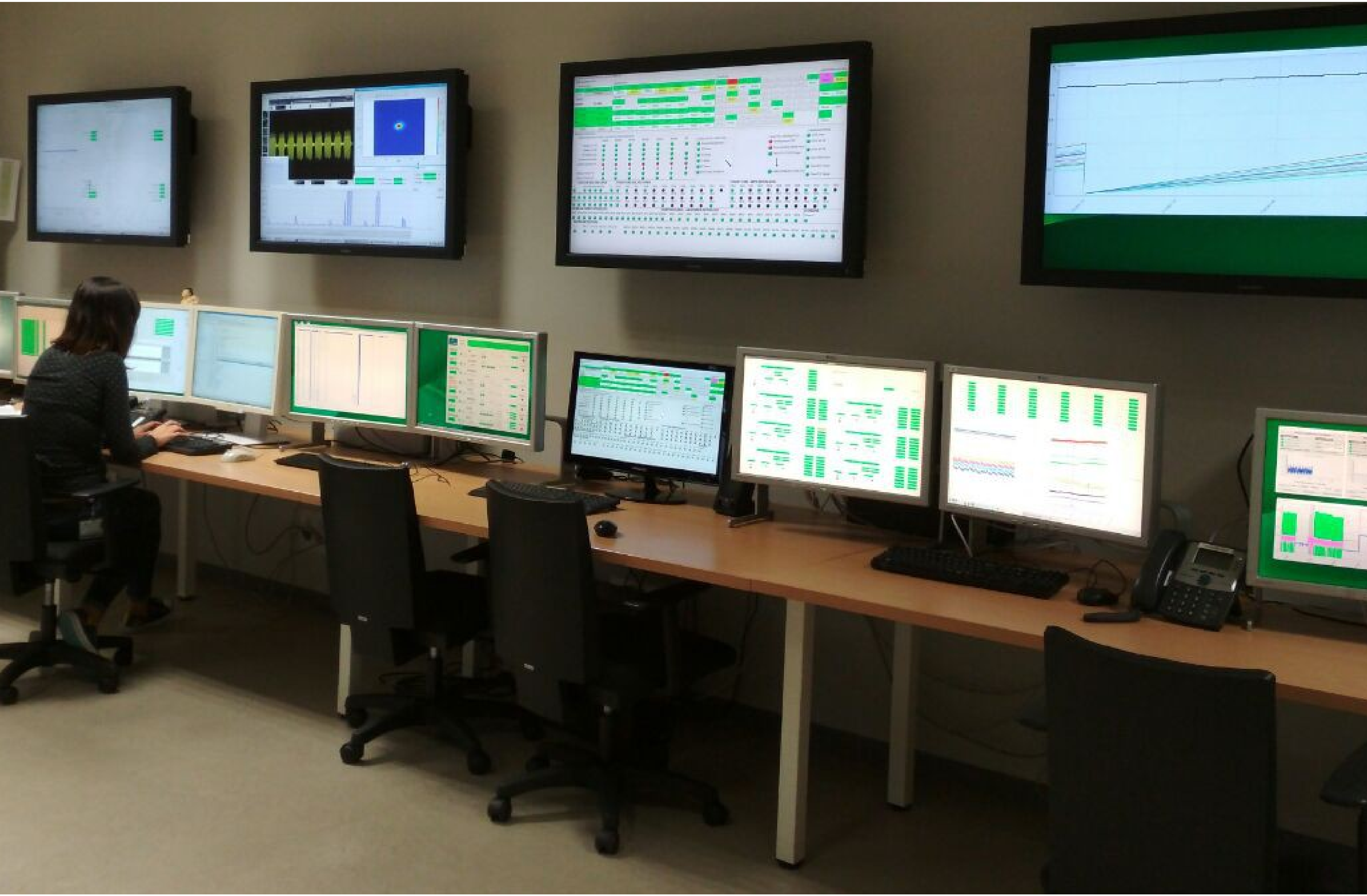
time 10.0 s

but… does taurus scale well?

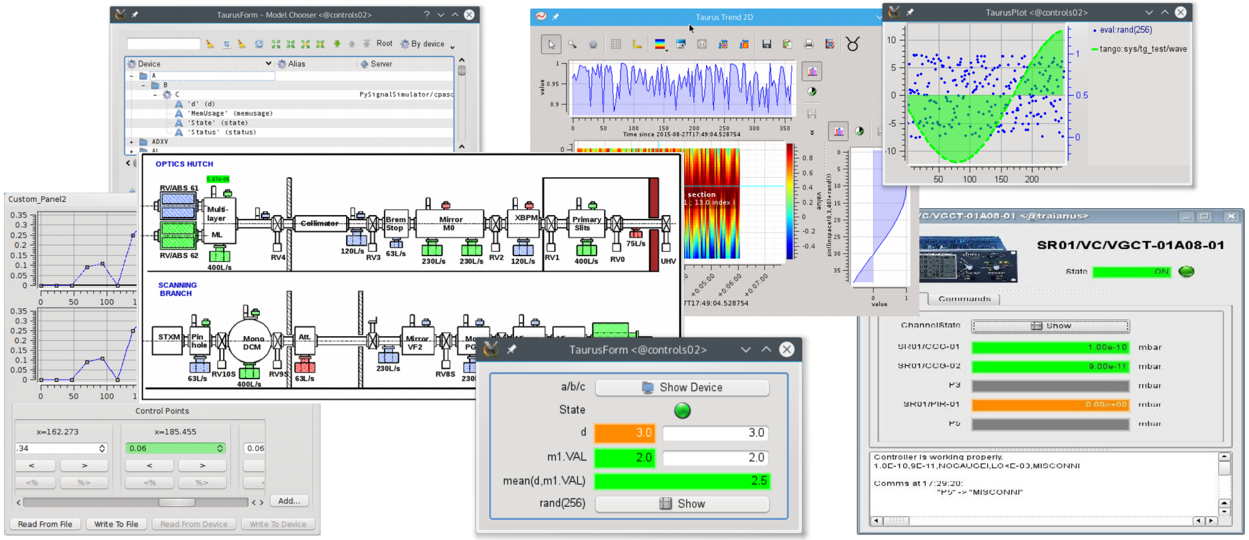


(...both up and down)

Taurus in large facilities



Taurus in large Facilities: example of ALBA



TaurusForm <@controls02>

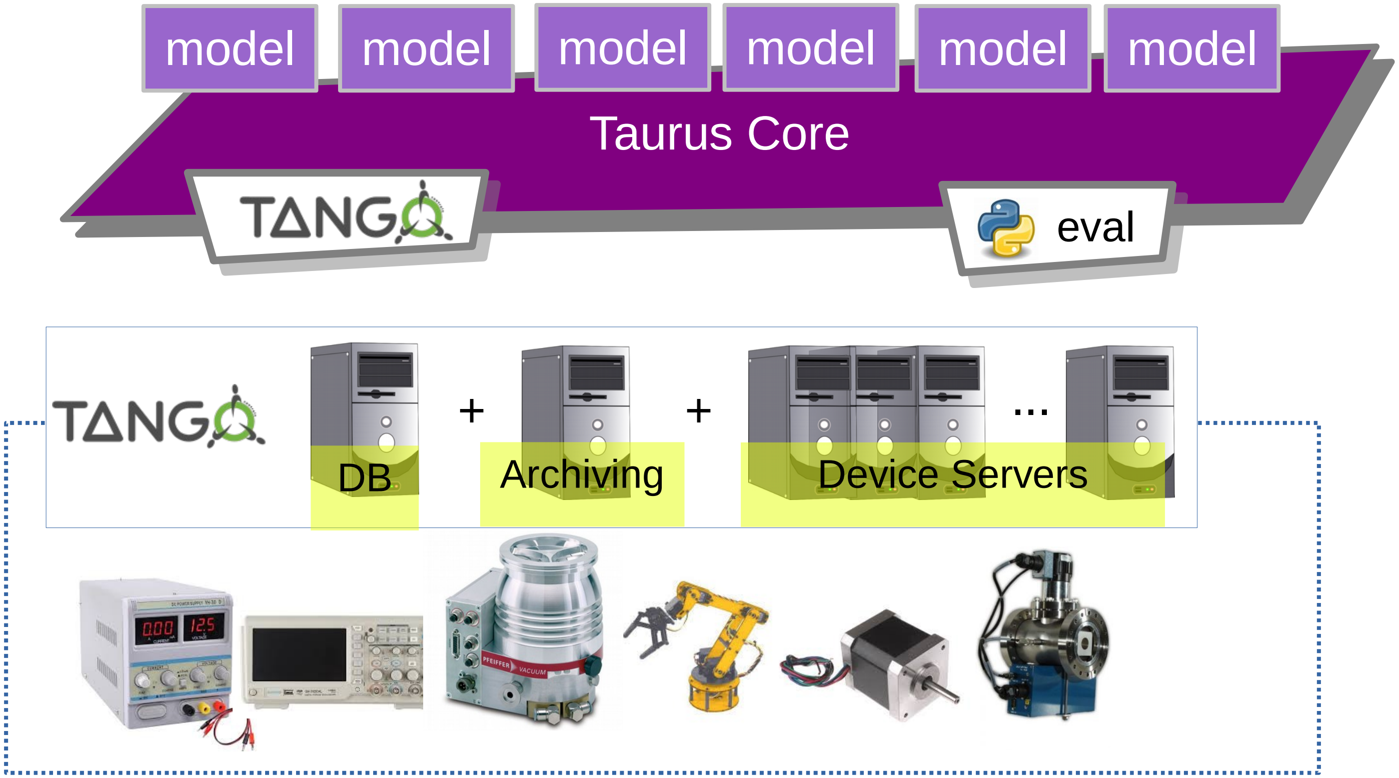
a/b/c	Show Device	
State	●	
d	3.0	3.0
m1.VAL	2.0	2.0
mean(d,m1.VAL)	2.5	
rand(256)	Show	

SR01/VC/VGCT-01A08-01

State ON

ChannelState	Show	
SR01/CCG-01	1.00e-10	mbar
SR01/CCG-02	9.00e-11	mbar
P3		mbar
SR01/PIR-01	0.00e+00	mbar
P5		mbar

```
Controller is working properly.
1.0E-10,9E-11,NOGAUGE,LO<E-03,MISCONN

Comms at 17:29:20:
          "P5" -> "MISCONN"
```



model

model

model

model

model

model

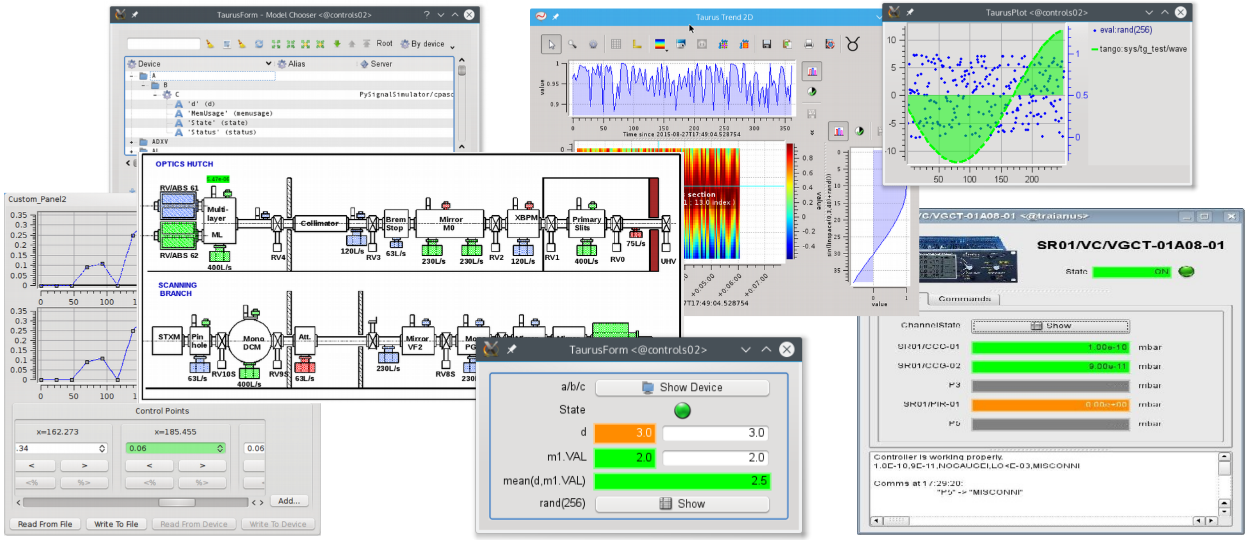
Taurus Core

eval

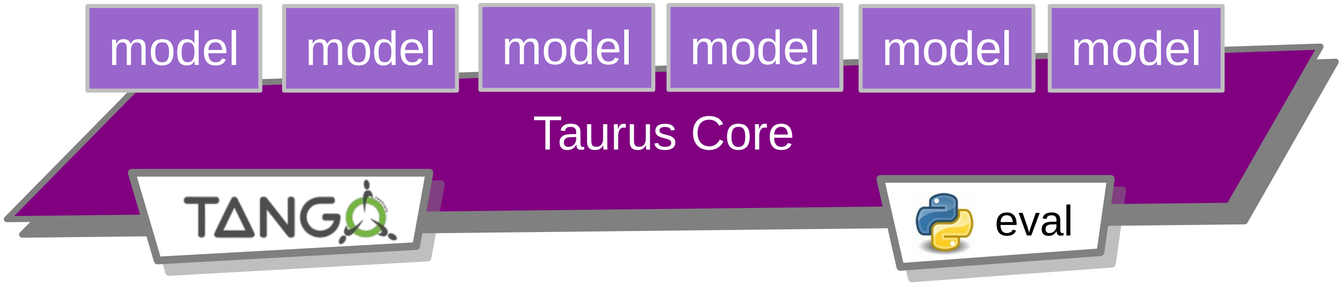
DB + Archiving + Device Servers ...

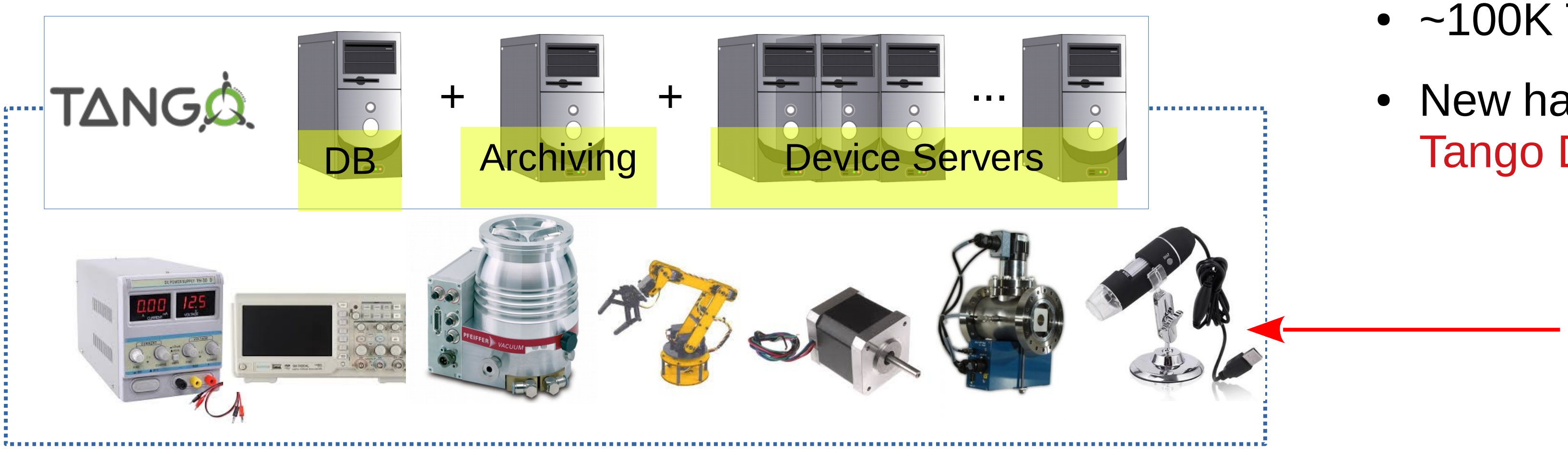
- Both the accelerators and Beamlines are controlled with Tango + Taurus
- ~100 Taurus GUIs
- ~300 machines
- ~10 Tango DBs
- ~100K Tango attributes
- New hardware ?

Taurus in large Facilities: example of ALBA



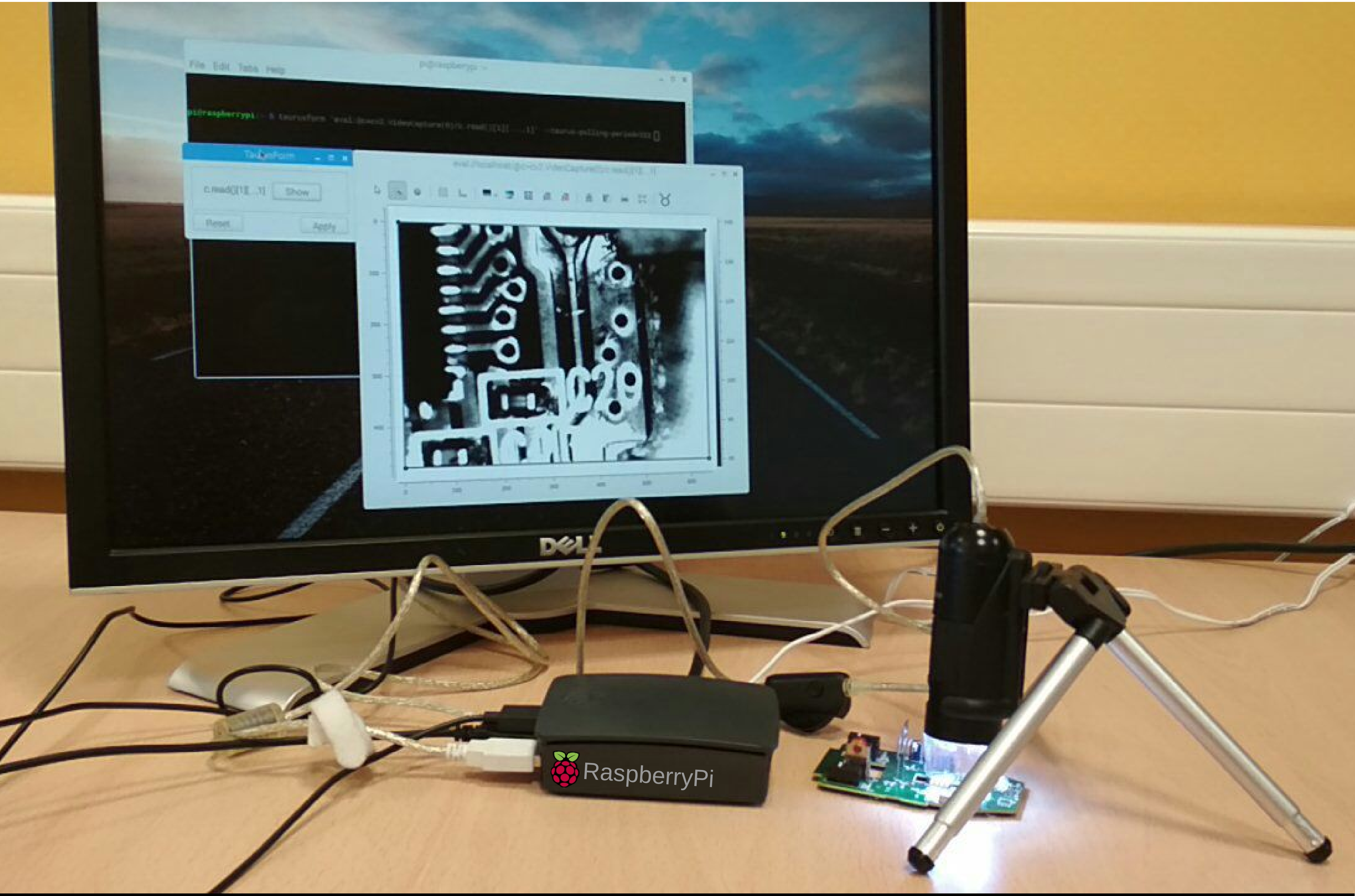
Controller is working properly.
1.0E-10,9E-11,NOGAUGE,LO<E-03,MISCONN
Comms at 17:29:20:
"P5" -> "MISCONN"


model
model
model
model
model
model
Taurus Core
TANGO
eval


TANGO
DB
+
Archiving
+
Device Servers
...

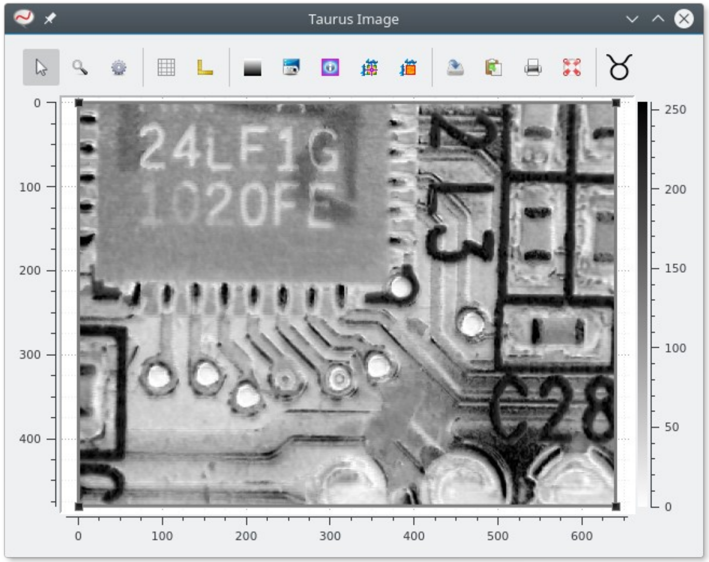
- Both the accelerators and Beamlines are controlled with Tango + Taurus
- ~100 Taurus GUIs
- ~300 machines
- ~10 Tango DBs
- ~100K Tango attributes
- New hardware → write a Tango Device Server

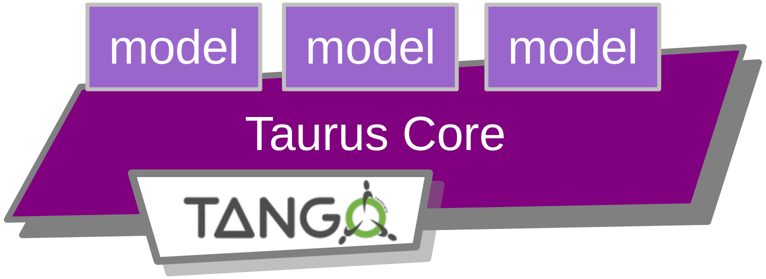
Taurus in “Desktop Labs”:



```
pi@raspberrypi:~ $ taurusform 'eval:@cv2.VideoCapture(0)/c.read()[1][...,1]' --taurus-polling-period=333
```

Taurus in "Desktop Labs"



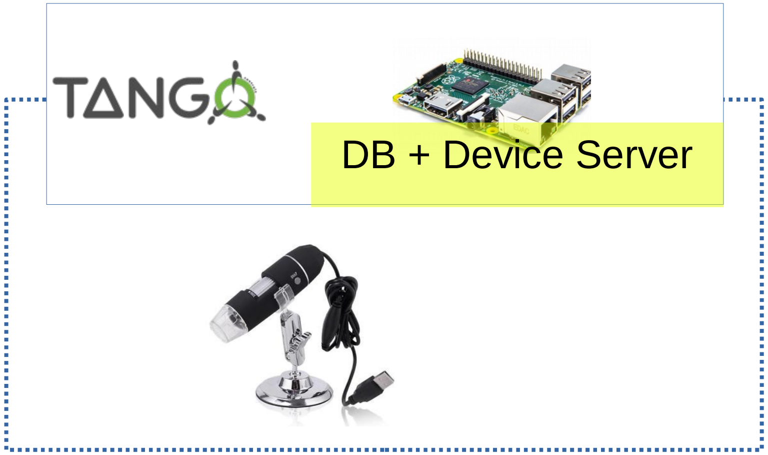

model

model

model

Taurus Core

TANGO


TANGO

DB + Device Server

Approach 1

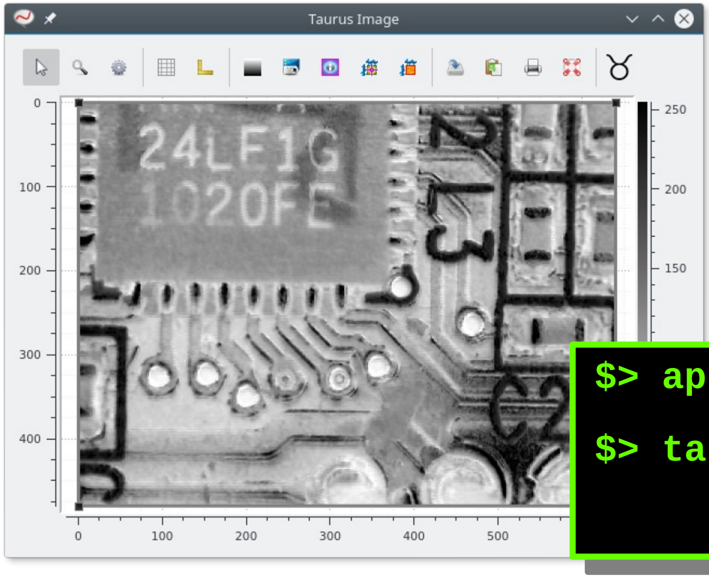
Use same tools as for large facilities (scale-down): use a Distributed Control System on a single machine

```
$ apt-get install default-mysql-server

$ apt-get install tango-db

$ apt-get install python-taurus

$ # write a DS, register it and DB and launch it

$ (…)

$ taurusimage tango:optics/microscope/1/image
```

Recommended **if**:

- you are already familiar with Tango/EPICS
- the hardware is already supported by Tango/EPICS
- you do not mind the communication overhead
- you plan to add more hardware

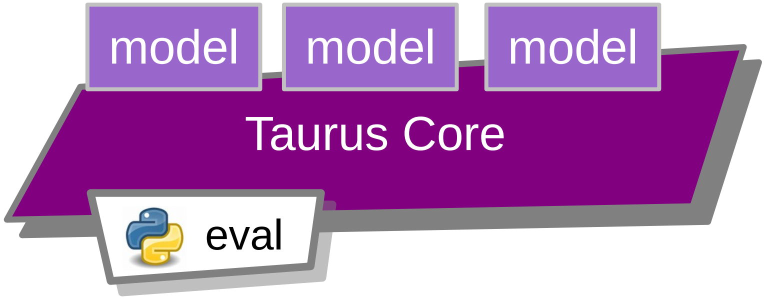
Taurus in “Desktop Labs”



Approach 2

Use the eval scheme and connect directly with the OpenCV module

```
$> apt-get install python-taurus

$> taurusimage 'eval:@c=cv2.VideoCapture(0)/c.read()[1][...,1]'
```



Recommended **for**:

- single-machine systems
- quick prototyping
- quick support of new hardware

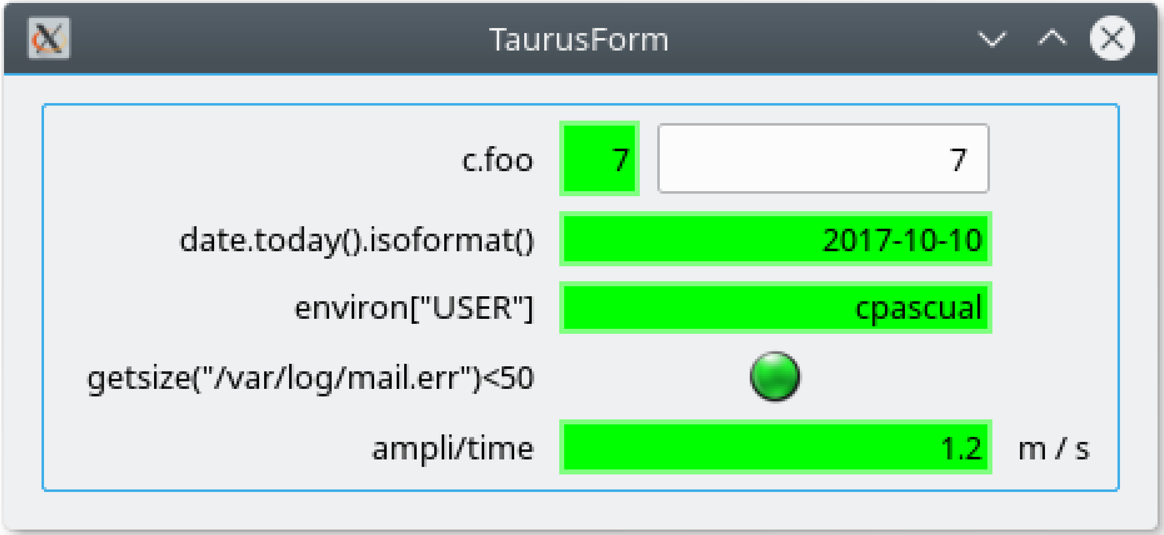
More eval examples...

- Use **any module or class** instance as an “eval device”
- Supports writable eval attributes
- Allows mathematical operations with other attributes

```
$ taurusform 'eval:@c=mymod.MyClass()/c.foo'                        \
             'eval:@datetime.*/date.today().isoformat()'            \
             'eval:@os.*/environ["USER"]'                           \
             'eval:@os.path.*/getsize("/var/log/mail.err")<50'      \

             'eval:{tango:sys/tg_test/1/ampli}/{h5file:/tmp/foo.h5::entry/time}'
```

mymod.py

```
class MyClass(object):

    _foo = 0

    def get_foo(self):
        return self._foo

    def set_foo(self, value):
        self._foo = value

    foo = property(get_foo, set_foo)
```



c.foo 7 7

date.today().isoformat() 2017-10-10

environ["USER"] cpascual

getsize("/var/log/mail.err")<50 ●

ampli/time 1.2 m / s

docs: http://www.taurus-scada.org/devel/api/taurus/core/evaluation.html
mymod example: taurus.core.evaluation.test.res.mymod

Summary

- Taurus is **successfully used in many large facilities**
- When used with a Distributed Control System (DCS), **it scales as well as the DCS**
- A full DCS + Taurus system can be **run on a single RaspberryPi**
- Taurus **can be used without a DCS** (using custom schemes and/or “eval”)
- The “eval” scheme is great for **prototyping** and quick integration of data sources

MORE INFO https://taurus-scada.org