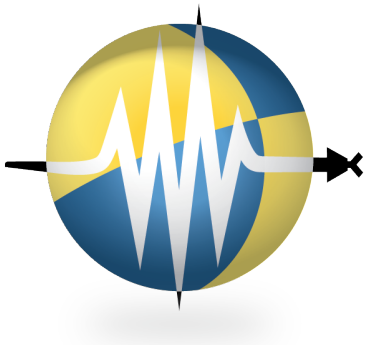
Python and ObsPy
an opportunity to drive innovation



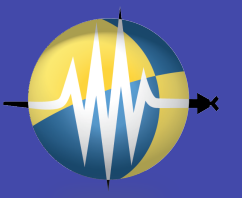
ObsPy and SeisComP workshop
Bucharest, 24/09/2015

Overview

- From my perspective:
  Some issues which currently influence new developments and innovation
- How Python does fit there

- Massive seismic data analysis
  A very preliminary preview of a tool leveraging on Python and ObsPy

- *Demo (maybe!)*



Perspective

Background:

- Coming from research in seismic wave propagation: methods and application mainly to regional scales and the studies of particular 3D propagation effects
- Recently more involve into data analysis and processing of substantial amounts of data
- Involvement with projects and initiatives dealing with infrastructure and service building
- Our group provides the infrastructure for the 24h / 7day surveillance and related services

My role is somewhere at the interface between research, research requirements and infrastructure and services building. I am to some extend concerned with the integration of infrastructures and resources for research, e.g. data archives and services, computational resources or operational systems.

Specialization

- Larger organization like INGV can structure work, task and responsibilities
- Specialization of personnel allows to develop specific skills and approaches

Our Group:

- Develops and maintains the infrastructure and services for the 24h / 7day surveillance service, both for seismic monitoring and tsunami alerts
- This includes real-time data acquisition and automatic processing
- Interactive tools systems for surveillance operators and technical personnel
- Services for information dissemination and formal communication services
- Services to data distribution to the scientific community

Drawback: This separation of work and thinking in categories, however, to some extent may lead to artificial barriers, due to different practices, approaches.

The different approaches

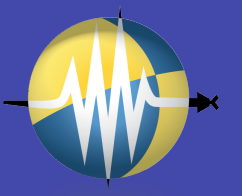
Arbitrary example of different working styles

Scientists

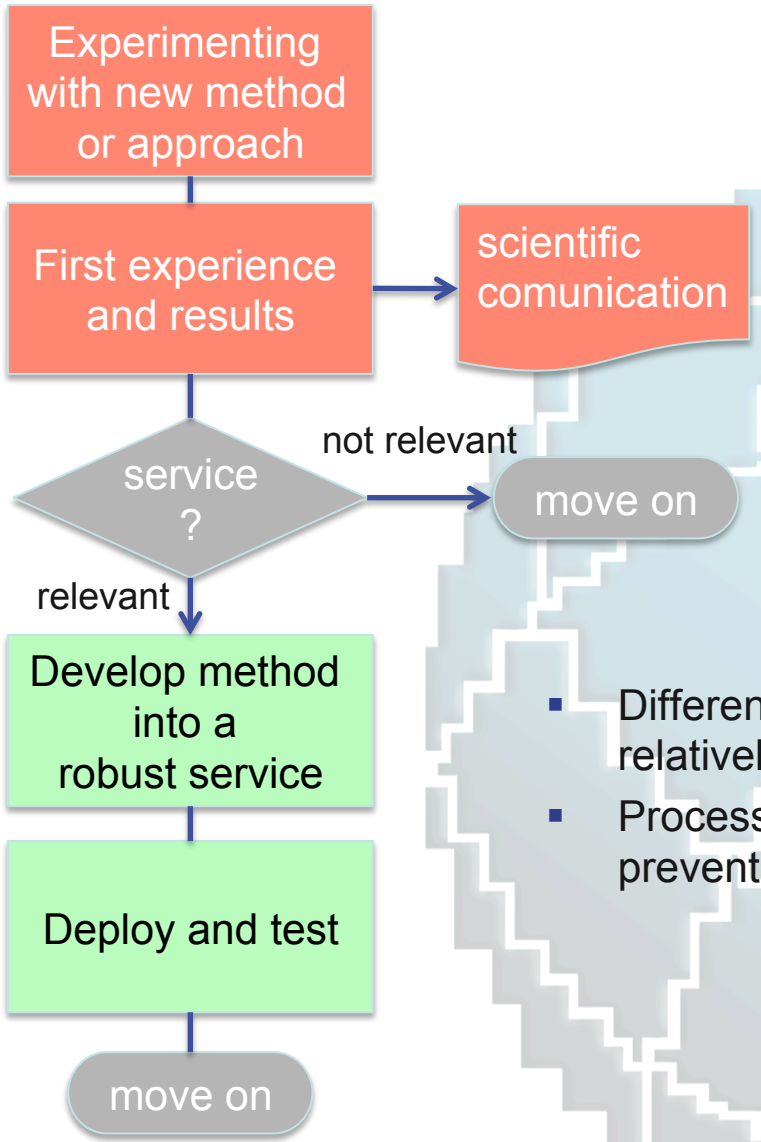
- Tools to explore data interactively
- Fast prototyping and simple scripting
- Data viewers
- Integrated interactive environments, like Matlab
- Ad-hoc Unix shell scripting with Unix and community tools
- Fortran
- Prefer no explicit compilation and languages dynamic typing

IT staff

- Reliable operation
- Consistent and traceable operation over long periods
- Integrated development environments and SDK
- Prefer statically typed languages or other modes of verification
- Typical languages C, C++, Java, PHP, SQL
- Involves database systems, messaging systems, Network protocols



Putting new methods and procedures in operation



- Different approaches become an obstacle for relatively smooth uptake of innovative services
- Process becomes resource consuming and may prevent the implementation

Why Python might help?

- Python could become the common language for different roles, notably scientists and IT staff
- Usage at different levels of sophistication
- Potentially satisfies the working styles and requirements of both groups

Scientists

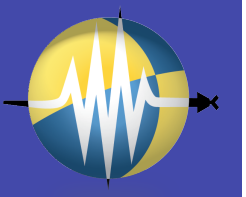
- Relatively easy to learn, read and understand
- Simple tasks are easy to formulate
- Interactive environment (Ipython, Jupyter) to *explore* problem and solution
- No compilation, immediate testing, dynamic data types
- Access to a vast scientific computing ecosystem

IT staff

- Well-designed programing language with advanced features
- Huge user base, no domain-specific solution
- Code is easy to share, develop and maintain by larger group
- Good support for web, internet protocol, databases, data formats, messaging, graphical interfaces etc.
- Integrates with other languages and components

Benefits for scientists

- Relatively easy access to modern tools and protocols
- **No licensing issues!**
- Share and reuse code and modules (e.g. shared by IT staff)
- If we take reproducibility of scientific results serious, any hand-make magic is probably not the best approach
- Working with growing amounts of data will require a more systematic and traceable approach
- Processing as a service or moving processing to data, instead of moving data (How we would this work?)



Interactive exploration of large datasets

qualitycheck

localhost:8888/notebooks/qualitycheck.ipynb

jupyter qualitycheck

File Edit View Insert Cell Kernel Help Python 3

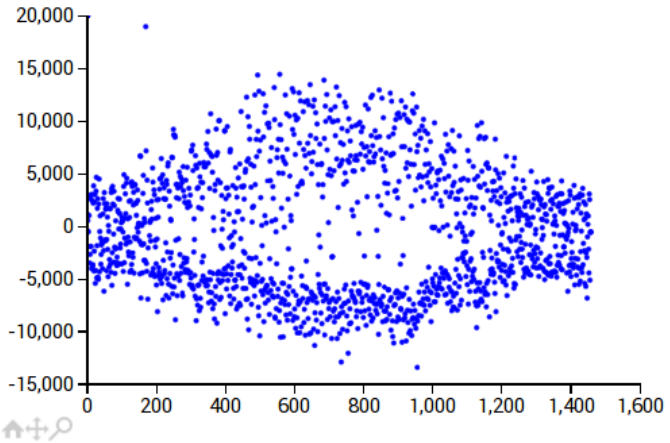
Code Cell Toolbar: None

In [16]:
```
# Here we plot, for each time window, the arithmetic
# mean of the samples.

means = rdd_tr.map(lambda tr: tr.data.mean()).collect()

#plt.figure(figsize=(10,5))
plt.plot(means, '.')
```

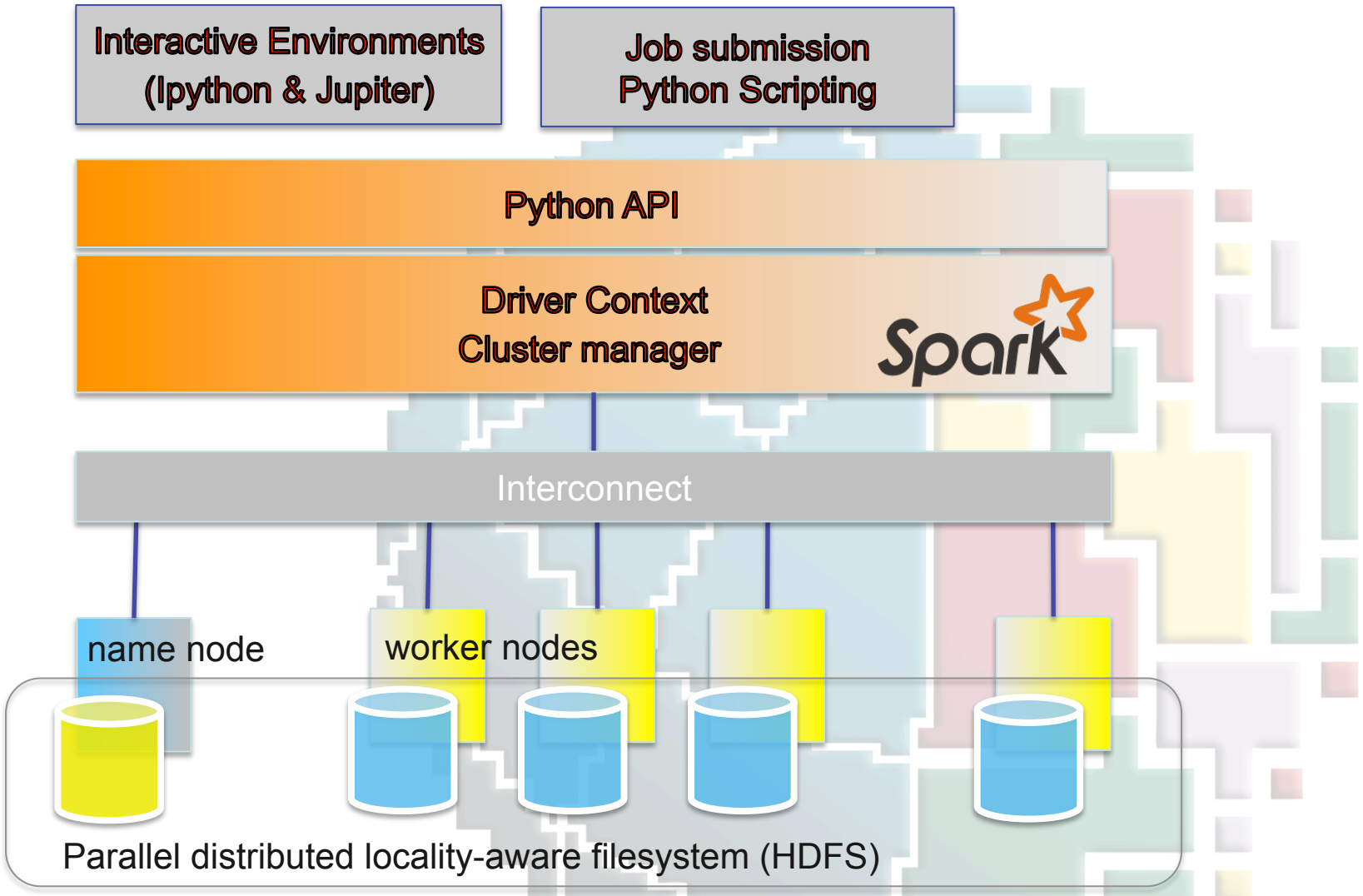
Out[16]: [<matplotlib.lines.Line2D at 0x7f69417ecd10>]



Power by

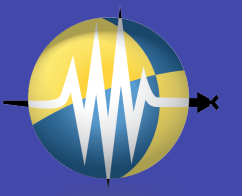
- Jupyter
- Python ecosystem
- ObsPy
- Spark

SeiSpark prototype

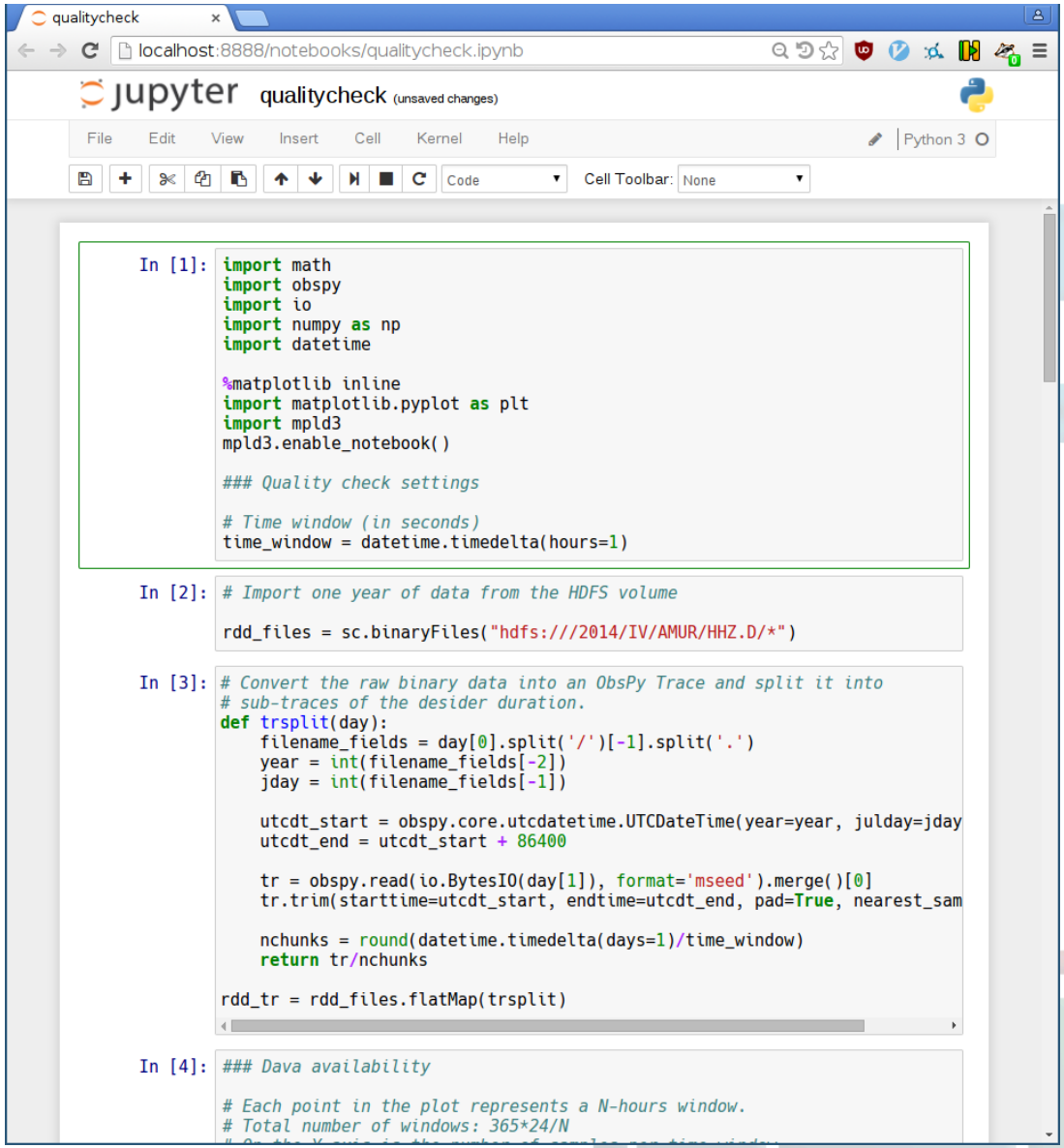


Processing framework

- The Spark framework introduces an abstraction for distributed memory, called Resilient Distributed Dataset (RDD)
- Leverages on the (large) main memory resources and allows in-memory computation and caching
- Appears sufficiently general purpose to allow the efficient implementations of several processing patterns and not a single application
- Allows for interactive use
- Provides good performance
- Lazy evaluation
- Nice APIs and in particular it provides an Python API
- Transparent access to binary files
- Build on top of the Hadoop filesystem
- Can be used with several data sources



Getting started



Example: data completeness

qualitycheck

localhost:8888/notebooks/qualitycheck.ipynb

jupyter qualitycheck

File Edit View Insert Cell Kernel Help Python 3

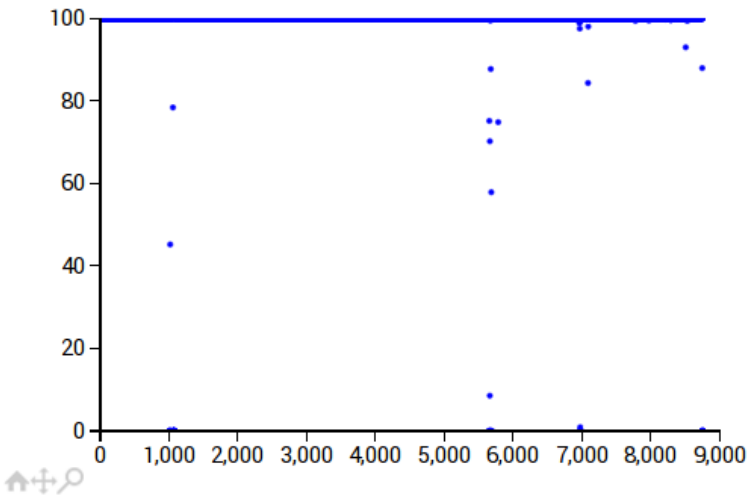
Code Cell Toolbar: None

In [13]:
```
# The same as above, but in percentage.

data_presence = rdd_tr.map(lambda tr:
                           np.ma.array(tr.data).count()/tr.data.size*100
                          ).collect()

#plt.figure(figsize=(10,5))
plt.plot(data_presence, '.')
```

Out[13]: [<matplotlib.lines.Line2D at 0x7f69417f3910>]



Example: Root mean square

qualitycheck

localhost:8888/notebooks/qualitycheck.ipynb

jupyter qualitycheck

File Edit View Insert Cell Kernel Help Python 3

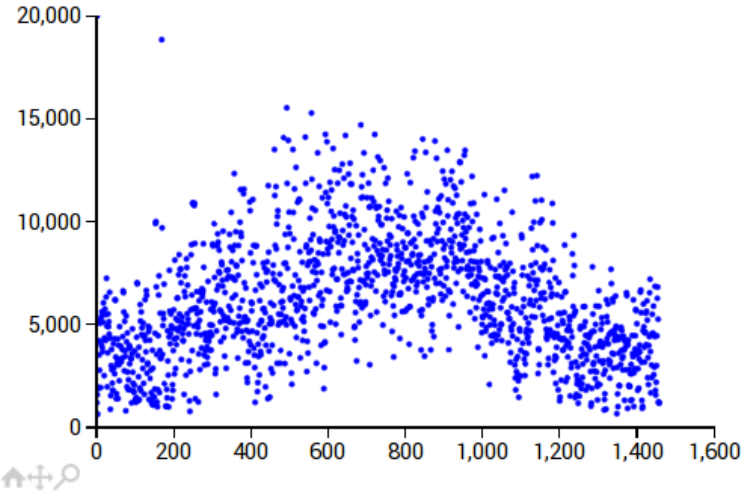
Code Cell Toolbar: None

In [18]:
```
# Root Mean Square

rms = rdd_tr.map(lambda tr:
                 math.sqrt(np.square(tr.data).mean())
                ).collect()

#plt.figure(figsize=(10,5))
plt.plot(rms, '.')
```

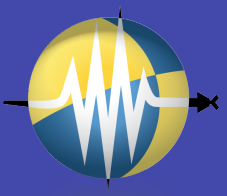
Out[18]: [<matplotlib.lines.Line2D at 0x7f6941792a90>]



Key statements – Comments

- Python represents an opportunity speed up the innovation cycle in the seismology community, both from a pure research point of view, as well as for operational services
- It could facilitates the exchange of resources and ideas between purely scientific activities and staff providing operational services
- ObsPy is the bridge for the seismology community into the vast Python ecosystem
- Seismologists can relatively easily profit from products and effort of a much larger community of programmers and scientists
- It can be relatively easy integrated into more complex frameworks

- To come up with more generic processing services at data centers will need some way to express the desired process. Could this be a Python API?



This is the end …

Thank you!