Pydron

Semi-Automatic Parallelization for Astronomy Data Processing

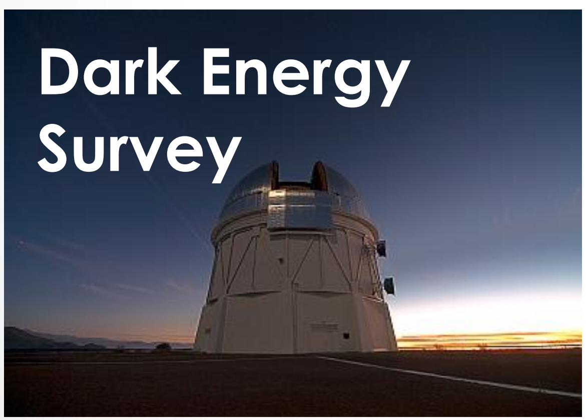
Stefan C. Müller
Gustavo Alonso
Adam Amara
André Csillaghy

Gustavo Alonso
Systems Group ETHZ

Adam Amara
Institute of Astronomy ETHZ

André Csillaghy
Institute of 4D Technologies FHNW

Stefan C. Müller
Systems Group ETHZ
Institute of 4D Technologies FHNW



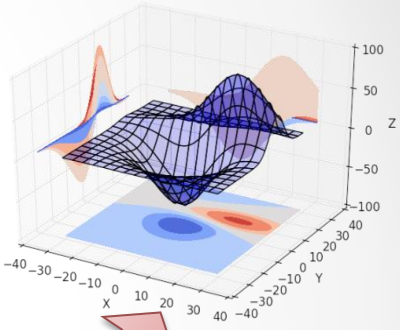
RHESSI: NASA/Goddard Space Flight Center Conceptual Image Lab
Solar Orbiter: ESA / AOES
Dark Energy Survey: T. Abbott and NOAO/AURA/NSF
Euclid: ESA/C. Carreau

Overview

Astronomy Processing Codes

```
def main(raw_images):
  calibrated_files = []
  for raw_image in raw_images:
    bias = measure_bias(raw_image)
    ft= measure_flat(raw_image)
    cal_file = calib(raw_image,bias,ft)
    calibrated_files.append(cal_file)
```

Python



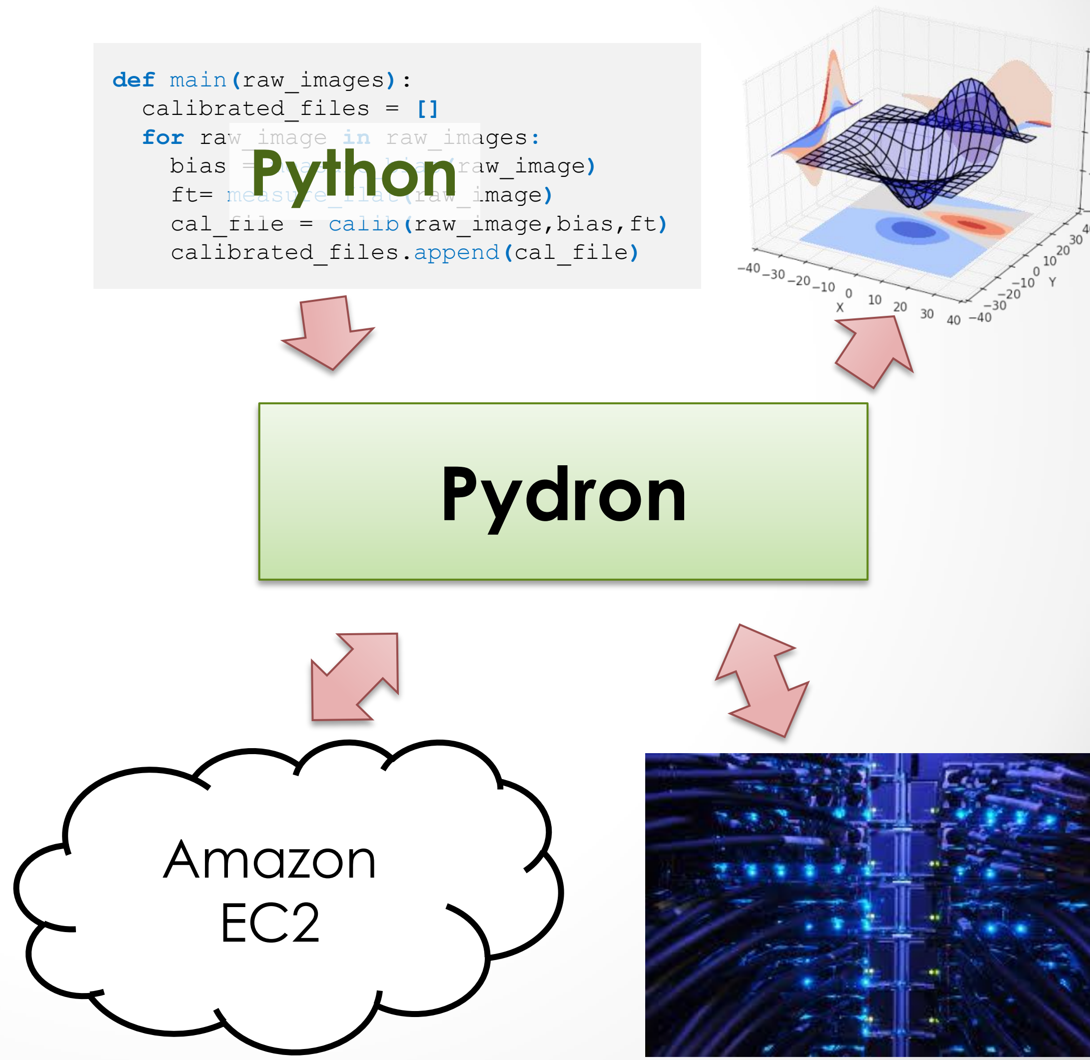
Semi-Automatic Parallelization

Pydron

Cloud, Cluster, Multicore

Outline

1. Astronomy Data Processing
2. Using Pydron
3. How it works
4. Results



Why is Astronomy Data Processing Different?

…

Lack of Economies of Scale

of Users

Commercial Systems:

- Thousands of deployments
- Millions of users

Astronomy Projects:

- Single Deployment
- Hundreds of Researchers
- < 10 that run jobs

One of a Kind

Innovative

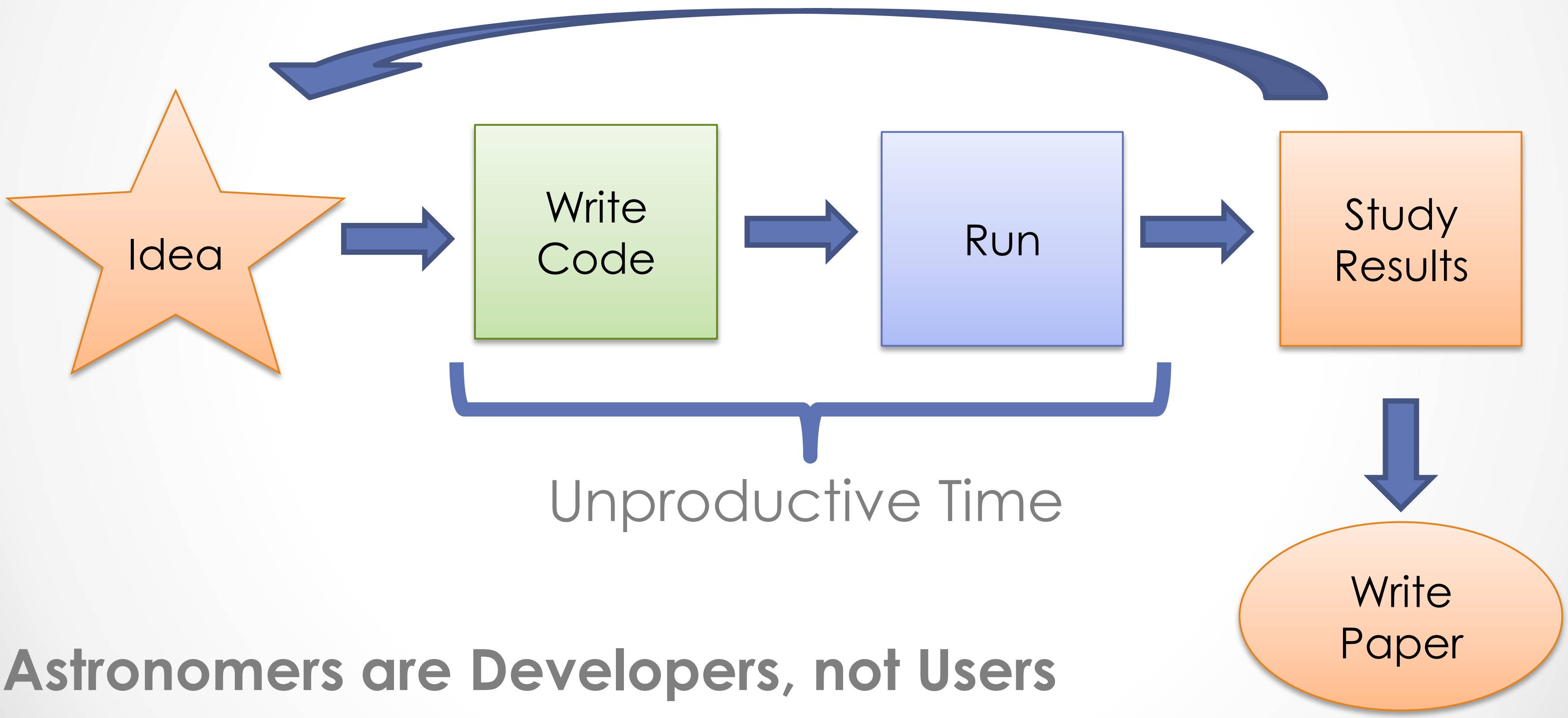
- Instrument
- Experiment
- …

Never seen before ..

- Kind of data
- Data volume
- Data Processing
- Analysis
- …

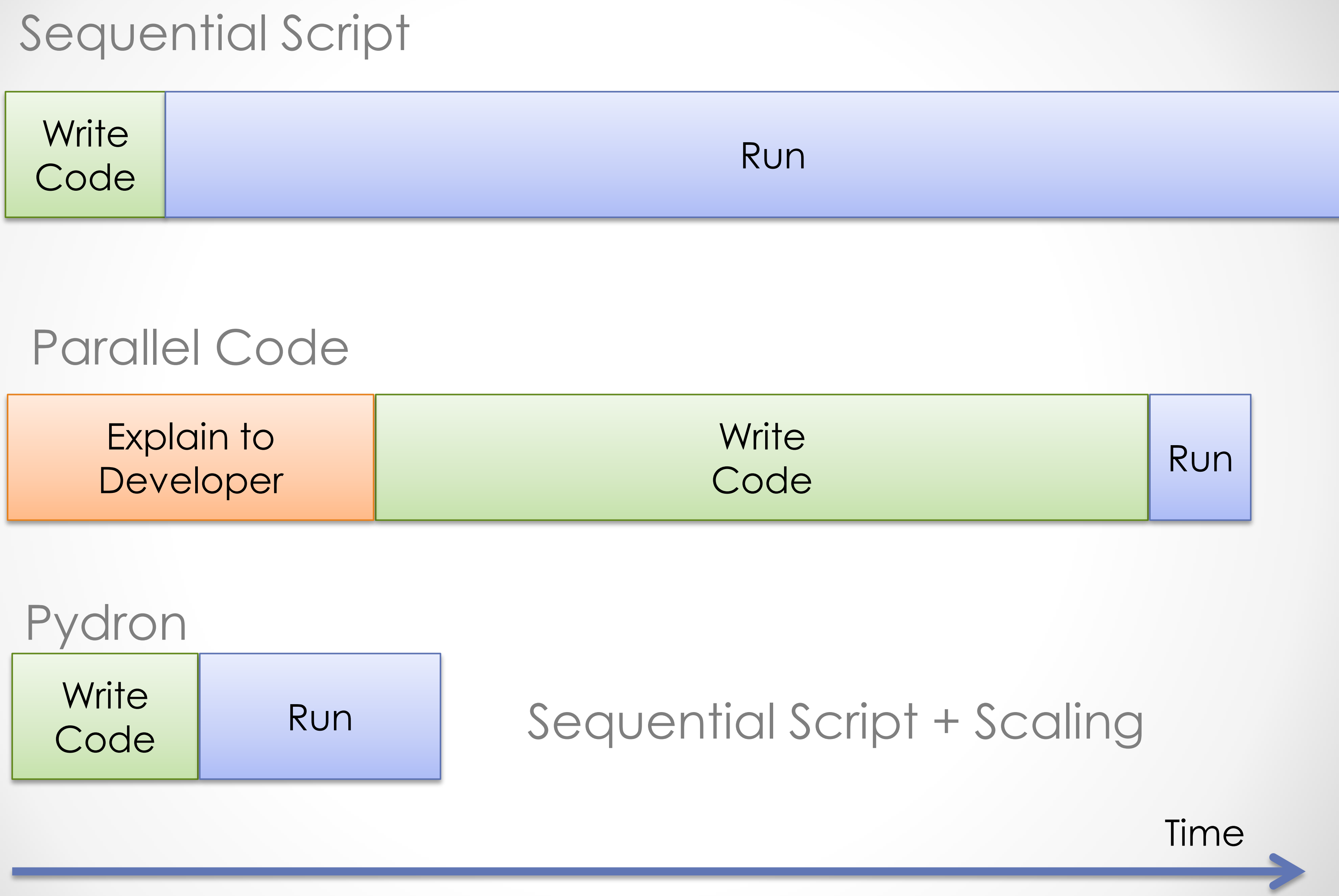
Code Reusability is very limited

Code changes constantly



Astronomers are Developers, not Users

never run again


Sequential Script
Write Code
Run
Parallel Code
Explain to Developer
Write Code
Run
Pydron
Write Code
Run
Sequential Script + Scaling
Time

How is Pydron used?

Sequential Python Code

```python
def main(raw_images):
  calibrated_files = []
  for raw_image in raw_images:
    bias = measure_bias(raw_image)
    flat = measure_flat(raw_image)
    cal_file = calibrate(raw_image,bias,flat)
    calibrated_files.append(cal_file)


def measure_bias(raw_image):
  ...


def measure_flat(raw_image):
  ...


def calibrate(raw_image,bias,bg):
  ...
```

Look for parallelism in here

Minimal code changes required

no side-effects

```
@schedule
def main(raw_images):
    calibrated_files = []
    for raw_image in raw_images:
        bias = measure_bias(raw_image)
        flat = measure_flat(raw_image)
        cal_file = calibrate(raw_image,bias,flat)
        calibrated_files.append(cal_file)

@functional
def measure_bias(raw_image):
    ...

@functional
def measure_flat(raw_image):
    ...

@functional
def calibrate(raw_image,bias,bg):
    ...
```

Just run it

Once the **@schedule** function is invoked, Pydron will ...

For the cloud:

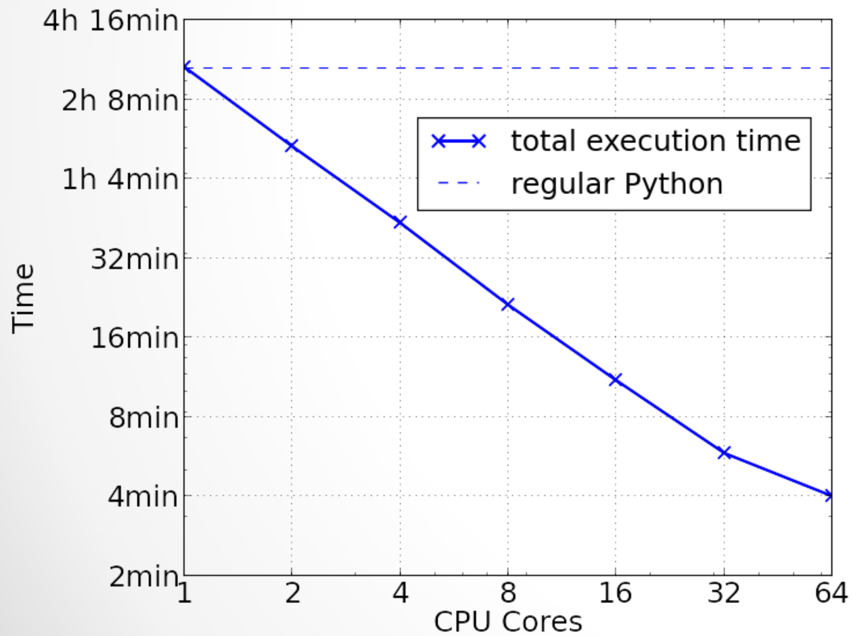
- Start cloud nodes & Start Python processes
- Transfer code (and libraries) to nodes
- Schedule tasks
- Send results back to workstation
- Shutdown nodes

The **@schedule** function returns with the result.

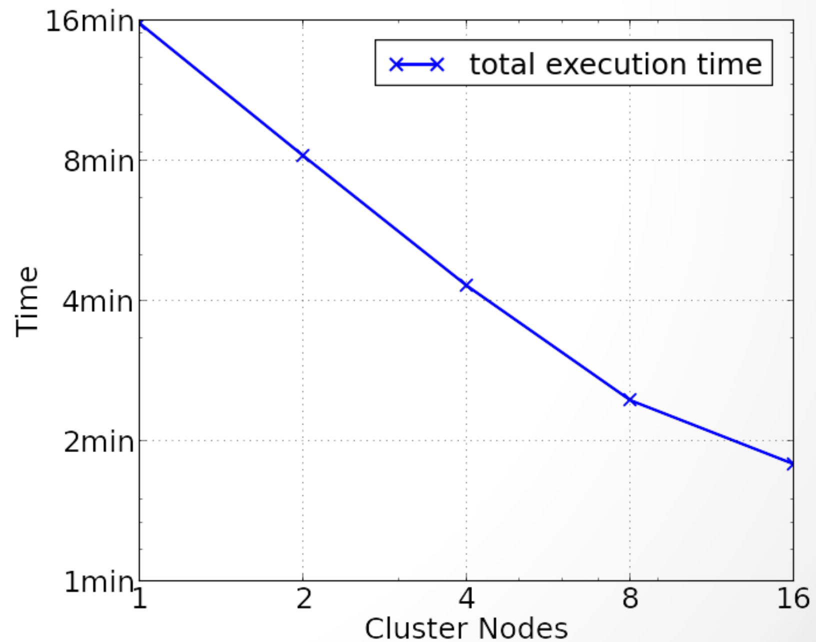
Machine Learning

Random Forest Training

Uses scipy native library



Multi-Core



Cluster

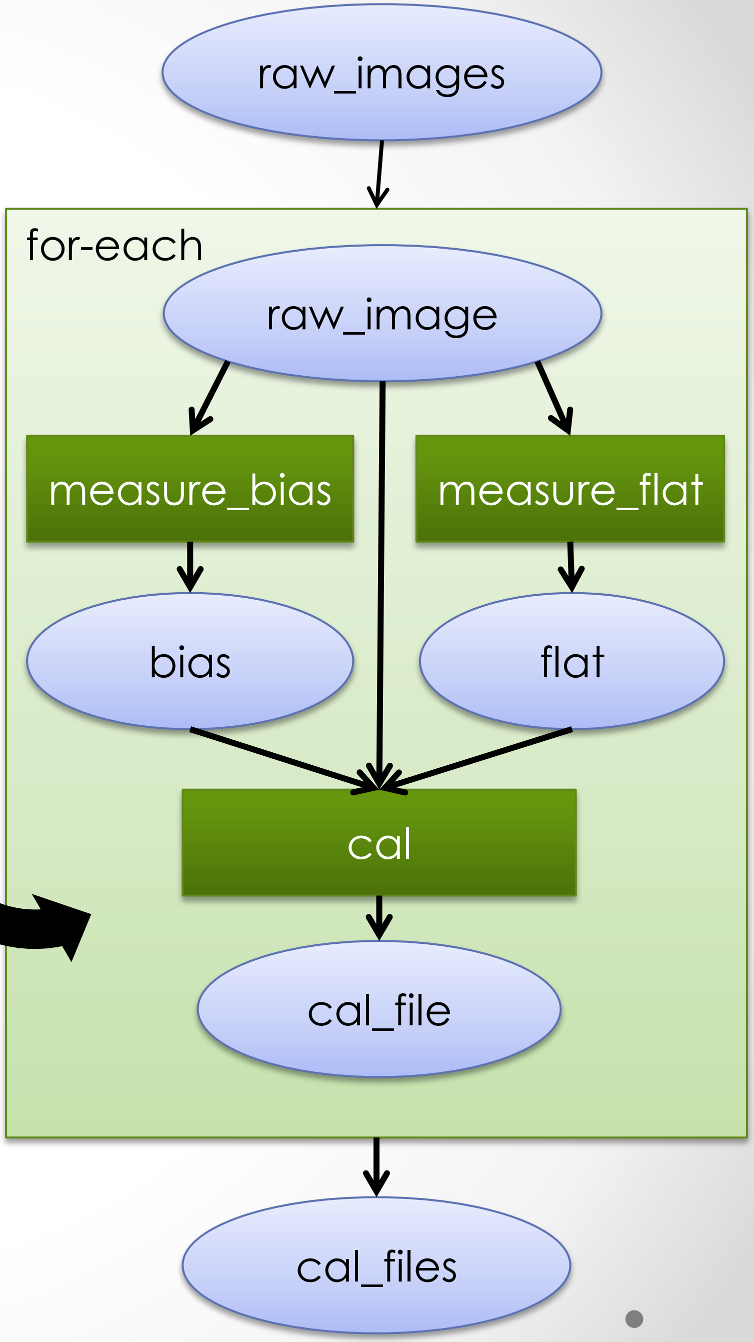
How does it work?

Data-flow Graph

```
@schedule
def main(raw_images):
  cal_files = []
  for raw_image in raw_images:
    bias = measure_bias(raw_image)
    ft = measure_flat(raw_image)
    cal_file = cal(raw_image,bias,ft)
    cal_files.append(cal_file)
```

- Expressions →Task nodes
- Variables → Value nodes

- Static Single Assignment Form
- Sub-Graphs for compounds



Python Challenges

- Data-dependent control-flow
- No static type information
- Invoked functions unknown
- Decorators unknown

Adapt Graph with Runtime Information

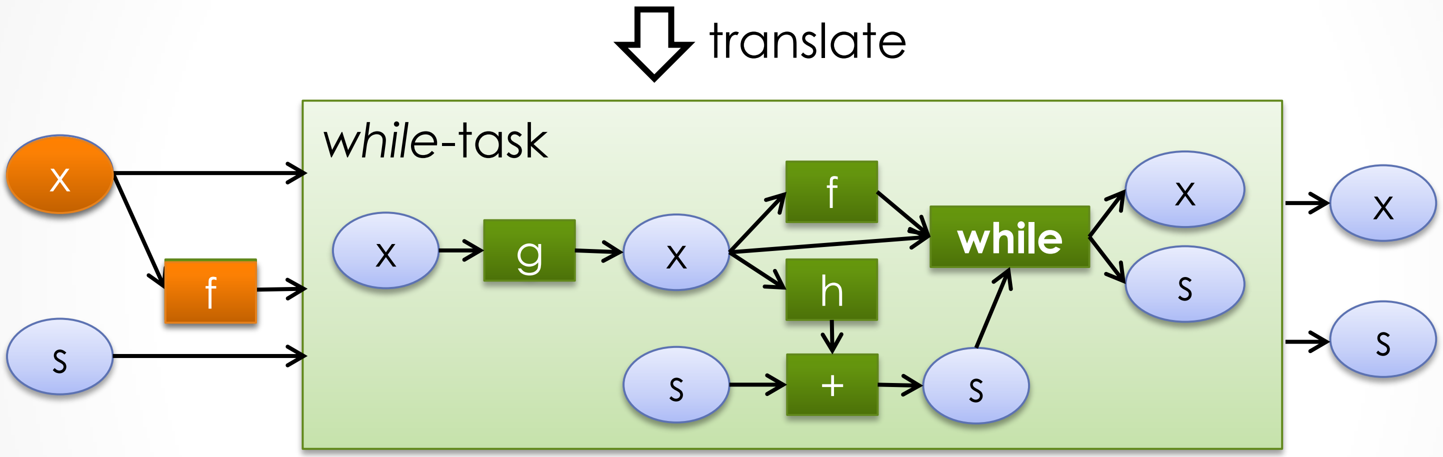
- Change graph depending on data
- Use dynamic type information
- Detect function decorators
- Dynamically adapt granularity

Data dependent Control-Flow

```
while f(x):
    x = g(x)
    s += h(x)
```

Data dependent Control-Flow

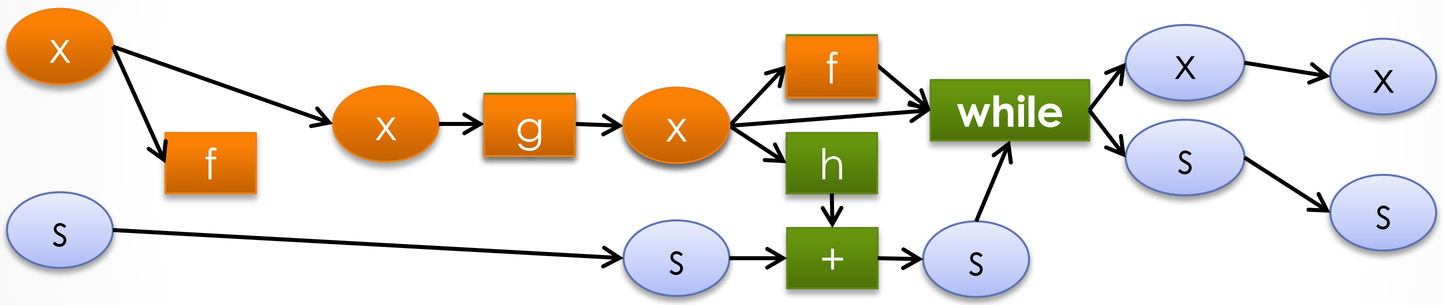
```
while f(x):
    x = g(x)
    s += h(x)
```



Data dependent Control-Flow

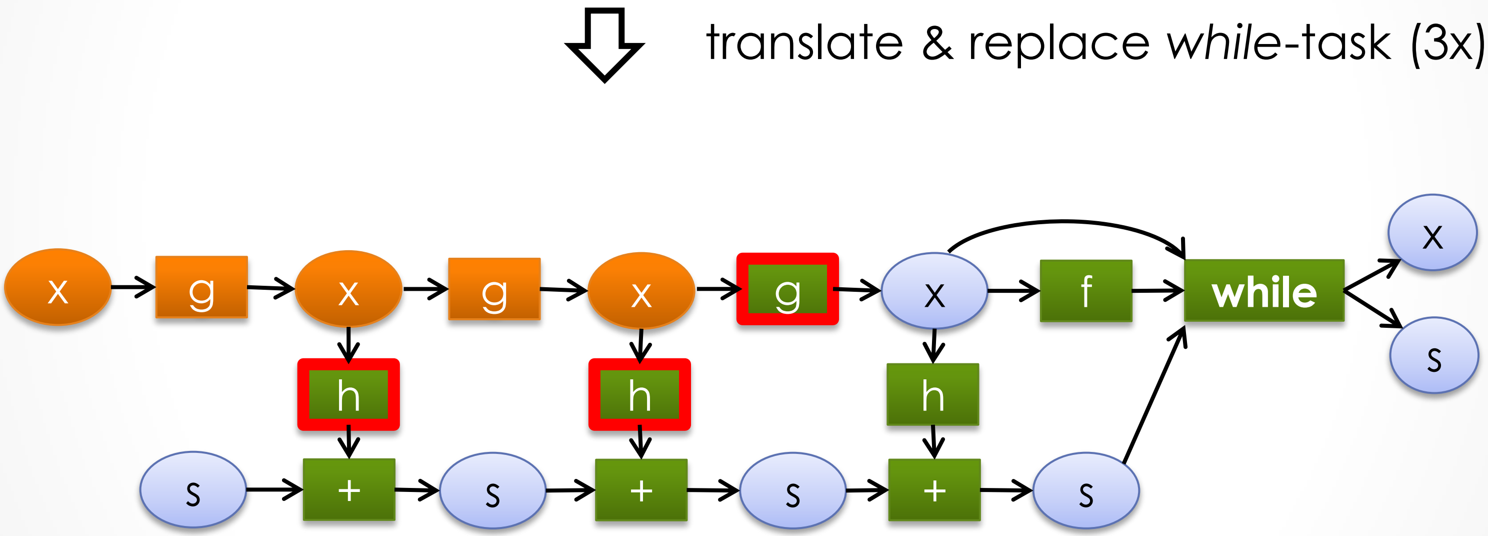
```
while f(x):
    x = g(x)
    s += h(x)
```

translate



Data dependent Control-Flow

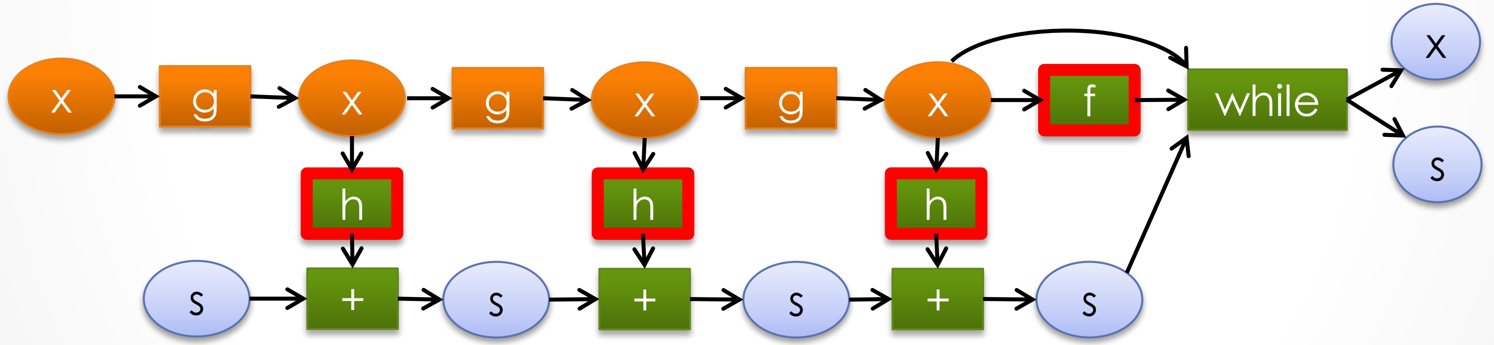
```
while f(x):
    x = g(x)
    s += h(x)
```



Data dependent Control-Flow

```
while f(x):
    x = g(x)
    s += h(x)
```

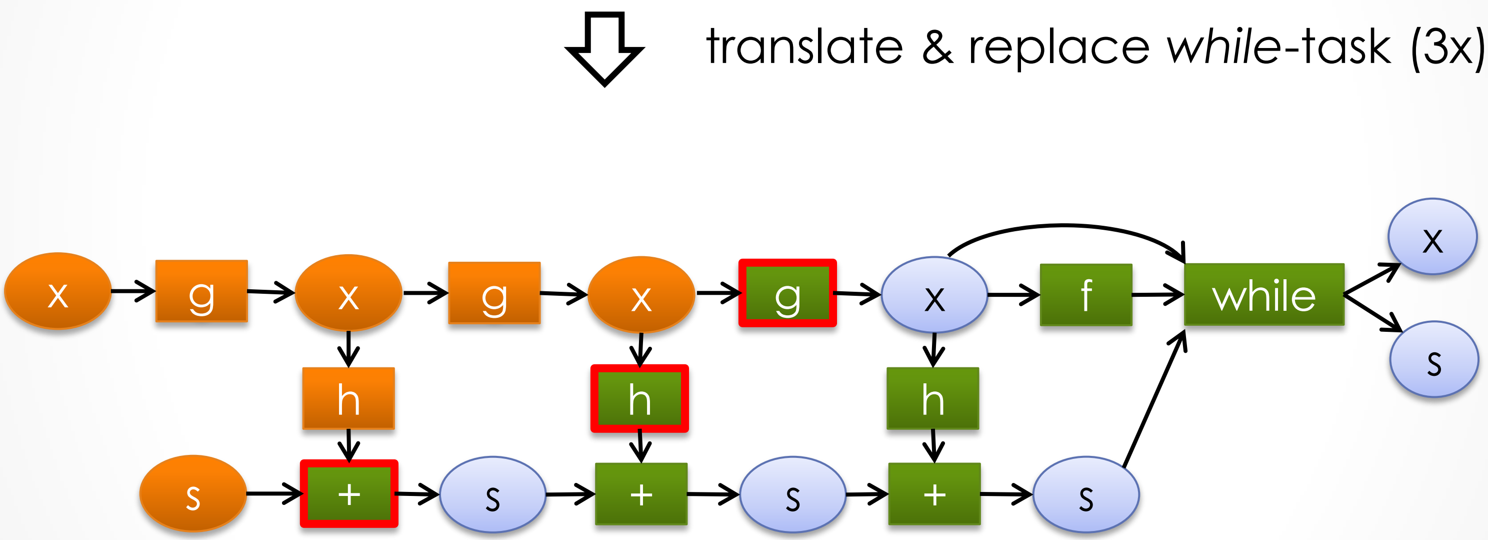
⇩ translate & replace *while*-task (3x)



Data dependent Control-Flow

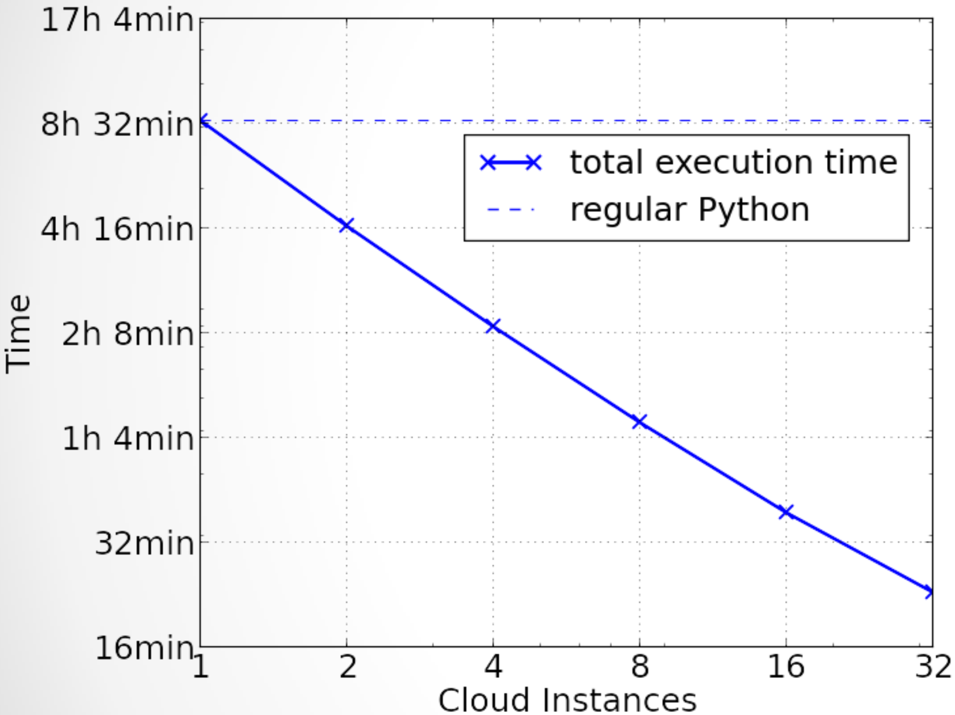
```python
while f(x):
    x = g(x)
    s += h(x)
```

translate & replace *while*-task (3x)

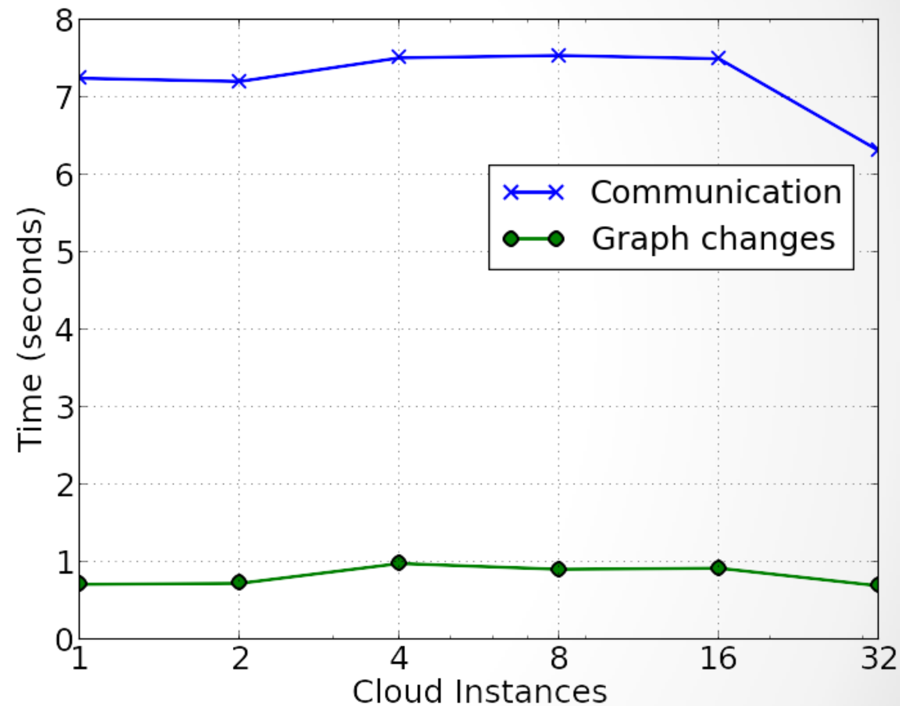


Results

Exo-Planet Detection



Execution Time



Overhead

Future Optimizations

- Scheduling algorithm
- Data specific optimizations
- Pre-fetching
- Dynamic Resource Allocation

Conclusions

Pydron lowers the boundary to use parallel infrastructure

Non-intrusiveness & low learning curve over ideal performance

Q & A

• • •

www.pydron.org