

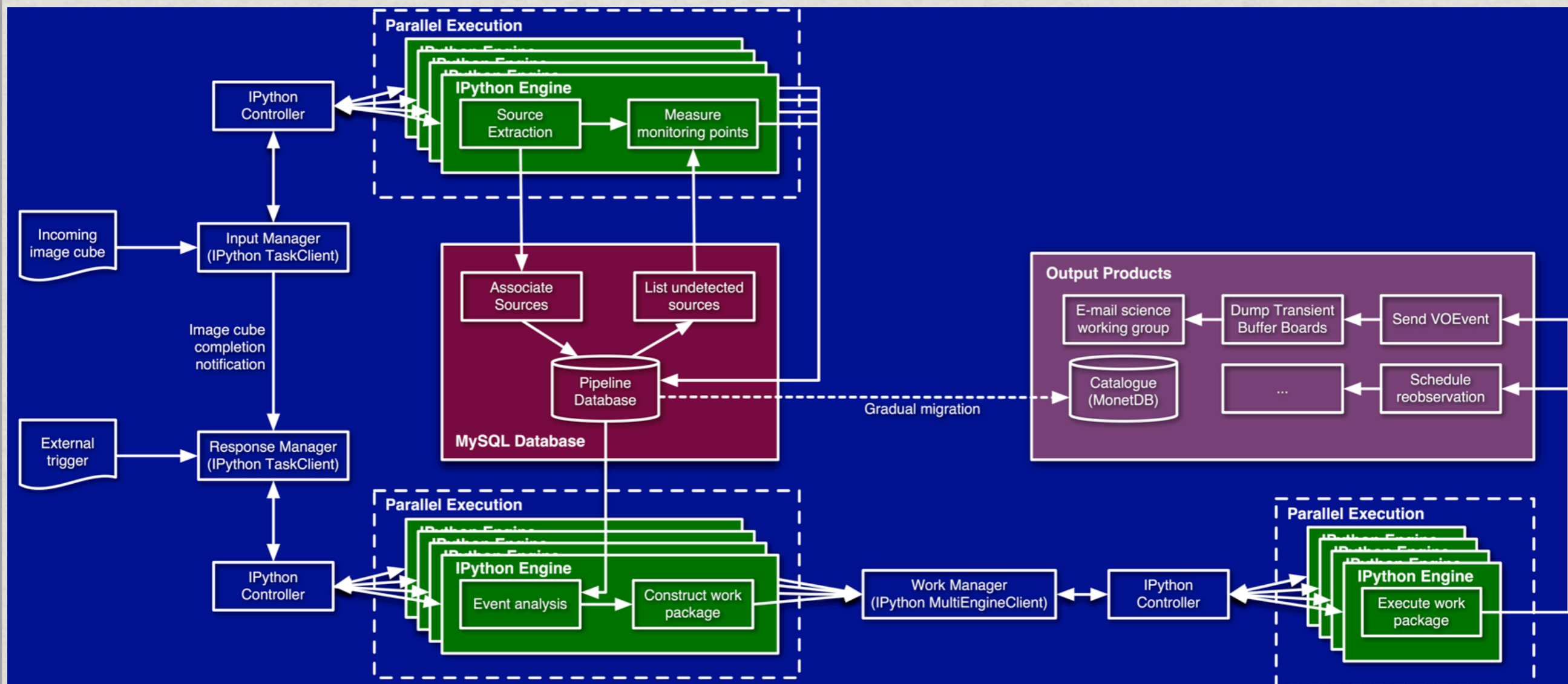
A transient detection pipeline for LOFAR

John Swinbank, University of Amsterdam
swinbank@transientskp.org

Overview

- * What is the “TKP pipeline”?
- * Highlight components
 - * Source extraction
 - * Databases
 - * Response system
 - * Management & parallelization

Bird's-eye view



Development

- * Python (2.5/2.6)
- * A little C++
- * Extensive use of external libraries:
 - * NumPy, SciPy, wcslib, wcstools, IPython, Boost.Python, Twisted, Foolscape, Fabric, etc
- * Developed/tested on Linux & Mac OS X

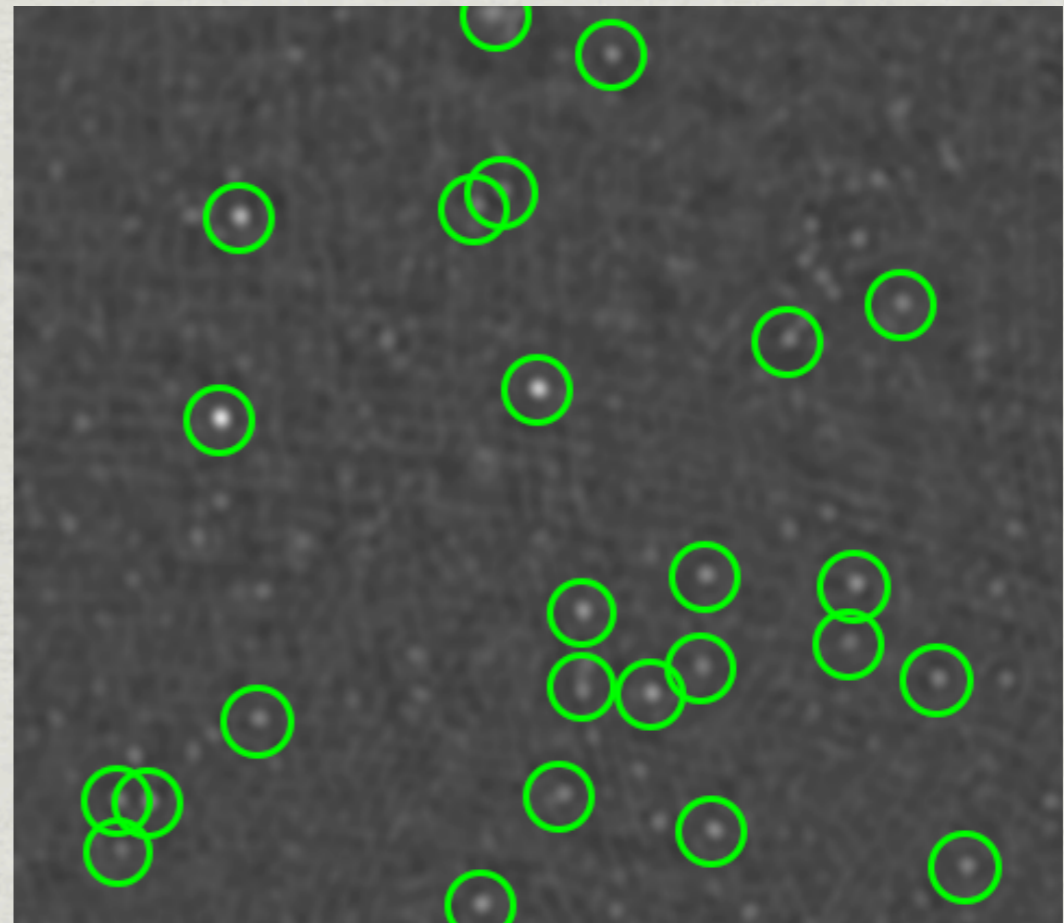
Source extraction

- * Simple Python call; example in a moment
- * Returns source list as Python object for further processing
- * Locate objects by threshold above RMS noise or using FDR algorithm
- * **Code available**

SE example

```
>>> import tkp_lib.dataset as ds
>>> image = ds.ImageData(ds.FitsFile('L3464_cal_ch_cl3k.fits'))
>>> source_list = image.sextract(det=10)
>>> len(source_list)
31
>>> for src in source_list[:5]:
...     print "RA: %.3f  dec: %.3f  flux: %.3f" % (src.ra, src.dec, src.flux)
RA: 237.544  dec: 62.653  flux: 9.764
RA: 242.429  dec: 65.908  flux: 23.604
RA: 319.505  dec: 60.837  flux: 30.525
RA: 222.317  dec: 63.221  flux: 11.497
RA: 227.453  dec: 70.754  flux: 12.767
```

Run time (this laptop): 3.6s



Databases

- * MySQL: 'pipeline'
 - * Snapshot of the current sky; LOFAR sky model
 - * Source association
- * MonetDB: 'catalogue'
 - * Performance & data mining

Events and responses

	Event		Response
Observed Event	> $n\text{-}\sigma$ increase new source	\sim day rise, flat spec, unpol'd < 10s, low freq, high DM single det'n, dispersed	Reobs. for position (within a day), Email SWG (XRB), Send VOEvent Reobs. for time resolution, Email SWG (GRB), Send VOEvent Save TBB, Reobs. for time resolution, Reobs. dispersed event, Email SWG (Pulsar, GRB)
		circ. pol. on \sim mins	Reobs. for position, Obs. with LT, Email SWG (Flare Star), Send VOEvent
	new brightest source all-time high/low > $n\text{-}\sigma$ drop repeating source MAGIC trigger XRB trigger GRB trigger	\sim day rise, flat spec, unpol'd	Reobs. for position (within a day), Email SWG (XRB), Send VOEvent Recalibrate data
		single int, dispersed	Flag event Flag event
			Reobs. for position, Email SWG (Pulsar), Send VOEvent
			Reobs. for position, Reobs. for dispersed event
			Reobs. for position (within a day)
			Reobs. for position, Reobs. for dispersed event, Reobs. for time resolution,
			Reobs. for position (after one day)
			Reobs. for DM Reobs. for spectrum
Known Source	added point to lightcurve	Email SWG (for special sources?)	
	XRB	Email SWG (XRB), Send VOEvent	
	GRB	Reobs. for position (low priority), Send Email, Send VOEvent	
	AGN	Flag event	
	Flare star	Reobs. for spectrum, Save TBB, Observe with LT, Email SWG (Flare Star), Send VOEvent	
	Pulsar/RRAT	Email SWG (Pulsar), Send VOEvent	
	Magnetar	Reobs. for time resolution, Reobs. for dispersed event, Email SWG (Pulsar), Send VOEvent, Send SMS	
	Other galaxy	Reobs. for time resolution, Reobs for position, Email SWG (GRB), Send VOEvent	
	Solar System planet Jupiter	cont. flare ell. pol, quasi-cont. flare	Save TBB, Reobs. for time resolution, Email SWG (Planets), Reobs. Tied-Array Save TBB, Reobs. for time resolution, Reobs. for spectrum (< 40 MHz), Email SWG (Planets), Reobs. Tied-Array
	Exoplanet/near star	ell. pol. quasi-cont flare	Reobs. for dispersed event, Reobs. for time resolution, Reobs. Tied-Array, Email SWG (Planets), Send SMS
SETI planet	spectral line flare	Save TBB, Reobs. for position, Recalibrate data (different flagging), Reimage data (high spec. resolution), Email SWG (SETI)	

Event/response system

- * Designed to be modular
- * Each lightcurve passed to stack of Responder objects
- * Responders are Twisted Python plugins
 - * Quick & easy to add more (example coming up)
- * Responders build 'work package' & submit to queue

Identifying events

- * Lightcurve analysis; from the simple to the detailed
- * Classifier: machine learning
 - * Decision trees, random forests
 - * Code developed by Thijs Coenen; **available**
- * Talk to external databases; AstroGrid
- * Start simple; build on experience

Example Responder

```
from twisted.plugin import IPlugin
from zope.interface import implements
from tkp_response.responders import IResponder
import tkp_response.work

class NewEvent(object):
    implements(IPlugin, IResponder)

    def run(self, lightcurve):
        packages = []
        if len(lightcurve) == 1:
            task = work.SendVOEvent(lightcurve.ra, lightcurve.dec,
                "New transient detected"
            )
            packages.append(
                work.WorkPackage(lightcurve.srcid, 1, tasks=[task])
            )
        return packages

new_event = NewEvent()
```

Building a pipeline

- * Tying everything together; a unified framework
- * Usable for other LOFAR pipelines
- * Built with Twisted, Foolscape
- * Fabric provides deployment and shutdown over the cluster

Pipeline components

- * Twisted application framework (“twistd”) does the hard work for us:
 - * Logging, networking, remote procedure calls (via Foolscape), daemonizing, ..., all built in
- * Components call on a library of customizable, pre-defined services
 - * e.g. IPython cluster, file patch watching, ...

Conclusions

- * We are developing a high performance, parallel processing pipeline to monitor the sky for transients with LOFAR
- * Python & its associated libraries makes this task tractable
- * We have developed various software components that may be of more widespread interest, and encourage you to talk to us about using them

MonetDB

TPCH scale factor 5, timing in seconds

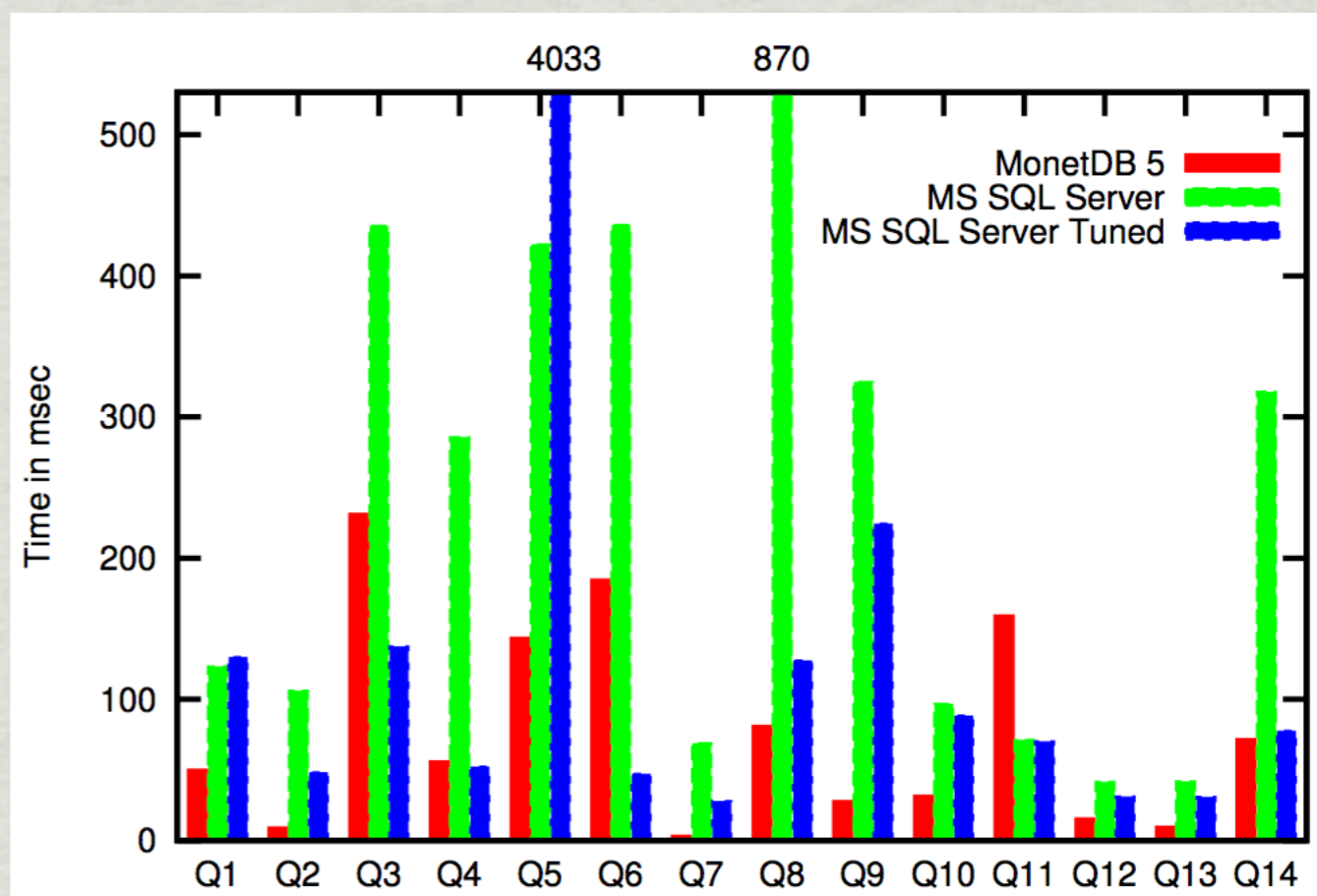
	MonetDB/SQL	PostgreSQL	MySQL
Q1	8.3 [9.6]	192	102
Q2	0.2 [2.4]	23	629
Q3	4.3 [11]	157	156
Q4	2.3 [6.8]	16.6	13
Q5	2.1 [5.0]	2.9	73
Q6	0.6 [0.6]	62	18
Q7	3.2 [3.2]	125	21
Q8	1.1 [3.8]	93	4.2
Q9	3.0 [6.8]	765	29
Q10	1.9 [15]	3.3	135
Q11	0.2 [0.3]	11.6	1.3
Q12	1.5 [3.8]	78	17
Q13	17 [18]	43	25
Q14	0.6 [0.6]	73	93
Q15	0.4 [1.9]	29	34
Q16	1.4 [1.7]	56	28
Q17	1.5 [2.7]		3
Q18	2.5 [3.3]	351	
Q19	3.9 [6.7]	45	0.8
Q20	1.3 [1.4]		1.4
Q21	5.6 [5.8]	166	18
Q22	0.9 [1.1]		1.3
load	6m50	25m14	8m42

MonetDB is somewhat slower

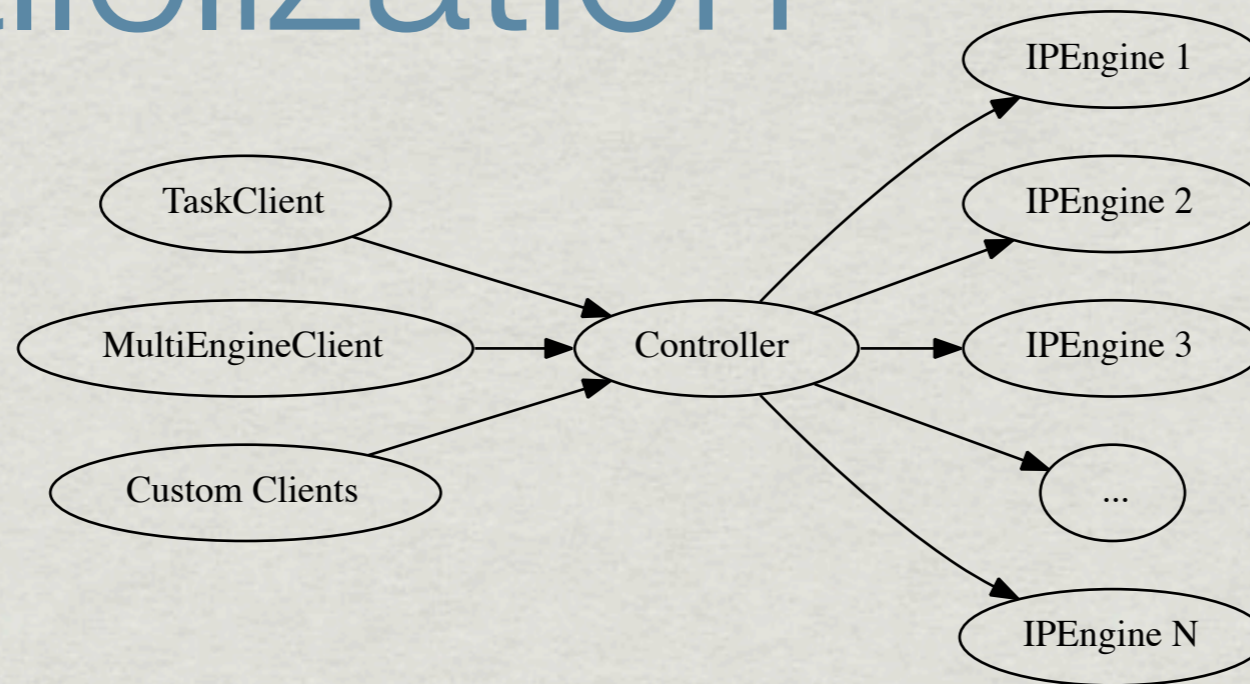
MonetDB is >10x faster

Takes >1hr to run

Error, empty result



Parallelization



```
def run_sextract(filename):  
    image = ds.ImageData(ds.FitsFile(filename))  
    results = image.sextract()  
    with closing(db.connection()) as con:  
        results.savetoDB(con)  
  
task_ids = []  
for filename in file_list:  
    task = tc.StringTask("run_sextract(filename)",  
        push=dict(run_sextract=run_sextract, filename=filename)  
    )  
    task_ids.append(tc.run(task))  
tc.barrier(task_ids)
```