Transients Key Project: Pipeline Overview and Status

John Swinbank
University of Amsterdam

DCLA Project Meeting
26/27 June 2007

Overview

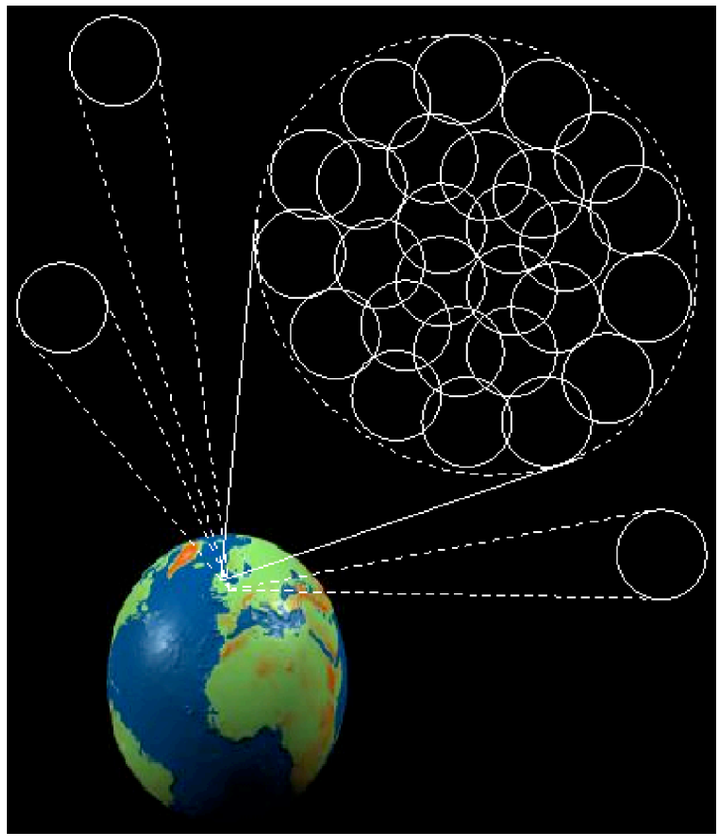
- Pipeline structure
- Data handling
- Identifying transients
- Finding & measuring sources
- Databases
- Classification
- Triggering
- Current status & demo

Pipeline Group Members:

Thijs Coenen,
Casey Law,
Joseph Masters,
James Miller-Jones,
Bart Scheers,
Hanno Spreeuw,
John Swinbank
Ben Stappers,
Ralph Wijers,
Michael Wise

What are we aiming for?

- The Radio Sky Monitor concept
  - Monitoring the whole sky continuously for radio transients
  - More practically: 25% of the sky in 24 hours, focusing on the zenith and galactic plane

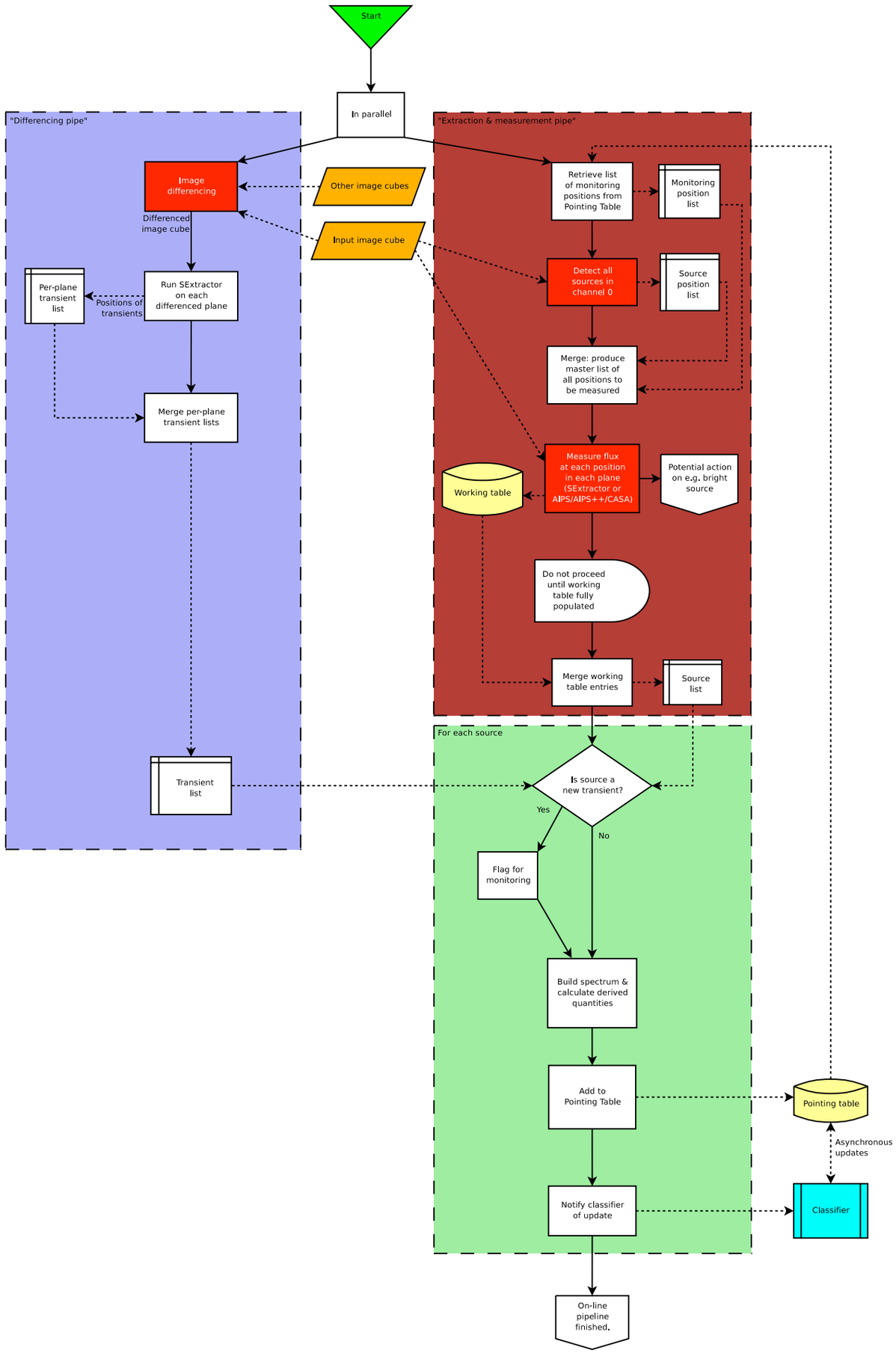


- Logarithmic series of timescales: 1 to 10^5 seconds
  - Complete processing of two datasets every second: performance critical.

- Collaboration
  - Maintain a publicly accessible lightcurve archive
  - Generate and respond to external triggers

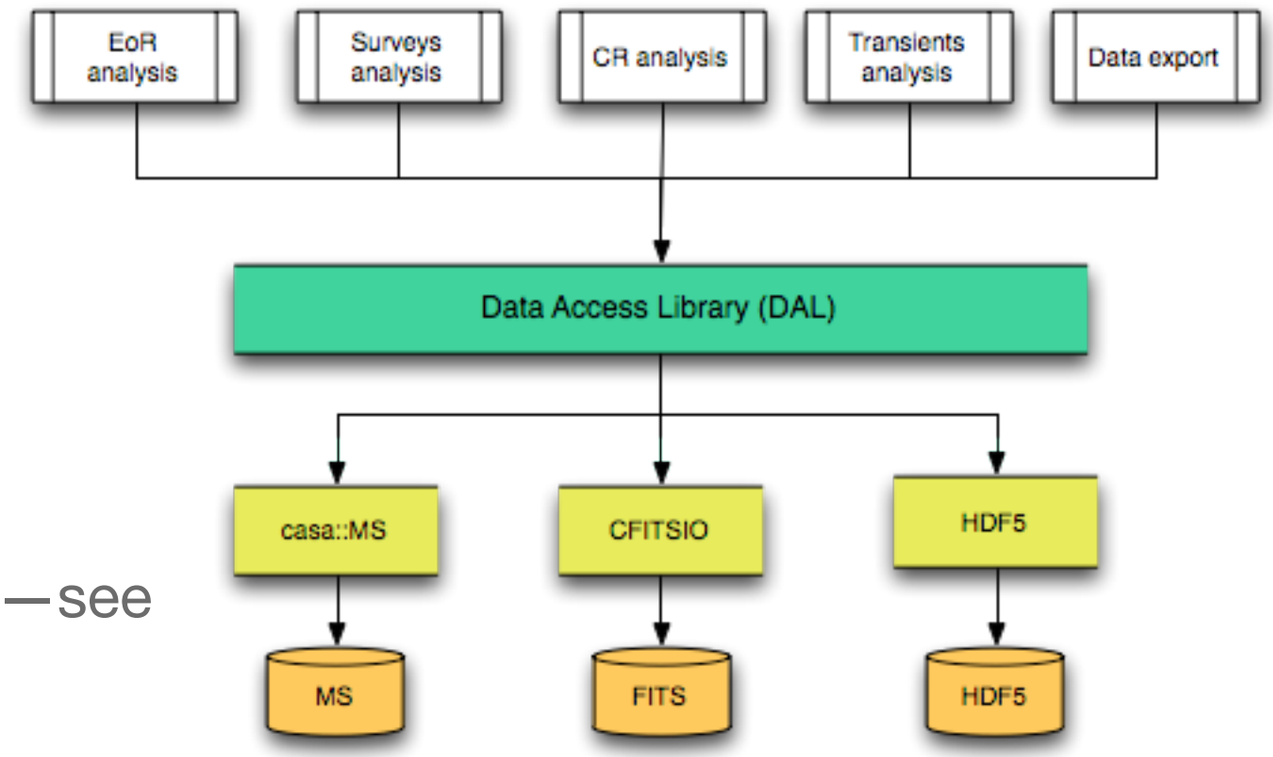
Pipeline Overview

- Input data already processed by main central LOFAR pipeline
- Three major stages to consider:
  - Detecting new transients: differencing
    - Monitoring known transients
    - Archiving all measurements
- Two (three) database areas
- Prototyping and development using *Python*; potential to implement performance critical components in *C++* if required.



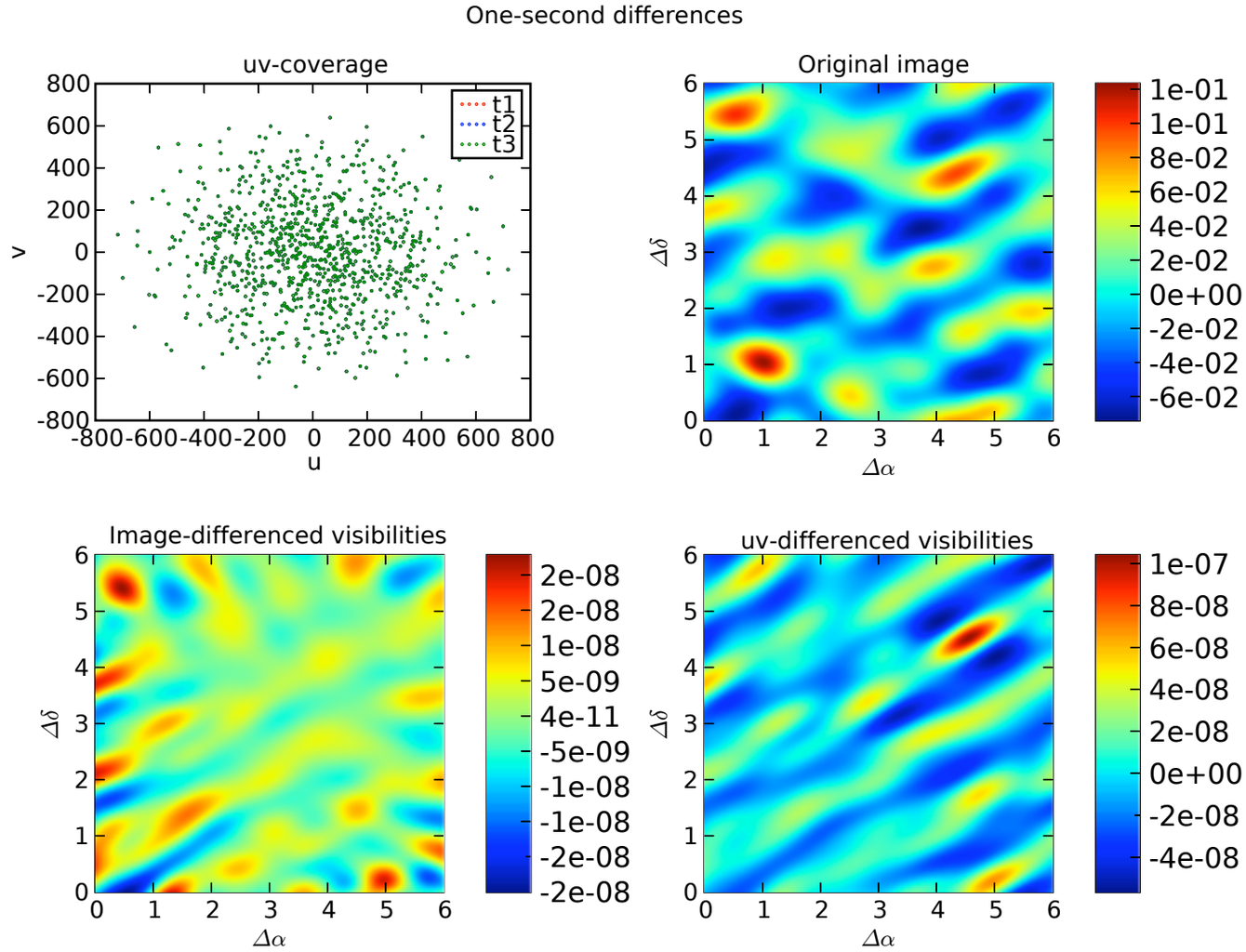
Data Handling

- We are flexible in our data input source:
  - MeasurementSets
  - FITS files
  - Later HDF5
- Current data input through a hacked up mixture of AIPS++, PyCASA & PyFITS
- Ready to switch to Data Access Library—see Joe Masters' talk next—as soon as it's mature.
- Internally, an 'imagedata' class stores data as a NumPy array of pixel values + associated metadata.
  - High performance & flexible
  - Easy to integrate with other tools (later)



Identifying Transients

- Differencing
- Simulations
  - *uv* versus image plane
  - Two or three point
- Best signal to noise achieved by three-point image differencing, i.e. $I_t - 0.5 \times [I_{t-\delta t} + I_{t+\delta t}]$



Source Detection and Measurement

Hanno Spreeuw

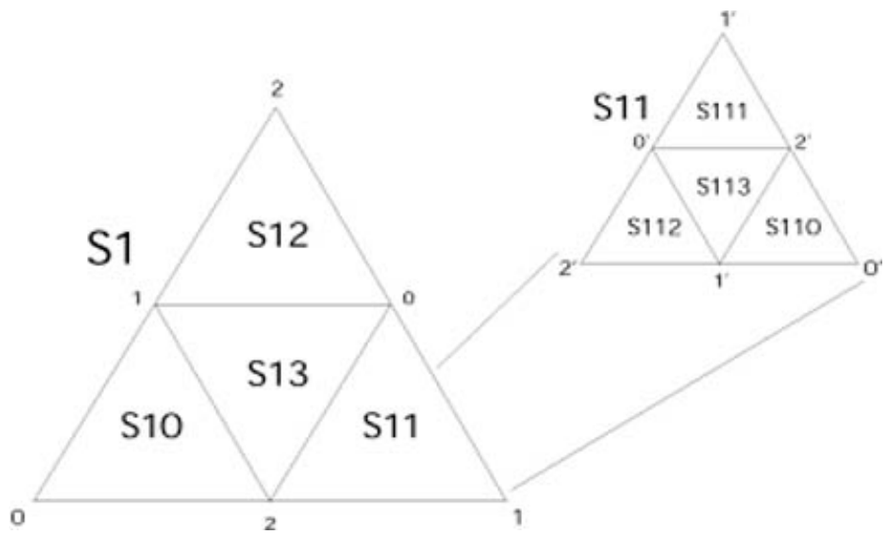
- Need for speed as well as accuracy
- Comparison of different source extraction packages
  - *SExtractor*
    - Fast!
    - Limited fitting options; poor results at high signal to noise
  - *STScI Python*
    - Flexible: *SExtractor* in a few lines of code, easily extended
    - Integration with *NumPy* infrastructure
    - Acceptable performance
    - Measure flux at arbitrary image points
- Pluggable architecture: easy to switch method
  - Possible future synergy with Surveys KP/*BDSM*

Building Lightcurves

- Fluxes measured in each image by flux extraction software; immediate save to database (‘Working Table’).

- Multiple frequency planes of image cube at each timestep → associate sources by position to build spectrum.
  - Once we have a spectrum, calculate derived quantities such as spectral index.

- A spectrum for each timestep is saved in the DB: associating them to produce lightcurves is done in a DB query.
  - Could search simply by position, or by other source properties.
  - HTMIDs make complex positional searches ~~easy~~ tractable.



Databases and Archiving

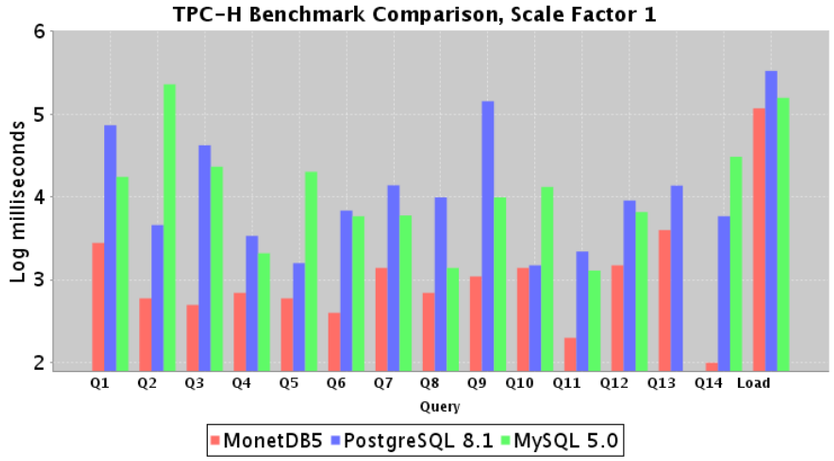
Bart Scheers

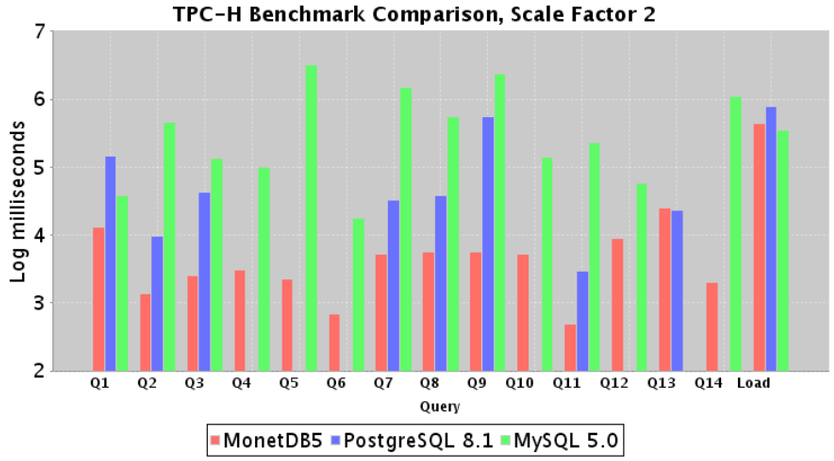
- Working Table
  - Temporary holding pen for raw measurements from images
- Pointing Table
  - Snapshot of the archive for the current pointing on the sky: contains positions for monitoring, measured spectra etc
- Archive
  - Long term storage for all measurements
  - Extreme performance demands: petabyte size, data-mining

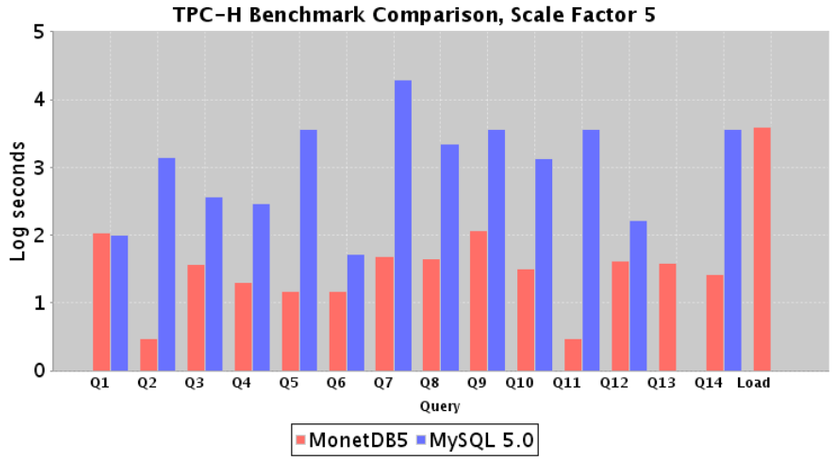
High Performance Databases: *MonetDB*

- *MySQL* is powerful and flexible; ideal for working area. However, it struggle under the archive load.

- *MonetDB* is a high-performance database under development at CWI in Amsterdam
  - Optimised for data mining applications
  - Column based layout (vs. rows in traditional RDBMS): queries are cheap
  - Self tuning, optimised for modern CPUs
  - 10x better performance than traditional systems under some workloads
  - TKP team (Scheers) liaising with CWI on database development







Source Classification

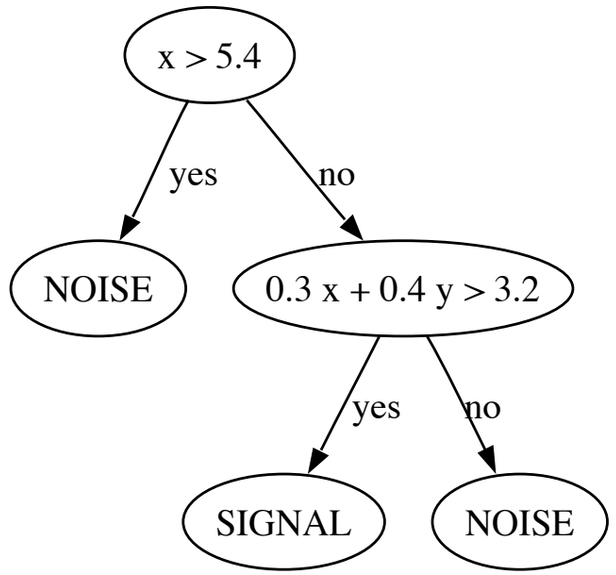
Thijs Coenen

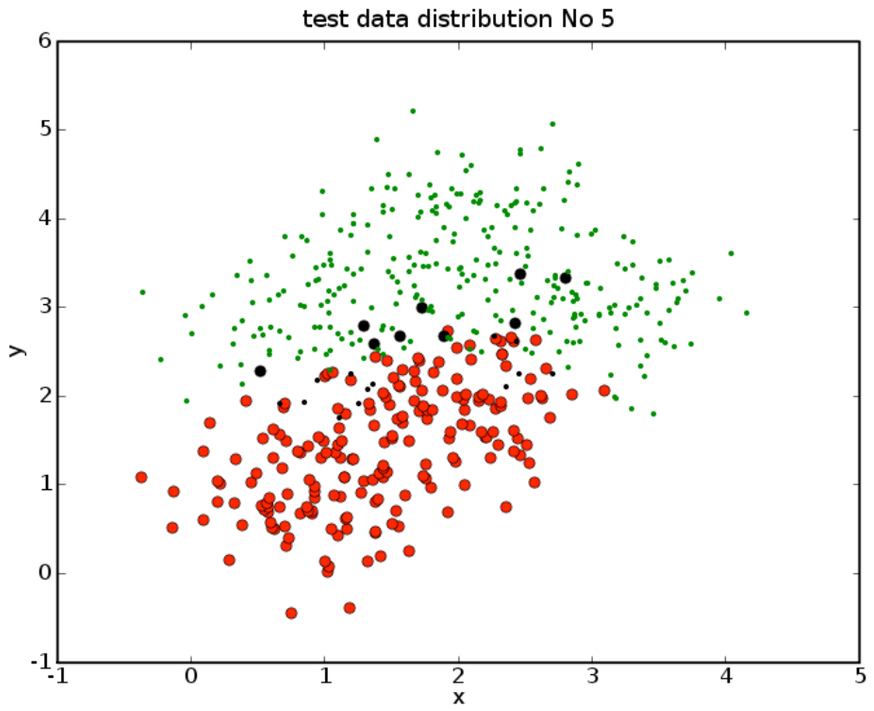
- Classification is needed both for effective data mining and to inform our response to new objects.

- Classifier runs asynchronously from the rest of the pipeline: receives notice of database updates, and processes them in its own time.

- Thus, can make decisions based on time series information without delaying the pipeline.

- Thijs Coenen investigated a variety of different classification algorithms and methods. Best results from a decision tree system.





Triggers and Alerts

- Intelligent response to data received.
- ‘Snap response’ trigger: we detected something noteworthy, now reconfigure the array, freeze the TBBs, etc.
- More considered alert: generated after classifier identifies a new source of a particular type that satisfies quality control constraints.
  - Foster direct collaboration with other instruments for responding to transients.
  - Also more generic alert systems like *VOEventNet.*
- Both types of alert easily generated by adding hooks to Python code.
- Also ability to respond to alerts: both our own and from other instruments (this feature not yet implemented).

Current Status & Demo

- Pipeline currently in prototyping and development stage.
- Most core functionality exists (see talk by Casey Law); database integration still being worked on.
- Classifier yet to be integrated.
- Next the three Ps: profiling, performance and parallelization.
- Full end-to-end prototype complete by the end of this summer.
- To finish: a demo of work in progress.