H2020-Astronomy ESFRI and Research Infrastructure Cluster
(Grant Agreement number: 653477).

UNIVERSITY OF CAMBRIDGE

Task-based distributed processing for radio-interferometric imaging with CASA

BOJAN NIKOLIC

2nd ASTERICS-OBELICS Workshop

16-19 October 2017, Barcelona, Spain.

UNIVERSITY OF CAMBRIDGE

Motivation

The “embarrassing” parallelism in Radio Astronomy

UNIVERSITY OF CAMBRIDGE

Typically each observation can be processed completely independently of all others

Each observation is typically an hour of time

8700 hours in a year => almost 10000 way parallelism if we are patient

That is enough to scale the SKA data reduction to a single node!

But:

- Sometimes can’t wait! Calibration, fast transients
- Very large storage requirements – about 1MW of spinning drives

UNIVERSITY OF CAMBRIDGE

Traditional approaches

Use a single, high specification, computer and be patient
Ideally with an automated pipeline script,

But sometimes :

The human pipeline: distributed data reduction for ALMA

Scott L. Schnee[*a], Crystal Brogan[a], Daniel Espada[b,c,d], Elizabeth Humphries[e], Shinya Komugi[c], Dirk Petry[e], Baltasar Vila-Vilaro[b], and Eric Villard[b]

[a]National Radio Astronomy Observatory, 520 Edgemont Rd, Charlottesville, VA, U.S.A.; [b]Joint ALMA Observatory, Alonso de Cordova 3107, Vitacura, 763-0355, Santiago, Chile; [c]National Astronomical Observatory of Japan (NAOJ), 2-21-1 Osawa, Mitaka, 181-8588, Tokyo, Japan; [d]The Graduate University for Advanced Studies (SOKENDAI), 2-21-1

Objectives for task based parallelisation

UNIVERSITY OF CAMBRIDGE

Time-to-solution

- Non observation-parallel use casses
- Free-up expensive fast storage space as quickly as possible
- Feedback on quality of observations quicker
- Scientific results sooner
- Need fewer People

Distributed memory

- Avoid the expense and scaling limitation of large shared memory cache-coherent machine
- Typical HPC nodes have 64/128 GB memory while increasing trend to 1TB+ RAM machines

Efficient use of I/O bandwidth

- Overlap of I/O and processing inside an application difficult to arrange
- Many simultaneous tasks allow a queue of I/O requests to built up

UNIVERSITY OF CAMBRIDGE

Tasks

UNIVERSITY OF CAMBRIDGE

Tasks

1. Take arguments and produce a result
2. Can decompose into further tasks

What is the minimum constraint on ordering on tasks? How do we avoid over-specifying this ?

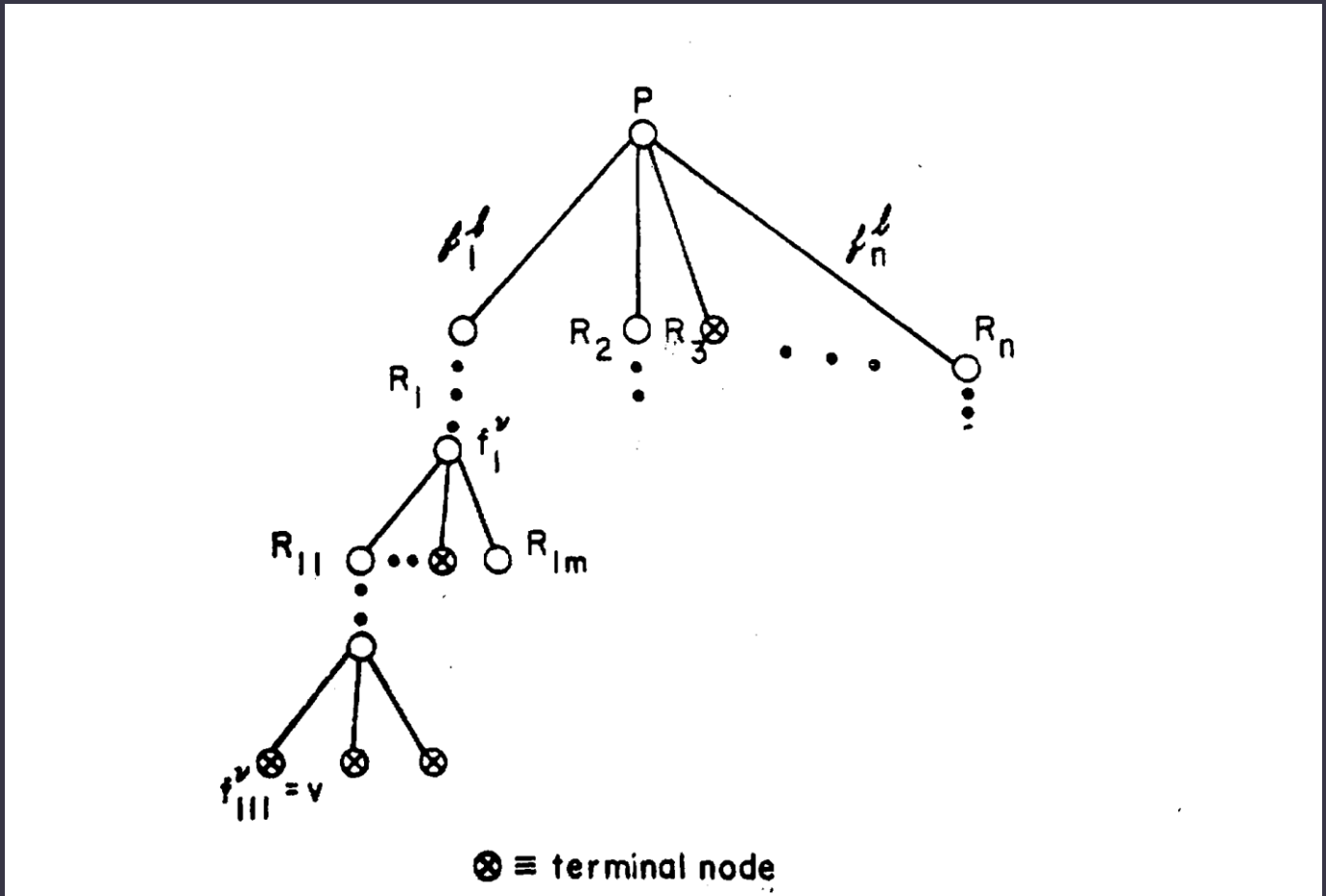


Fig. 1. Service hierarchy.

M. J. Flynn, “Some computer organizations and their effectiveness,” IEEE Trans. Computers, vol. 21, no. 9, pp. 948–960, Sep. 1972.

UNIVERSITY OF CAMBRIDGE

Tasks

1. Take arguments and produce a result
2. Can decompose into further tasks

Does (lexical & run-time) program structure correspond to tasks?

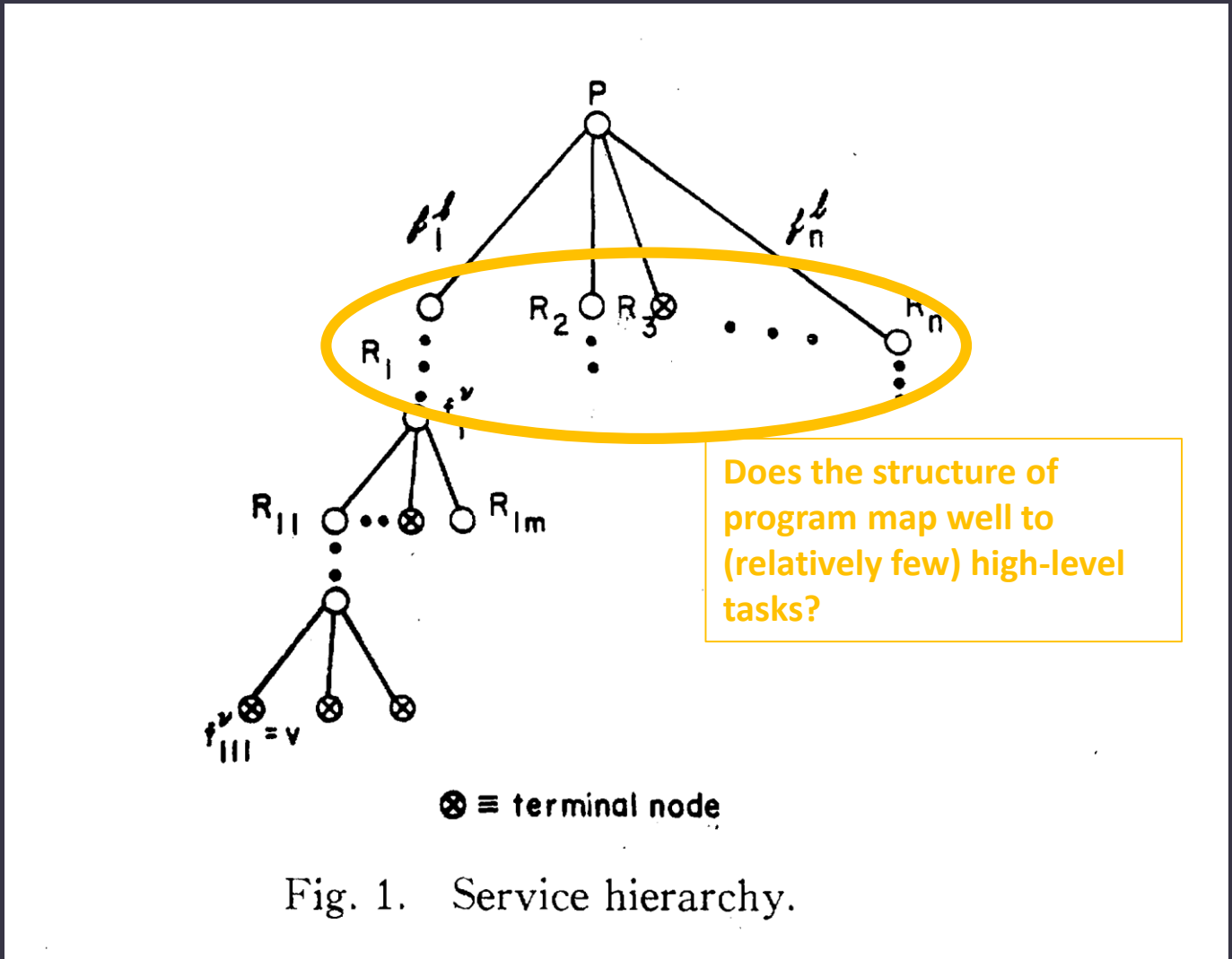


Fig. 1. Service hierarchy.

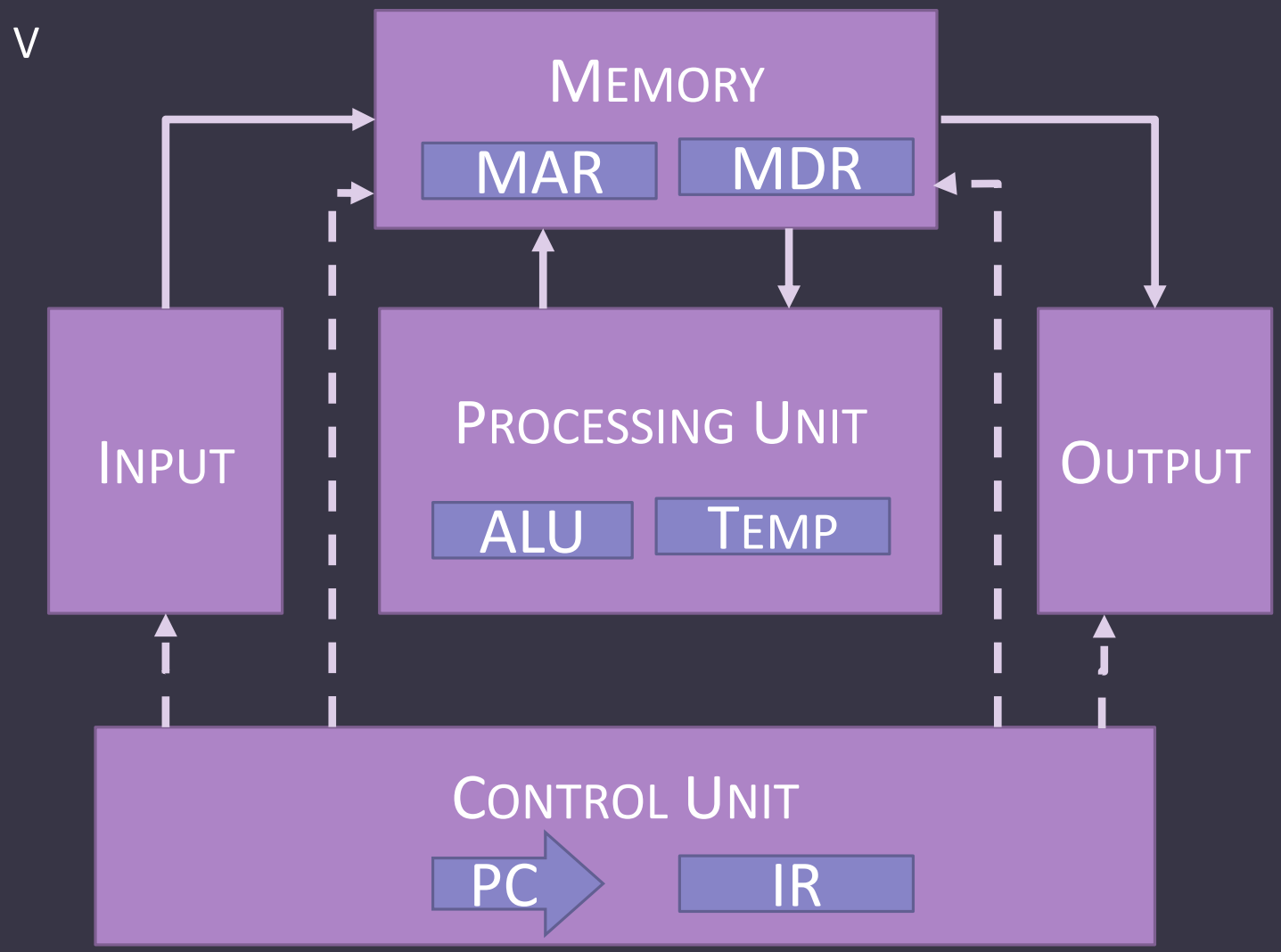
Does the structure of program map well to (relatively few) high-level tasks?

M. J. Flynn, “Some computer organizations and their effectiveness,” IEEE Trans. Computers, vol. 21, no. 9, pp. 948–960, Sep. 1972.

Von Neumann Model

Instruction == Task

Ordering == Program Counter



Tasks

Precedence definition

$$f_i^l(x, \tau) \rightarrow (y, \tau^*)$$

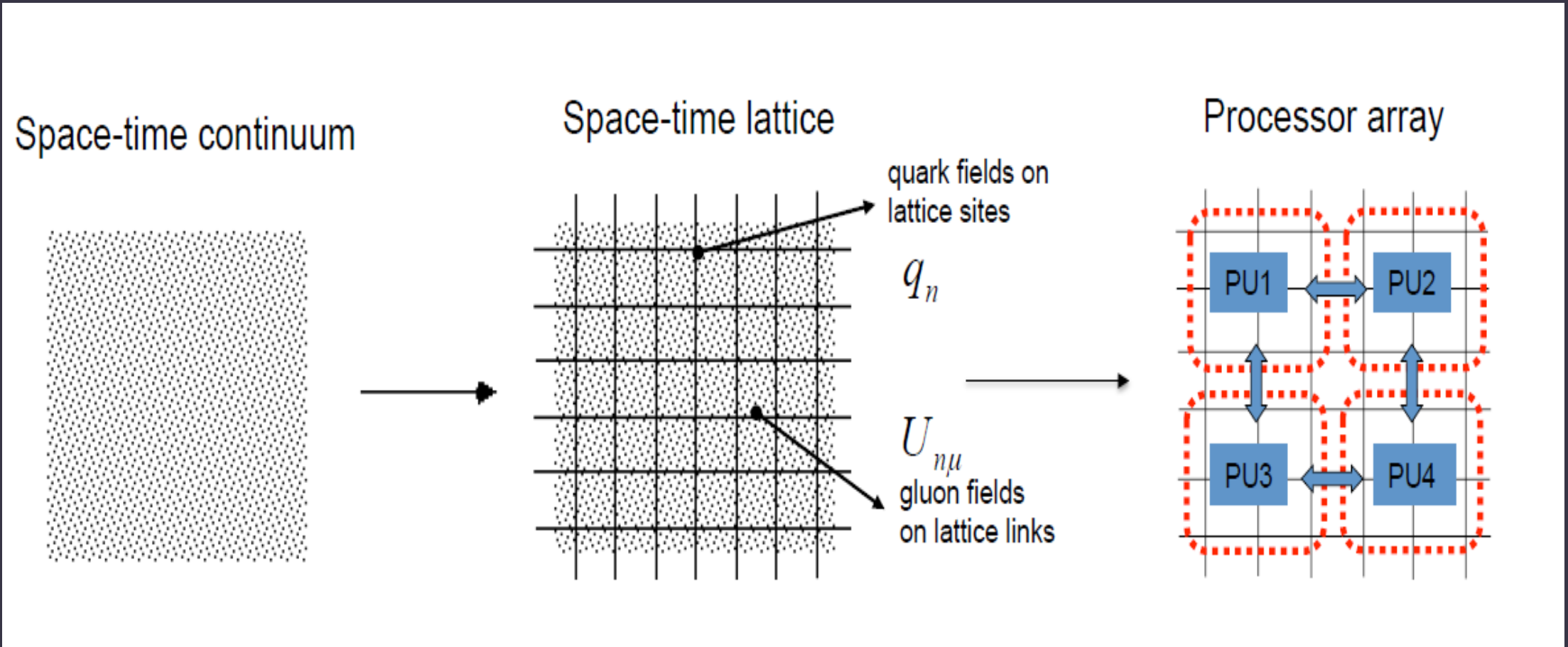
x : Input data

τ : Input control variable, i.e., x valid for use in computation?

y : Task result

τ^* : Output control variable, i.e., is y valid?

QCD: domain decomposition & fine grain global iteration



From Ukawa 2014:
https://indico.cern.ch/event/284433/contributions/1634880/attachments/525606/724848/ukawa.pdf

QCD: domain decomposition & fine grain global iteration

https://www.flickr.com/photos/35734278@N05/3323018571 , BlueGene/P Argonne National Lab

From Ukawa 2014:
https://indico.cern.ch/event/284433/contributions/1634880/attachments/525606/724848/ukawa.pdf

UNIVERSITY OF CAMBRIDGE

AIPS++ vs CASA

"TOOLS": AN OBJECT ORIENTED USER INTERFACE

"TASKS": A TASK-BASED INTERFACE

CASA Script and equivalent task graph

Example: Simple calibration using standard CASA/Python

```
cl.addcomponent(flux=1.0,
                dir="17h45m40s -29d0m28s")
cl.rename("GC.cl")
cl.close()

ft('indata.ms', complist="GC.cl", usescratch=True)

gaincal('indata.ms',
        caltable='K.cal',
        gaintype='K')

gaincal('indata.ms',
        caltable='G.cal',
        gaintable='K.cal',
        gaintype='G',
        calmode='p')

applycal('indata.ms', gaintable=['K.cal', 'G.cal'])

split('indata.ms', 'caleddata.ms')
```

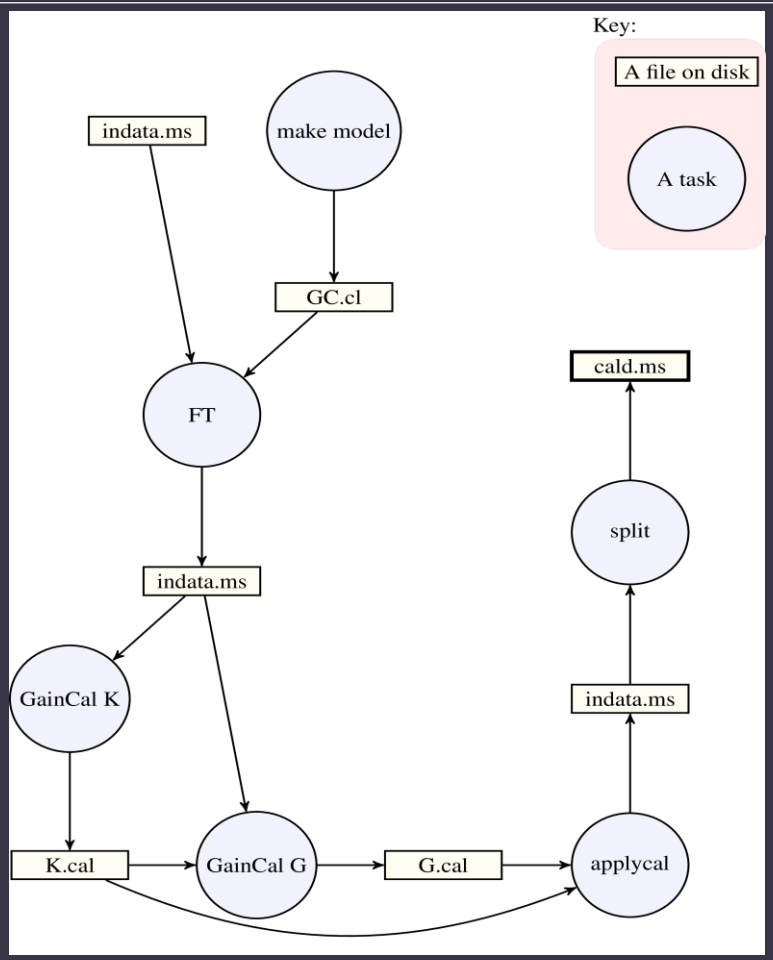
CASA *Tool* invocation

Non-atomic

Dependency on 'GC'

Explicit output name

Dependency on 'K'



UNIVERSITY OF CAMBRIDGE

Architecture

UNIVERSITY OF CAMBRIDGE

Drivers

Modular, readable, source code structure. Small increment from standard CASA usage

Short development time, low maintenance

Easy to use on standard HPC clusters and single nodes

Reasonable performance and scalability

UNIVERSITY OF CAMBRIDGE

SWIFT/T

swift

LANGUAGE

Functional language with only I-Structure variables

- Every (lexical) variable can only be assigned to at most once
- This enables a compile time and run-time exact dataflow inference

First class support for **files** and **arrays**

Simple, familiar looking, syntax (not Haskell!)

Simple but adequate type system

EXECUTION ENGINE (TURBINE)

Designed for HPC clusters with a shared filesystem (Lustre/GPFS) and MPI-3 libraries

High task issue rate

Fully dynamical load-balancing

Capability for in-process tasks (including CASA!)

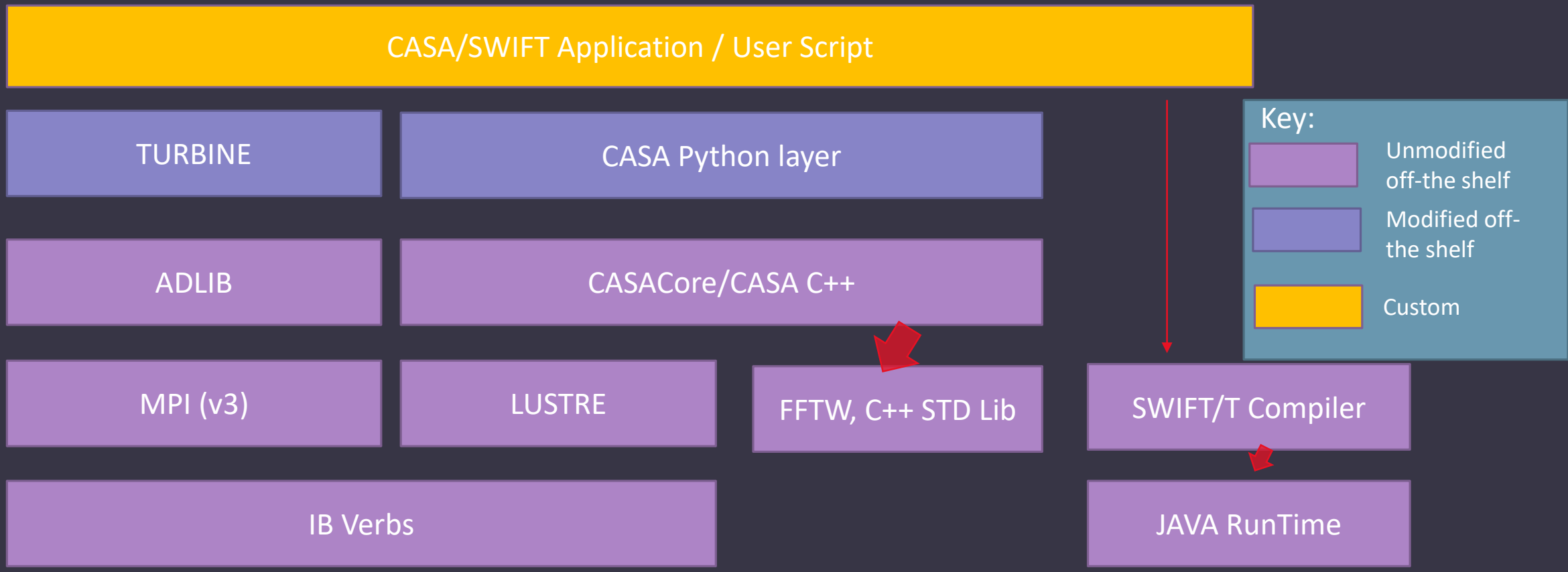
- No overhead of starting a process for each task

Integrates with cluster schedulers (SLURM, etc). Works out of box on typical cluster.

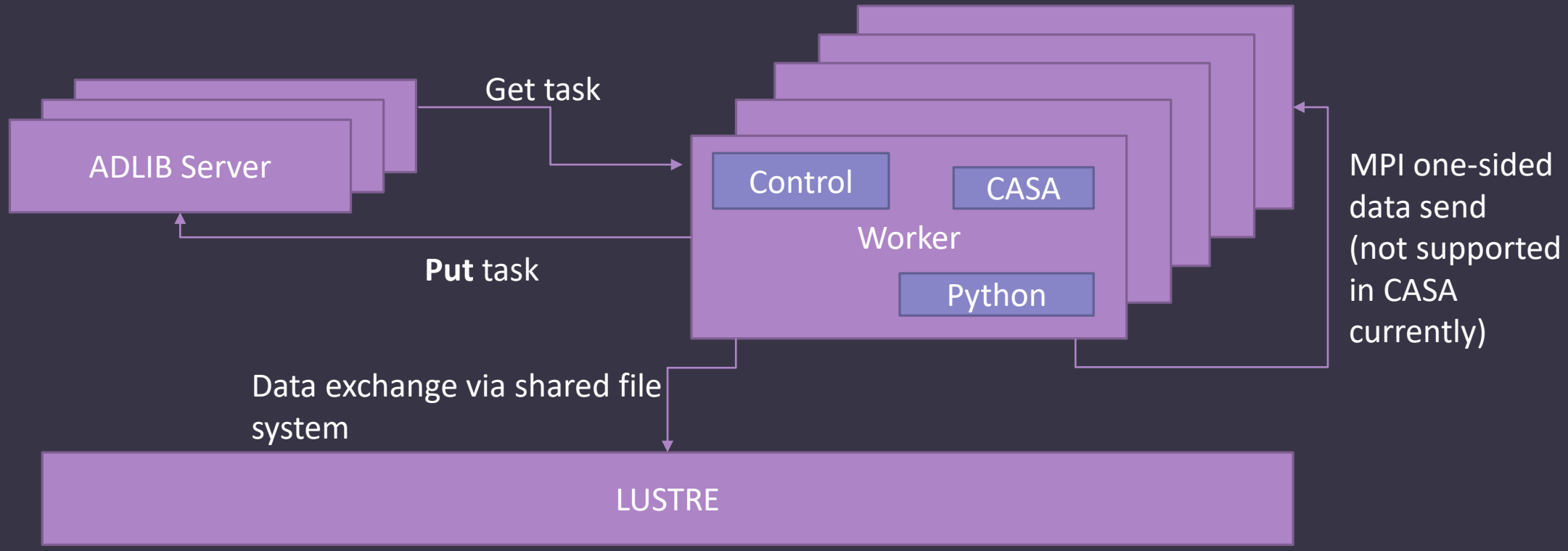
See Wilde (2011) for the SWIFT language; See Wozniak (2013) for description of SWIFT/T

UNIVERSITY OF CAMBRIDGE

Module Architecture (simplified)



RUN TIME ARCHITECTURE (COMPONENTS)



Drivers: Scalability, Load balancing, high task issue rate, low task issue overhead

UNIVERSITY OF CAMBRIDGE

CASA to SWIFT/T

Example: Simple calibration using standard CASA/Python

```
cl.addcomponent(flux=1.0,
                dir="17h45m40s -29d0m28s")
cl.rename("GC.cl")
cl.close()

ft('indata.ms', complist="GC.cl", usescratch=True)

gaincal('indata.ms',
        caltable='K.cal',
        gaintype='K')

gaincal('indata.ms',
        caltable='G.cal',
        gaintable='K.cal',
        gaintype='G',
        calmode='p')

applycal('indata.ms', gaintable=['K.cal', 'G.cal'])

split('indata.ms', 'caleddata.ms')
```

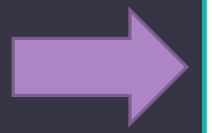
CASA *Tool* invocation

Non-atomic

Dependency on 'GC'

Explicit output name

Dependency on 'K'



Example: Simple calibration using new CASA/SWIFT-T which enforces single assignment (IVars)

```
(file calvis) calinitial(file visin)
{
  file model=mkinitmodel("17h45m40s -29d0m28s",
                         1.0);
  file m3=casa_ft(visin, model,
                  usescratch=true);
  file c1=casa_gaincal(m3, [], gaintype="K");

  file c2=casa_gaincal(m3, [c1], gaintype="G",
                       calmode="p");

  file m4 = casa_applycal(m3, [c1,c2]);

  calvis= casa_split(m4);
}
```

Definition of a re-usable function

Dependency on 'K'-cal explicit

Return by (single) assignment to output IVar

SWIFT/T program and equivalent graph

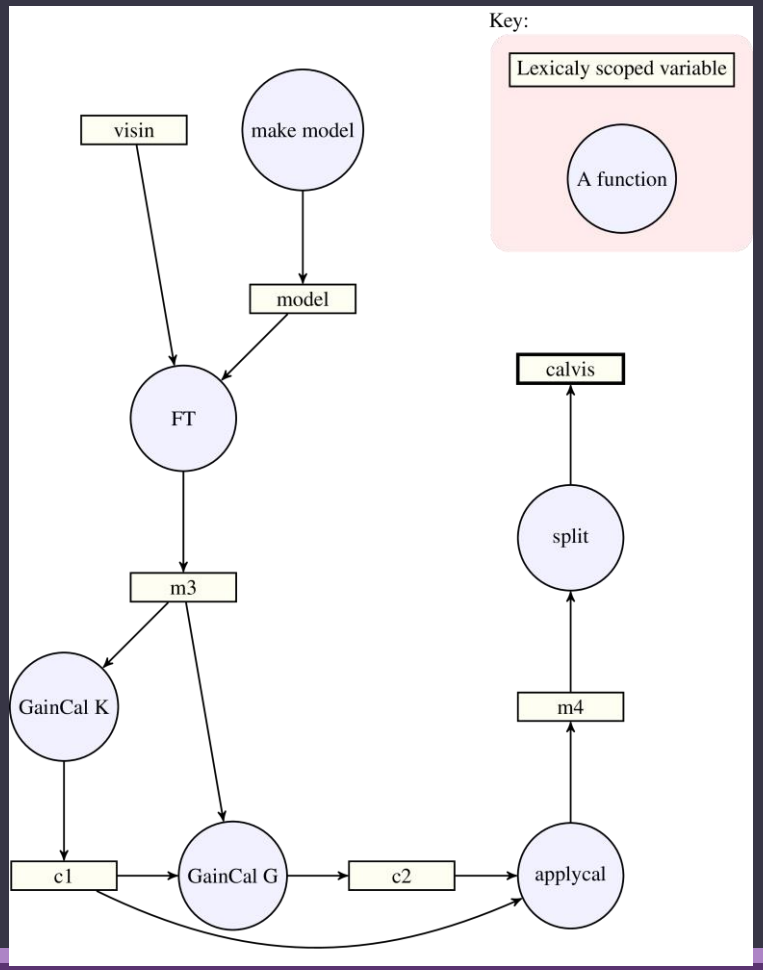
Example: Simple calibration using new CASA/SWIFT-T which enforces single assignment (IVars)

Definition of a re-usable function

```
(file calvis) calinitial(file visin)
{
  file model=mkinitmodel("17h45m40s -29d0m28s",
                         1.0);
  file m3=casa_ft(visin, model,
                  usescratch=true);
  file c1=casa_gaincal(m3, [], gaintype="K");

  file c2=casa_gaincal(m3, [c1], gaintype="G",
                       calmode="p");

  file m4 = casa_applycal(m3, [c1,c2]);

  calvis= casa_split(m4);
}
```

Dependency on 'K'-cal explicit

Return by (single) assignment to output IVar



UNIVERSITY OF CAMBRIDGE

RESULTS

UNIVERSITY OF CAMBRIDGE

MAIN RESULTS

Short development cycle

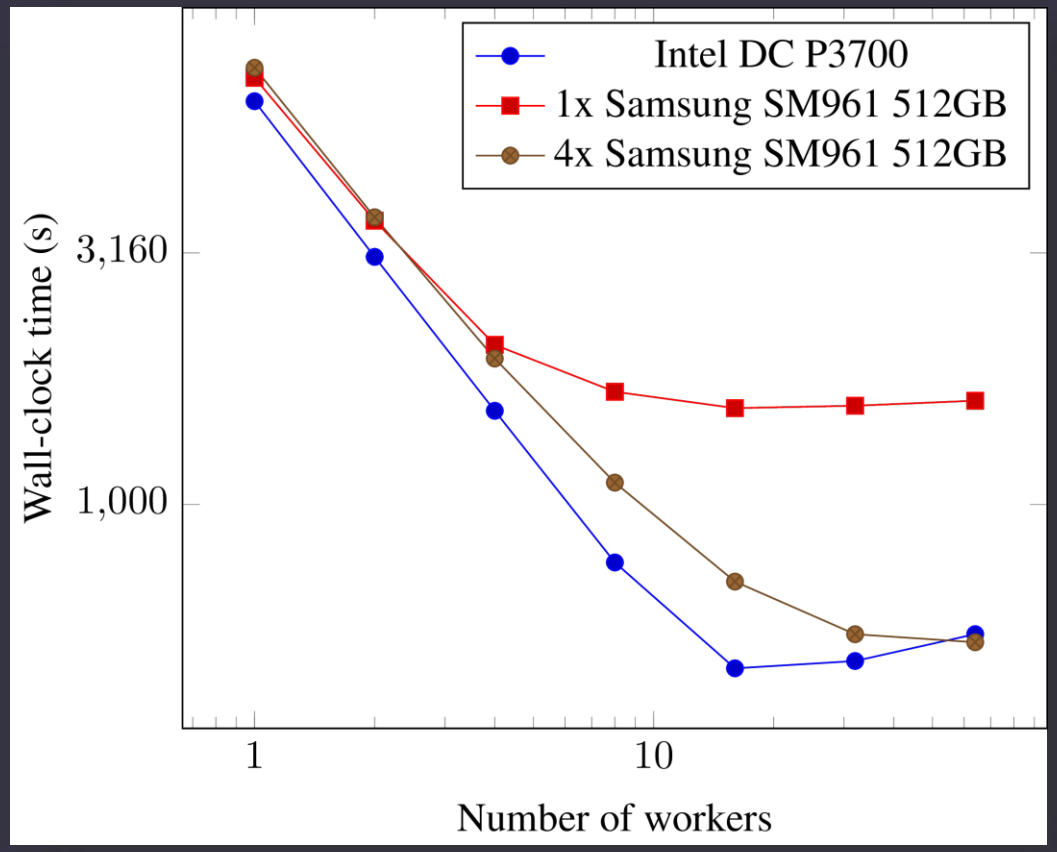
- Two weeks to full implementation (but I was already very familiar with CASA & SWIFT/T)
- Minimal need for interaction with upstream developers

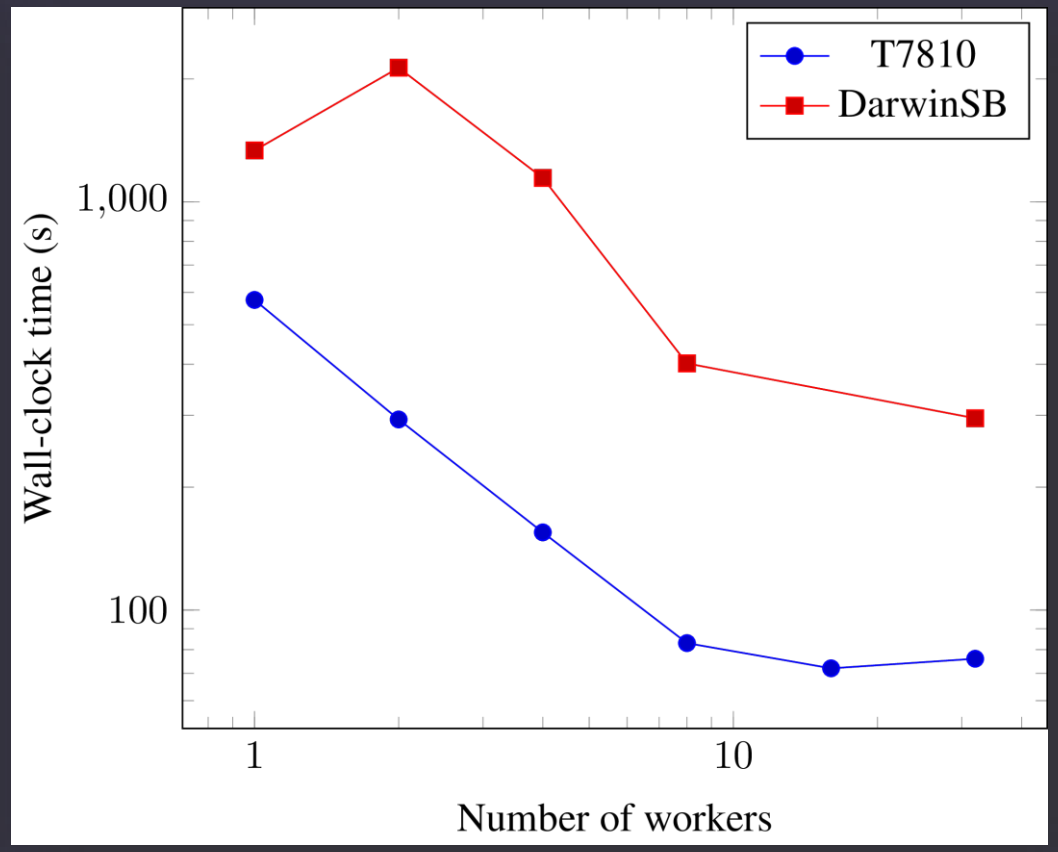
Understandable program structure

- SWIFT if anything more understandable from scientific viewpoint than the underlying Python
- Accurate, reliable, modularisations

UNIVERSITY OF CAMBRIDGE

Scalability





UNIVERSITY OF CAMBRIDGE

More information

Memo with the details full source code

UNIVERSITY OF CAMBRIDGE

Acknowledgement

H2020-Astronomy ESFRI and Research Infrastructure Cluster (Grant Agreement number: 653477).