UNIVERSITY OF CAMBRIDGE

e-MERLIN Software in the SKA Era

Paul Alexander

Aim and approach

- Convergence between SKA and e-MERLIN software environment
  - Possible since SKA software will run in regional centres as well as at the SKA sites
- Leverage UK-SKA investment for e-MERLIN
  - UK is leading the SKA Science Data Processor (SDP)
  - Early access to developing SKA software
- Enable UK community to have early access to the SKA software environment
  - Ability to experiment and develop new algorithms and approaches tested on e-MERLIN used on SKA and vice versa

UNIVERSITY OF CAMBRIDGE

Aim and approach

- Convergence between SKA and e-MERLIN software environment
  - Possible since SKA software will run in regional centres as well as at the SKA sites
- Leverage UK-SKA investment for e-MERLIN
  - UK is leading the SKA Science Data Processor (SDP)
  - Early access to developing SKA software

Key aspect which makes this sensible for e-MERLIN is the co-development between SKAO and NRAO / CASA team of next generation of the measurement set MSV3

SKA will use MSV3 as its data format

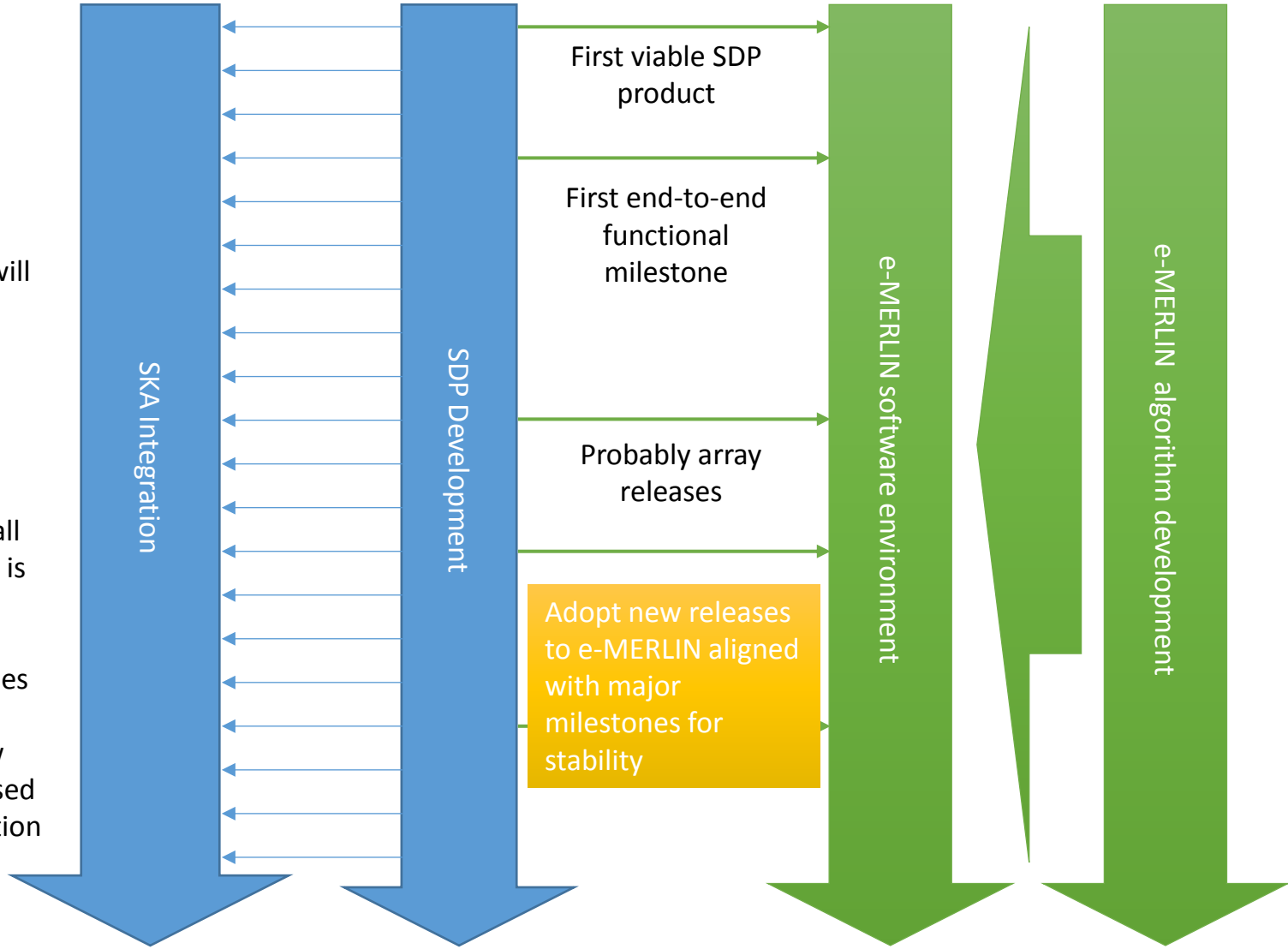
Interface layer will mean MSV2 can be read by MSV3 compatible software and vice-versa

How will this work?

Separate project – no coupling, low risk

2019

- SKA Open Source
- SKA development will use a scaled-AGILE Approach, SAFe
- Maintain a working system throughout
- Integration at overall solution level (SKA) is every 3 months
- Important milestones include the Array Releases when new antennas are released for science verification

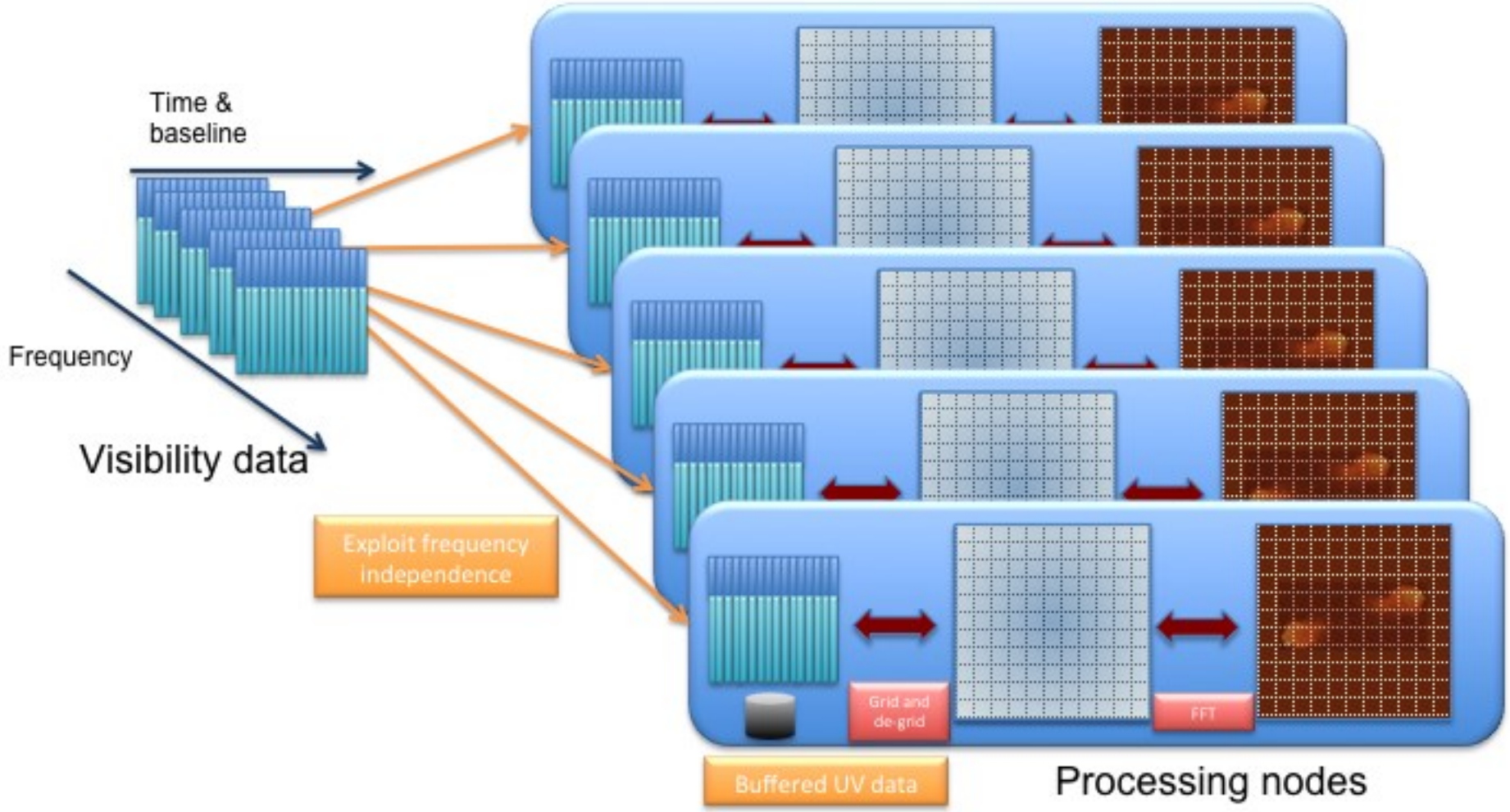


2024

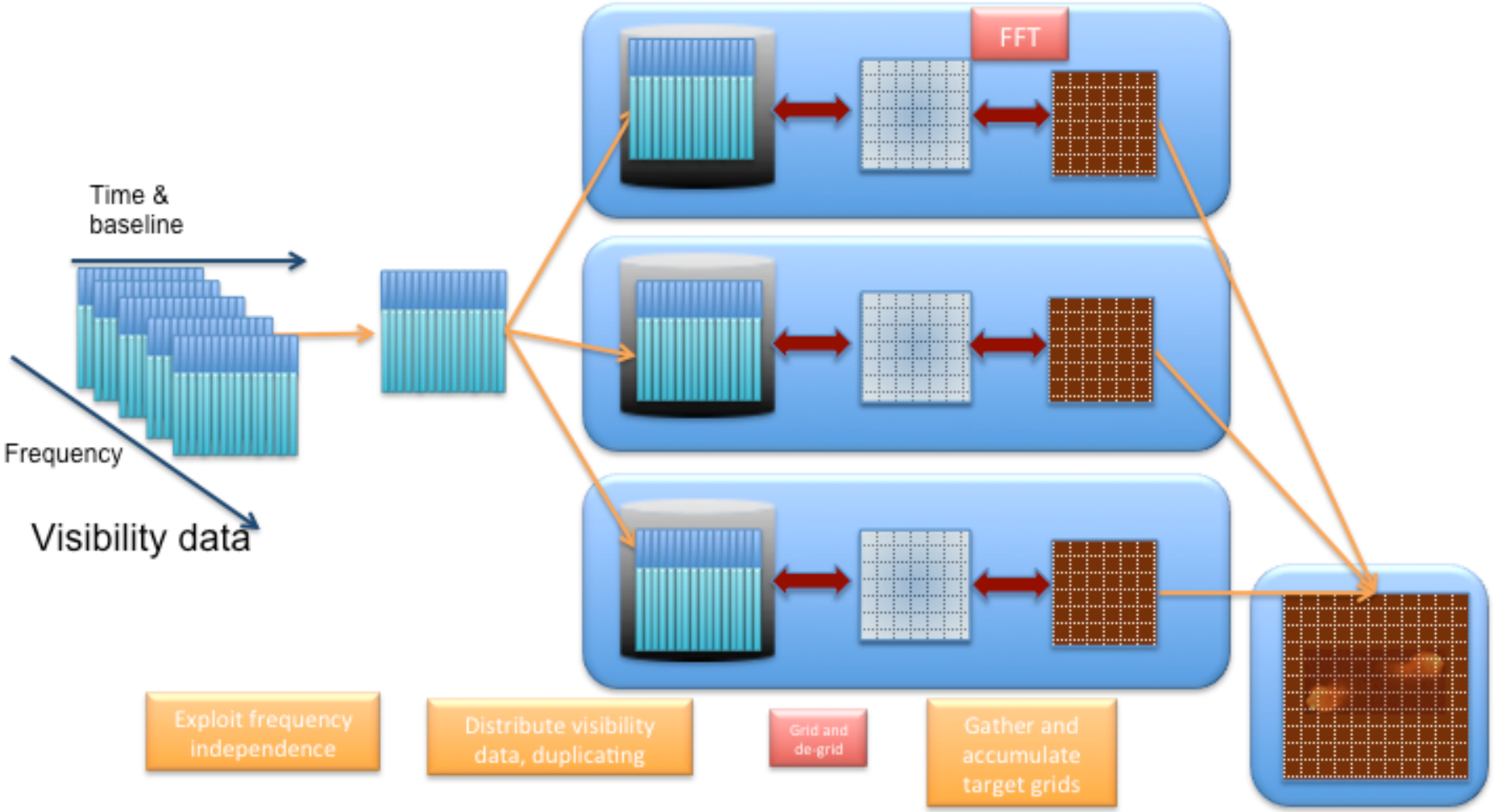
What do we get and what does it look like?

- SKA SDP environment provides:
  - Scalable performance
  - Ease of writing work flows (pipelines) from simple functional components
  - Execution framework is the scripting environment for the work flows which also delivers the scalable performance
  - Functional components are basic functions you want to apply to the data, some high level some lower—level
  - Very likely it will be a Python interface (see prototype later)

Data Driven Architecture

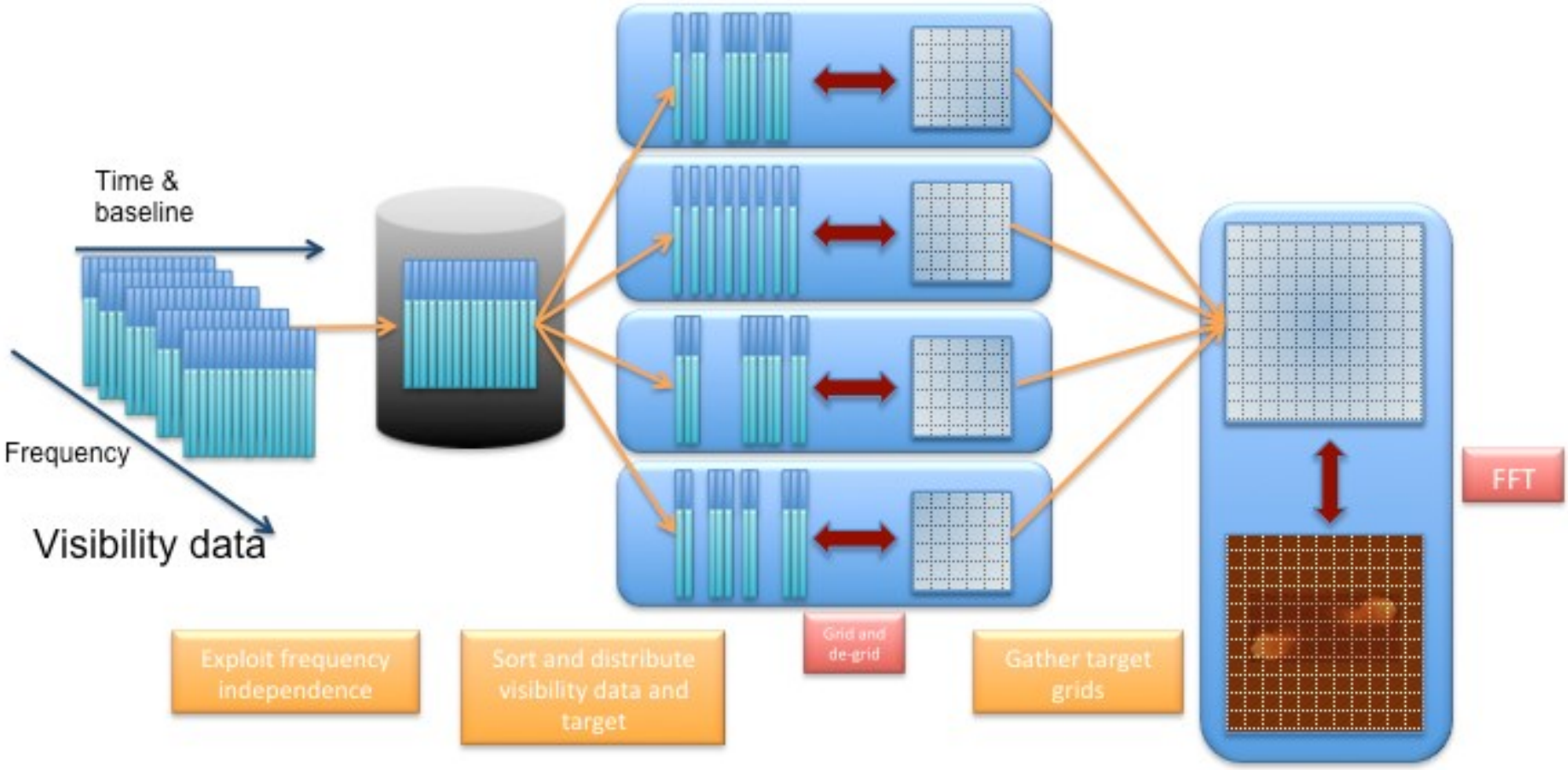


Data Driven Architecture



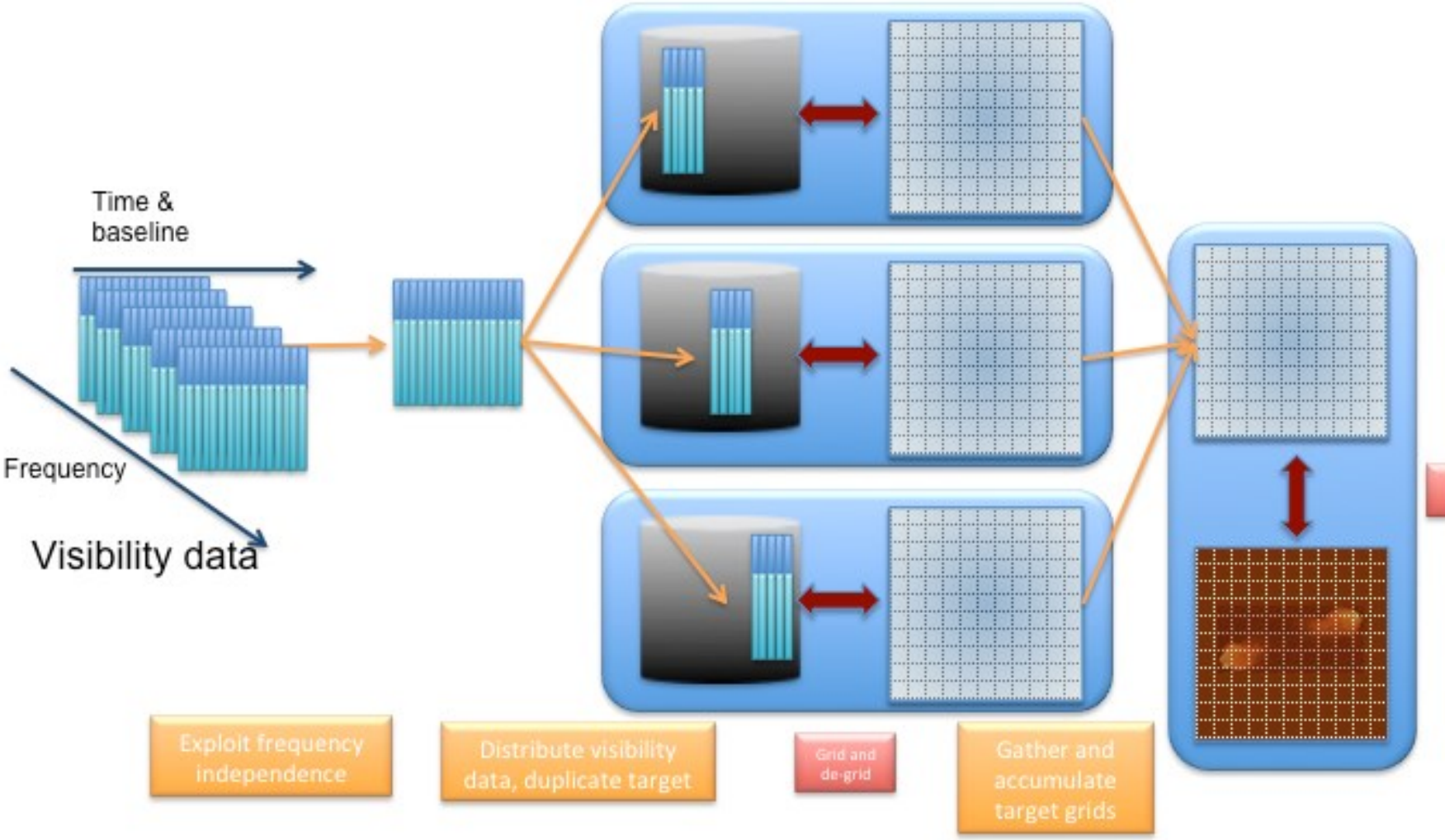
- Smaller FFT size at cost of data duplication

Data Driven Architecture



- Further data parallelism in spatial indexing (UVW-space)
- Use to balance memory bandwidth per node
- Some overlap regions on target grids needed

Data Driven Architecture

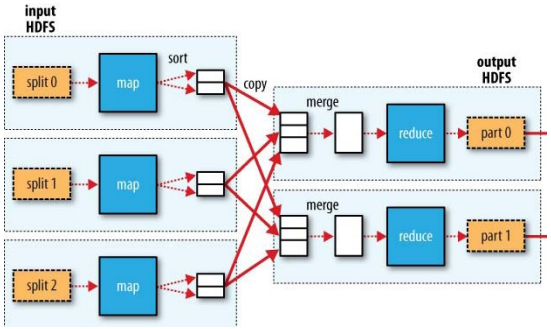


How do we get performance and manage data volume?

Approach: Build on BigData Concepts

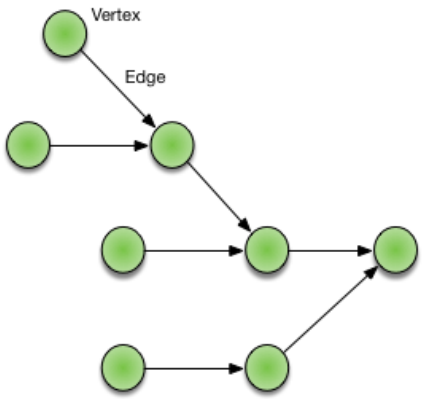
"data driven" → graph-based processing approach receiving a lot of attention

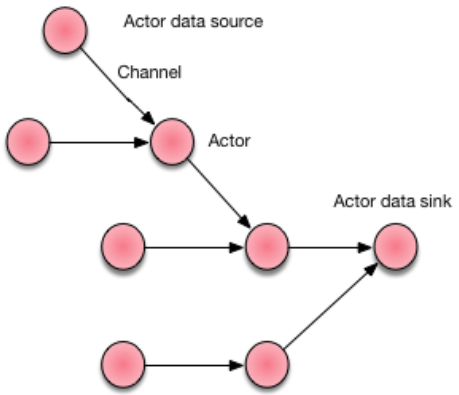
Inspired by Hadoop but for our complex data flow



Hadoop

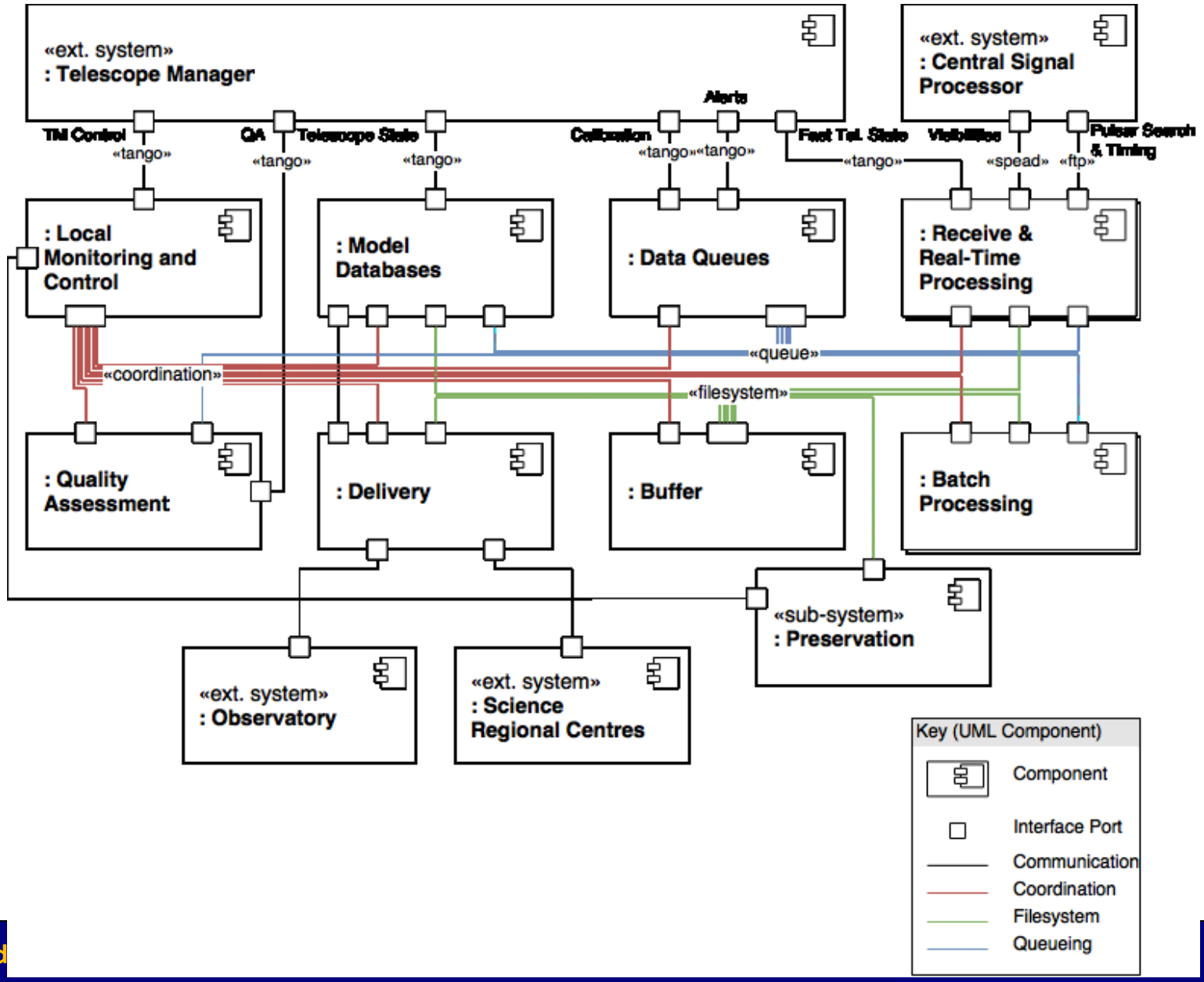
Graph-based approach



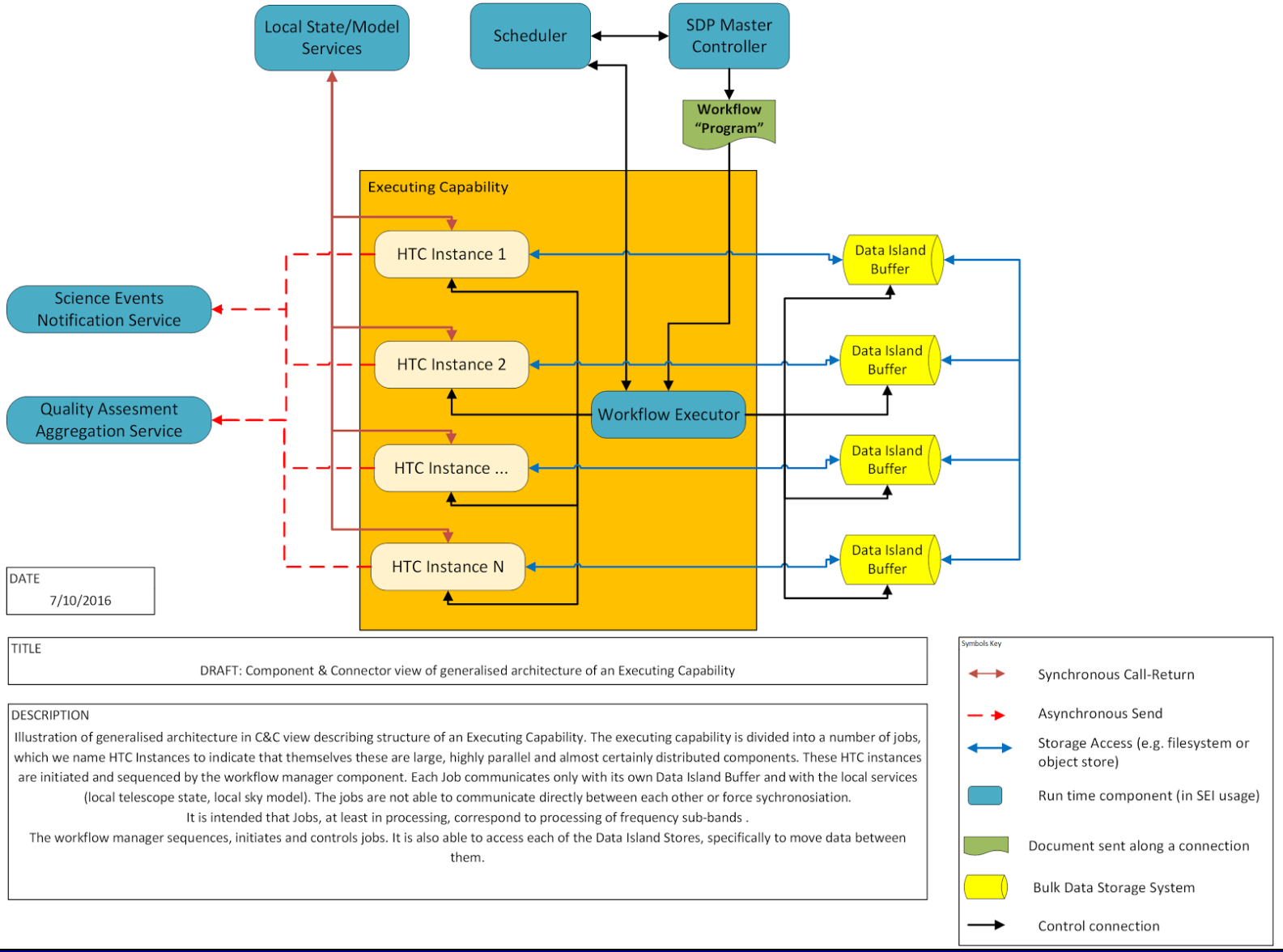


UNIVERSITY OF CAMBRIDGE

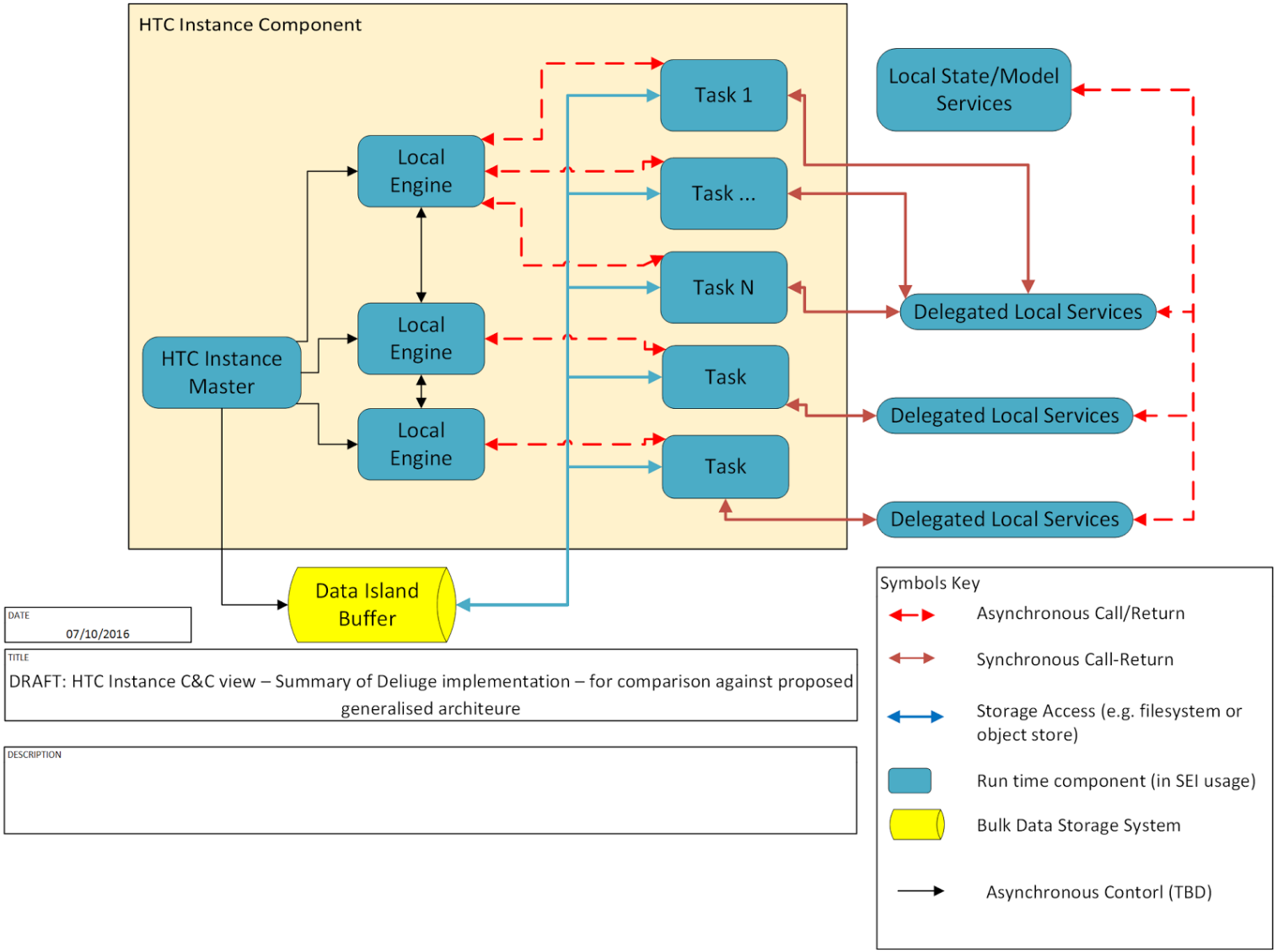
Top-level Component Architecture



Execution Framework



Execution Framework



In addition to performance

- Software to manage and provide complex sky models to the analysis
- Software to manage and provide telescope models
- A data archive
- Ability to run in a standard cloud environment or on an HPC
- Functional components from which all start-of-the-art pipelines can be constructed
  - Easy to experiment with new ideas at scale

What might it look and feel like

- Prototyping using SPARK and DASK execution frameworks
- Toy example using lazy evaluation to build an execution graph wrapping existing functions and evaluate

```
from dask import delayed
L = []
for channel in filenames:
  data loadMSV3(channel)
  L.append(image(data))

result = channelsJoin(L)
```

Serial

```
from dask import delayed
L = []
for channel in filenames:
  data = delayed(loadMSV3)(channel)
  L.append(delayed(image)(data))

result = delayed(channelsJoin)(L)
result.compute()
```

Parallel

And finally a real example – parallel predict

dask predict

```
1  def predict_dask(vt, model, predict_single=predict_2d, iterator=vis_timeslice_iter, **kwargs):
2
3      def accumulate_results(results):
4          i=0
5          for rows in iterator(vt, **kwargs):
6              visslice = create_visibility_from_rows(vt, rows)
7              vt.data['vis'][rows] += results[i].data['vis']
8              i+=1
9
10         return vt
11
12     results = list()
13
14     for rows in iterator(vt, **kwargs):
15         visslice = copy_visibility(create_visibility_from_rows(vt, rows))
16         result = delayed(predict_single, pure=True)(visslice, model, **kwargs)
17         results.append(result)
18
19     return delayed(accumulate_results, pure=True)(results)
```

Courtesy Tim Cornwell

Benefits

- High performance workflows on cloud or HPC systems – increased productivity
- Learn about SKA software environment early – get ready for the SKA era
- Access state of the art algorithms as they become available
- Fully exploit STFC hardware – Dirac or Cloud Infrastructure
- Develop new algorithms
- e-MERLIN ready for the SKA era