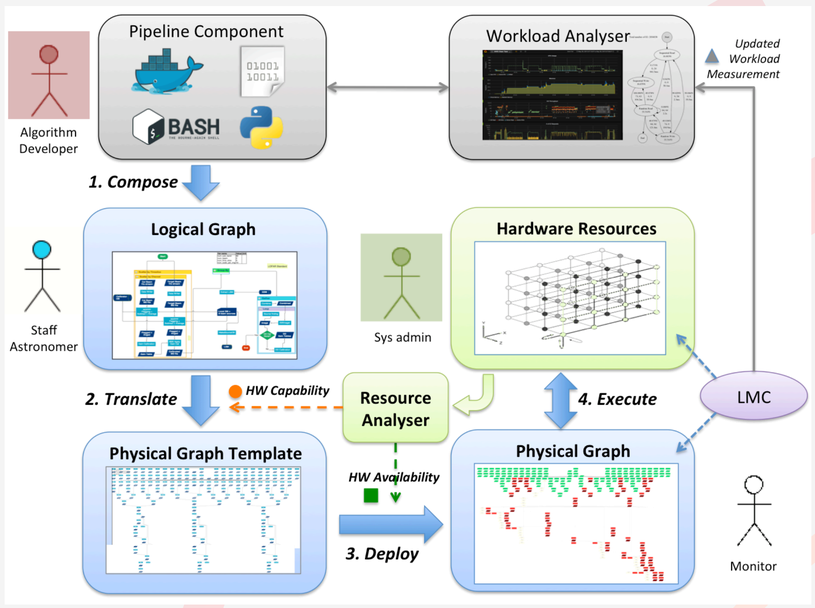
Imaging SKA-Scale Data on Cloud and Supercomputer Infrastructure using Drops and DALiuGE

Mark BOULTON, Ian COOPER, Richard DODSON, Markus DOLENSKY,
Dave PALLOT, Rodrigo TOBAR, **Kevin VINSEN**, Andreas WICENEC, Chen WU
International Centre for Radio Astronomy Research

DALiuGE
Data-Activated 流(Flow) Graph Engine



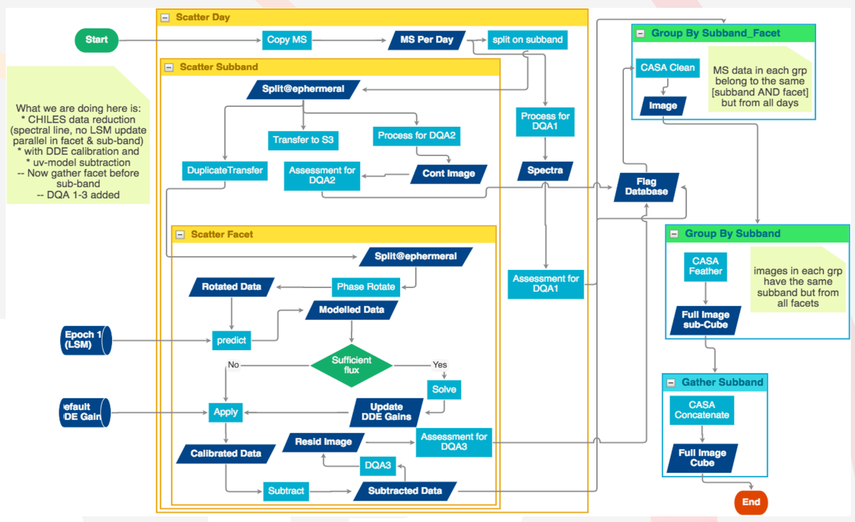
The Astronomer starts by composing **Logical Graph**, which represent high-level data processing capabilities, for example, “Image Visibility Data”. A completed logical graph expresses pipeline processing logic with resource-independent dataflow constructs and **Pipeline Components**. Astronomers build a logical graph by linking a set of selected pipeline components, each of which denotes a computational task wrapped in an executable container such as: a docker image, a binary executable, or a shell script. Each pipeline component is also characterised by workload metrics measured by the **Workload Analyser**.

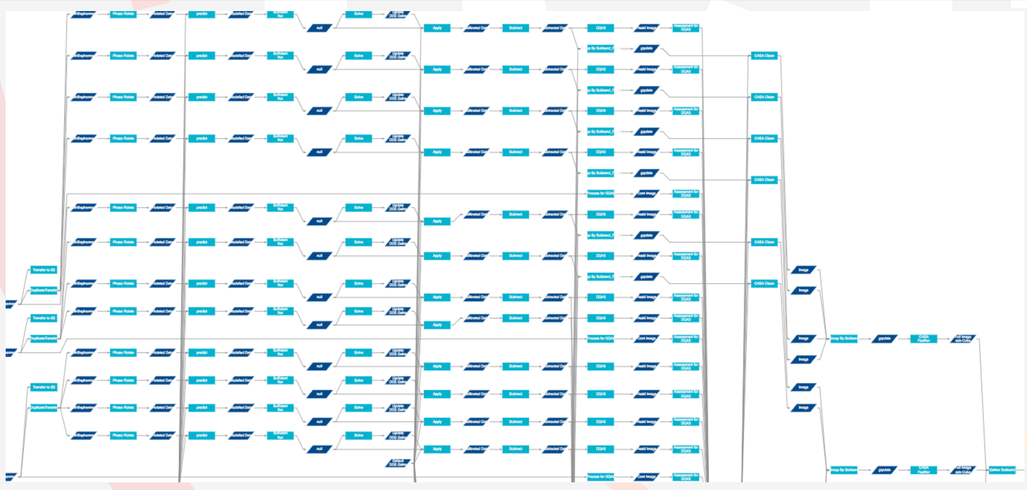
Next, *DALiuGE* translates a logical graph into a **Physical Graph Template (PGT)**, which defines all Drop “specifications” as stipulated by the logical graph but without creating concrete Drops or allocating any physical resources. The translation from a logical graph to PGT is automated and uses **Hardware Capability** information obtained from the **Resource Analyser**.

Using the **Hardware Availability** from the Resource Analyser, *DALiuGE* instantiates each Drop and associates it with an available resource unit. This transforms the physical graph template into a **Physical Graph**, consisting of inter-connected Drops mapped onto a given set of resources. *DALiuGE* deploys all the Drops onto these resources as per the location information stated in the physical graph.

Control flow constructs form the “skeleton” of the logical graph, and determine the final structure of the physical graph to be generated. *DALiuGE* currently supports the following flow constructs:

- ***Scatter*** indicates data parallelism. Constructs inside a Scatter construct represent a group of components consuming a single data partition within the enclosing Scatter.
- ***Gather*** indicates data barriers. Constructs inside a Gather represent a group of components consuming a sequence of data partitions as a whole.
- ***Group By*** indicates data resorting (e.g. corner turning in radio astronomy). The semantic is analogous to the GROUP BY construct used in SQL statement for relational databases, but applied to data Drops.
- ***Loop*** indicates iterations. Constructs inside a Loop represent a group of components and data that will be repeatedly executed/produced for a fixed number of times. Each Loop construct has a property named *num_of_iterations* that must be determined at logical graph development time, and that determines the number of times the loop is “unrolled”.

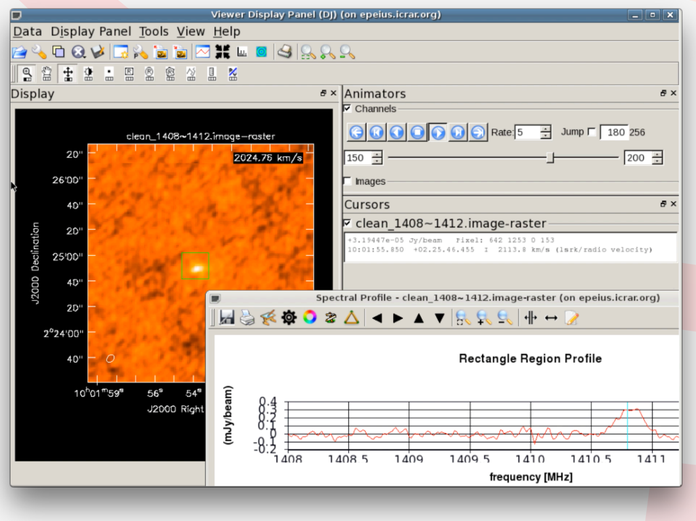


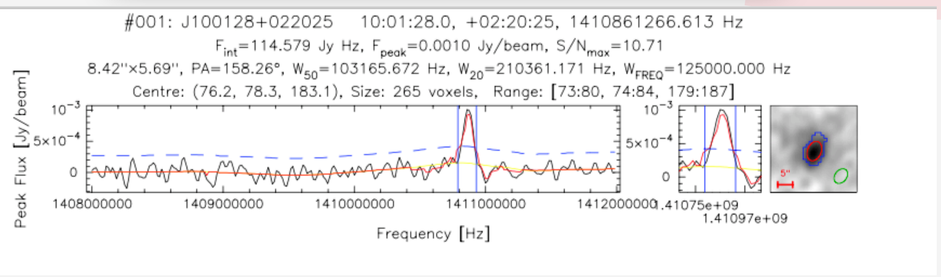


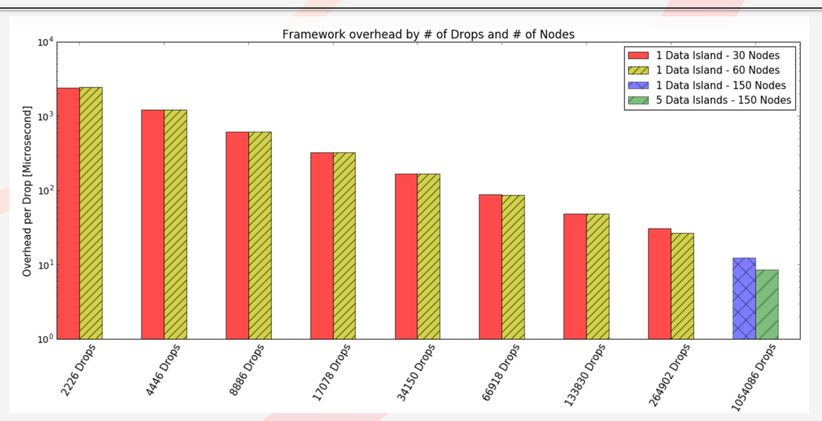
The Physical Graph describes how a collection of inter-connected Drops will be placed across multiple compute nodes to form a distributed execution plan — the physical graph. The nodes of a physical graph are Drops representing either data or applications. This establishes a set of reciprocal relationships between Drops:

- A data Drop is the input of an application Drop; on the other hand the application is a consumer of the data Drop.
- Likewise, a data Drop can be a streaming input of an application Drop in which case the application is seen as a streaming consumer from the data Drop’s point of view.
- Finally, a data Drop can be the output of an application Drop, in which case the application is the producer of the data Drop.

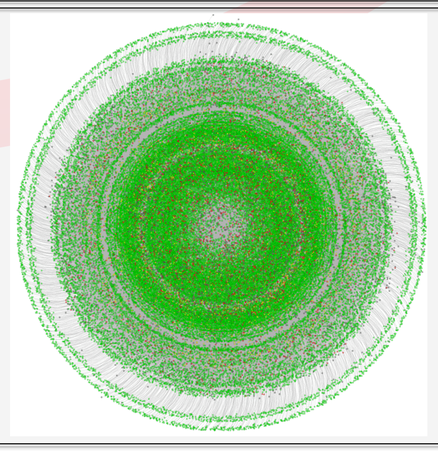
A CHILES galaxy at ~1410 MHz







Better than linear scale out to 1,000,000 drops on Tianhe-2



A physical graph running on Tianhe-2

- 500 compute nodes,
- 66,473 drops in total,
- 61,268 drops completed (green),
- 4,612 drops errors,
- 593 drops unfinished, due to those upstream errors

Curtin University

THE GOVERNMENT OF WESTERN AUSTRALIA

THE UNIVERSITY OF WESTERN AUSTRALIA

Look out for me,
Kevin VINSEN
kevin.vinsen@icrar.org