DALiuGE: A Science Workflow System Using Your Code! No change required!

Andreas Wicenec, Ryan Bunney, James Strauss, Rodrigo Tobar, Moritz Wicenec
International Centre for Radio Astronomy Research, University of Western Australia, Perth, Australia

Abstract

How do you present and discuss the functionality of a science pipeline/workflow? Chances are very high, that you will draw some connected boxes with a bit of explanation around them on a whiteboard or in a publication figure. The DALiuGE framework enables you to do just that by dragging and connecting abstract component symbols or, far more useful, by using automatically generated component symbols from your code in a web-based workflow graph editor called EAGLE$^{\pi}$. Graphs can be submitted for execution to the DALiuGE engine running on laptops or the biggest clusters in the world.

DALiuGE and EAGLE$^{\pi}$

Visual workflow development in your browser: No need to code. Just drag your components onto the canvas and connect them to build-up your workflow logic! Since EAGLE$^{\pi}$ does not actually touch or even know about the code, this can also be done before the code really exists and the workflow graph can be used as part of the software design work or even in presentations.

Share and Collaborate

Workflows can be shared between and developed by multiple people with full version control provided by standard GitHub and GitLab functionality.

dlg_paletteGen: Use Existing Code

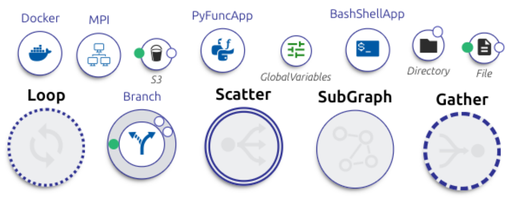
A stand-alone tool called *dlg_paletteGen* generates palettes of workflow components from any installed Python module. These palettes can be loaded into EAGLE$^{\pi}$ to construct your workflows. Newly developed code does not need to import anything specific to DALiuGE or use any special decorators. Just write your functions, classes and methods as you would do otherwise and run *dlg_paletteGen*!

Wide Support

The tool traverses the module tree and inspects all sub-modules, classes and functions and generates JSON descriptions of them. Existing code can be plain Python or full PyBind11 modules. Standard packages like Numpy, scipy or astropy can all be used without any code changes or adjustments. Running *dlg_paletteGen* on the complete AstroPy 6.1.7 package takes less than 30s and produces 24 palettes with a total of 13628 components!

Shell, Docker or MPI Anybody?

DALiuGE also supports more complex, stand-alone applications, which are typically called on the command line or as a docker/singularity container. Such applications can be arbitrarily complex and can even be complete MPI applications occupying thousands of ranks. On the other extreme side users can write in-line function code directly in EAGLE$^{\pi}$ using a small web-based code editor. This is very handy for small little helper functions for which writing a full stand-alone module is overkill.



A collection of some of the default components (top row) and constructs (bottom row).

A Simple Graph

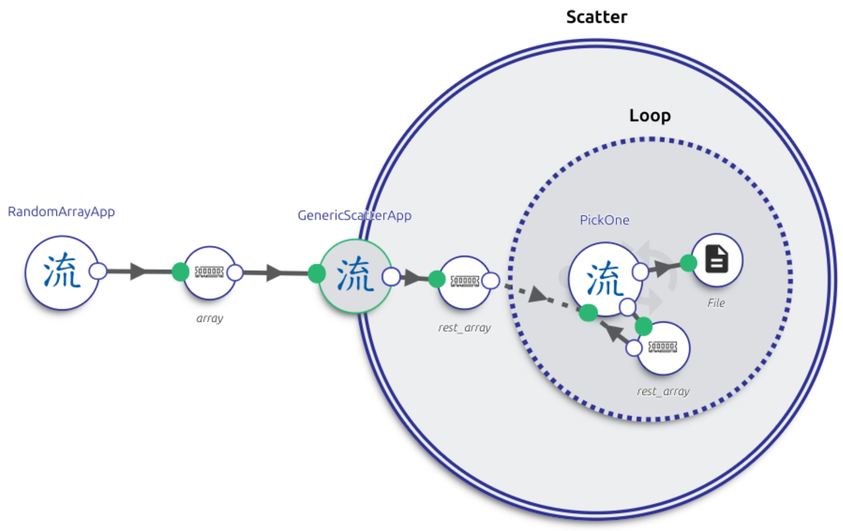


Figure 1: This simple workflow graph depicts a typical graph pattern: Splitting an input array into sub-arrays and looping over the elements of these sub-arrays. The components with the Liu symbols are Python application components. The slightly smaller data components are using icons indicating their storage type, Memory and File, respectively. The graph is also using two nested graph constructs, Scatter and Loop.

A Palette

Figure 2: The generated astropy.cosmology palette.

Summary

The stand-alone auto-generation tool *dlg-paletteGen* can introspect any installed Python module, including PyBind11 based code, to extract the signature of functions, methods and classes. It creates so-called component palettes, which in turn can be loaded into EAGLE$^{\pi}$ and used there to design workflow graphs. dlg-paletteGen also extracts any in-line documentation and argument descriptions, including types, which can then be examined in EAGLE$^{\pi}$. If no in-line documentation can be found, a LLM is used to generate it from the (Python) code. This enables the user to inspect the functionality of every single component of an existing graph or palette. Users can also add graph descriptions and comments. Palettes and graphs are version controlled in GitHub or GitLab repositories and can be shared and co-developed by a team of people. The best of all of that is that the rest of the DALiuGE system then allows you to translate and schedule such a graph and deploy and execute it to an engine running on a laptop, server, small cluster or the biggest HPC facilities in the world. Builtin, high-level components support scatter, gather, loop and branch constructs, enabling the development and execution of highly sophisticated, parallelised workflows.

Main landing page: https://daliuge.icrar.org
EAGLE$^{\pi}$ live: https://eagle.icrar.org
EAGLE$^{\pi}$ GitHub: https://github.com/ICRAR/EAGLE

DALiuGE GitHub: https://github.com/ICRAR/daliuge
dlg_paletteGen: https://github.com/ICRAR/dlg_paletteGen
Project template GitHub: https://github.com/ICRAR/daliuge-component-template