Science at the Speed of Thought: Enhancing Jupyter for "Human-in-the-loop" Supercomputing

Matt Henderson, Oliver Evans, Shreyas Cholia, Fernando Pérez

Lawrence Berkeley National Laboratory

Abstract

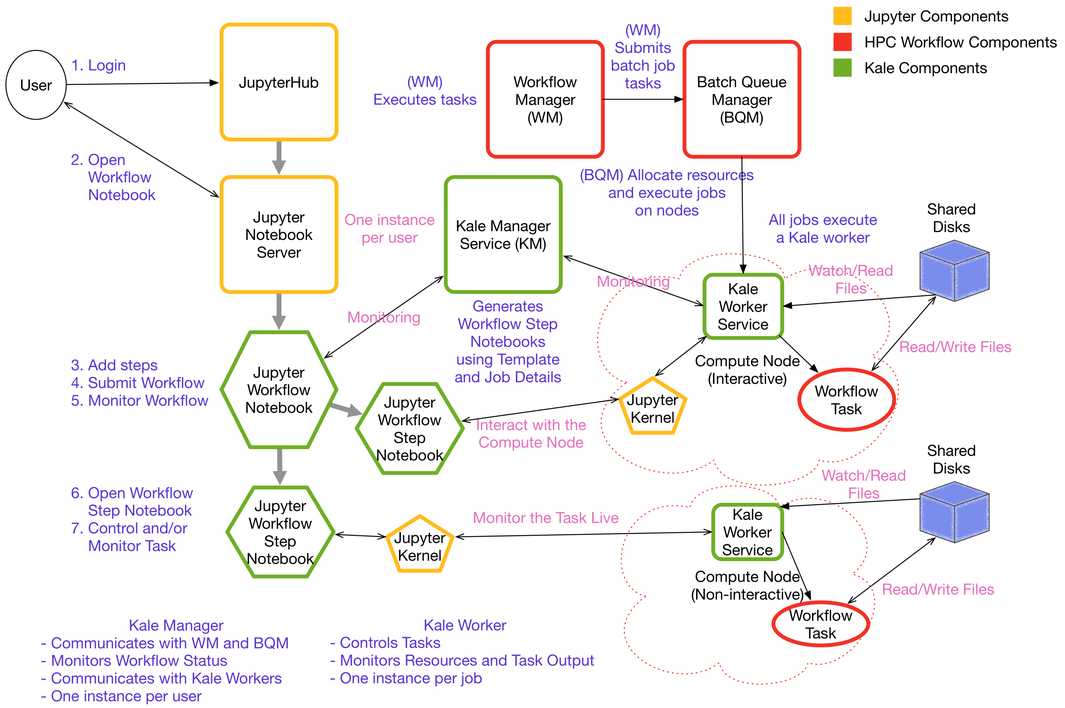
Abstract High Performance Computing (HPC) systems and workflows process and analyze data produced by large-scale experiments and simulations, such as first-principles materials structure calculations, supernovae simulations, and mass spectrometry image analysis. These largely focus on a non-interactive, asynchronous, batch execution process that can use thousands of cores and run for hours to days. Historically, these systems have not been designed to maximize the human utility and ease of interactive use, but rather to optimize for raw performance.

Simplifying and accelerating the mode of experimentation, which aligns with how scientists think and operate, is key to enhancing their productivity. This includes easy job submission and resubmission, introspection of jobs and their contents as they run, and easy ways of intercepting and manipulating data inputs and outputs for analysis and chaining of operations into pipelines. Introducing interactivity to scientific HPC applications and workflows provides a key missing human-in-the-loop capability to inspect the state of an execution in real time. The Jupyter architecture (kernels, Notebooks, widgets, etc.) provides a solid foundation to address these challenges, and is already familiar to many scientists, but is missing certain key ingredients.

Key Features

- Build & submit workflows in a notebook via Python API
- Jupyter Notebooks as interactive workflow steps
- Rich markdown workflow/task descriptions
- Submit batch jobs
- Use existing Workflow Software
- Interact with tasks via kernels
- Monitor HPC jobs via plots/logs
- Modular Widget Interface
- Run all or part of a workflow
- Export workflows to YAML

System Schematics



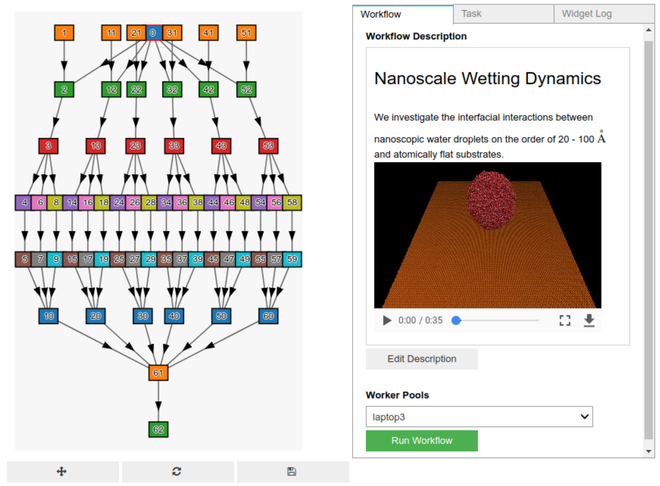
Define Workflows

```
example_workflow = kale.Workflow(name='example')
for i in range(10):
    example_workflow.add_task(
        kale.CommandLineTask(
            name='echo_test_{num}',
            command='echo "Test #{num}" >> {out_file} &&',
            output_files=["{out_file}"],
            params=dict(
                num=i,
                out_file="output.txt"
            )
        )
    )
```

- Define workflows via Python
- Simple API
- Refer to tasks by name
- Data provenance
- Easy parameter substitution

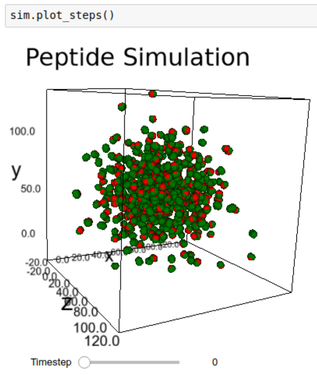
Visualize & Run

```
WorkflowWidget(droplet_wf, wpw)
```



Observe Live Output

```
sim.plot_steps()
```




```
t = TailWidget('out.txt')
t
```

Principles

Jupyter brings several key features to the world of computational science: Exploration, Narrative, and Interactivity

While these are powerful qualities, Jupyter presently operates at the level of an individual notebook in one language. We seek to harness the power of Jupyter to bring these qualities to full scientific workflows, which often involve many steps across a variety of languages.

Exploration

Scientists love to explore. But being constrained by computational resources, scientific computing tends to focus on maximizing computational efficiency, while sacrificing human involvement. By combining the raw power of high performance computing resources with the flexibility of Jupyter, we can go from idea to implementation in minutes, even on the biggest problems.

Narrative

Having Markdown and LaTeX right next to your code allows you to tell the full story of what's going on in a notebook in a clear and cohesive manner. With formulas, images, tables, and even movies embedded right in the document, communicating complex concepts is so much more feasible than through code comments alone. While sharing code is essential for open science and reproducibility, it usually doesn't go very far without context.

By bringing the power of narrative from the individual notebook to full workflows, we hope to make HPC workflows as easy to share effectively as small individual notebooks.

Interactivity

Being able to touch your data and models provides a deeper ability to understand them. The Jupyter widget ecosystem provides a wide range of interactive elements that allow for the visualization and exploration of high dimensional spaces. In HPC systems, interactivity has generally been at the bottom of the food chain. We hope to bridge that gap by providing an easy means by which to examine simulations and analysis codes as they run, and to be able to have interactive pieces of a scientific workflow, where you know that you're going to need human interaction in between computational tasks.

Dashboard Widgets

- Workflow DAG visualizer/runner
- Live updating plots/logs
- Local/remote worker creation
- Batch queue status
- Resource monitors
- SSH/REST authenticators