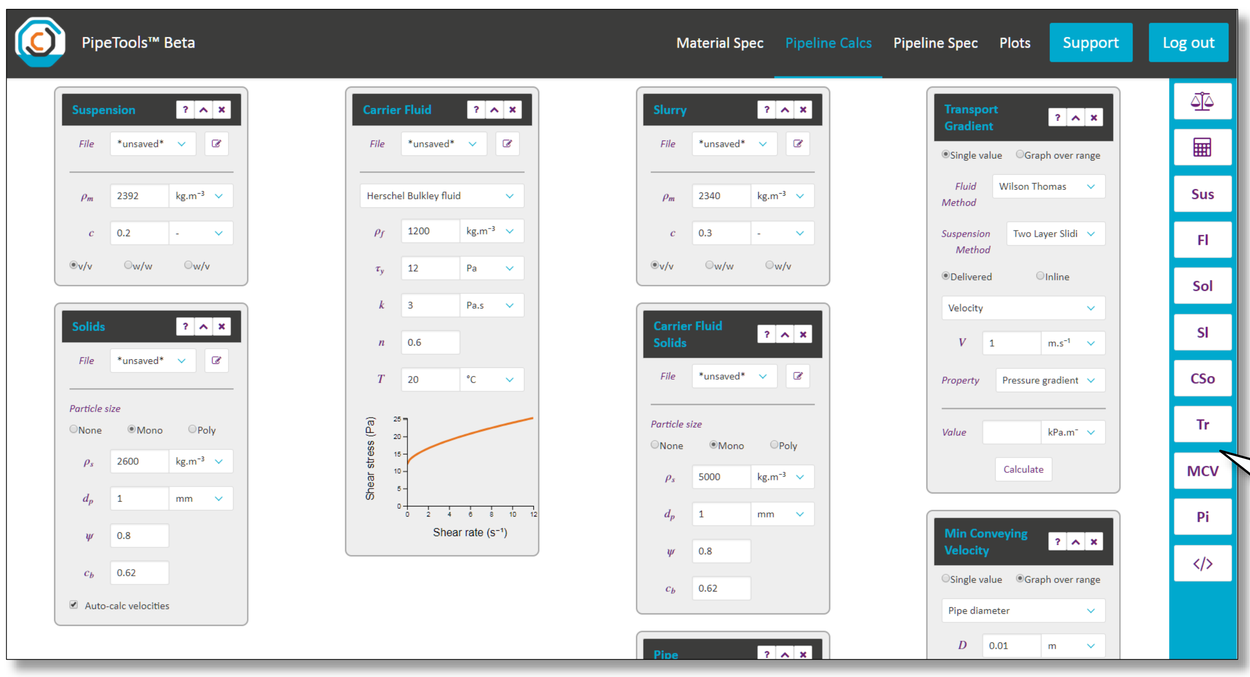
Transforming Research Code to A Modern Web Architecture: Pipetools

A Software Development Progress Note

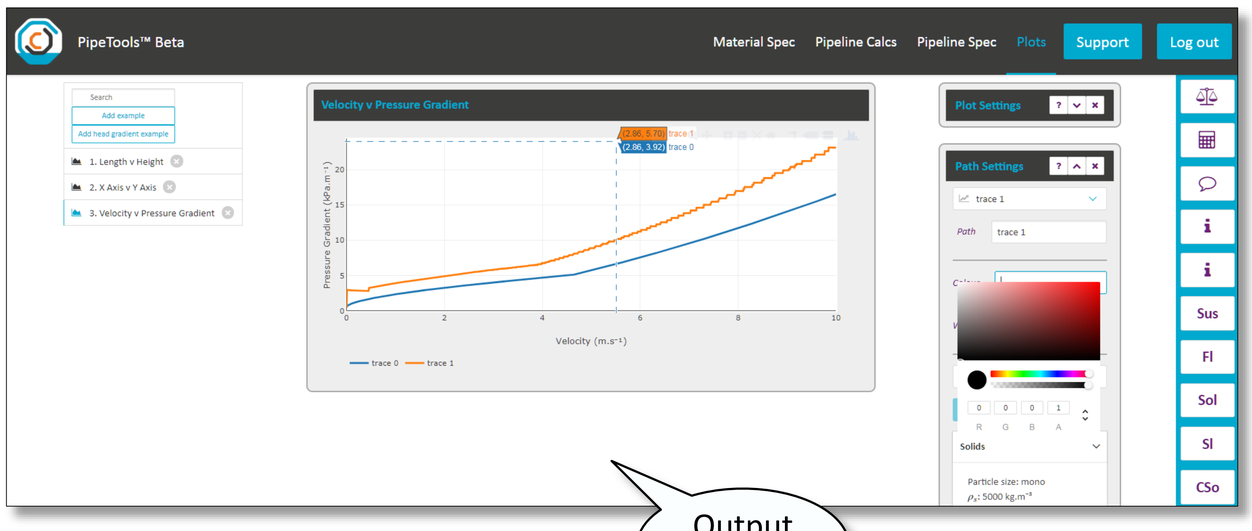
Paulus Lahur and Kieran Lomas

SCIENTIFIC COMPUTING, IMT
www.csiro.au

Pipetools started its life as a research code, written in C++ for Windows XP. It is a powerful tool that models the flow of suspensions and slurries (a mixture of fluids and solids) inside a pipeline. We aim to transform the code into a web application so users can access the tool from anywhere, with the interactive experience of a native app. The modern technology stack used in this project can be applied to code built on an aging platform to gain many of the advantages of current technologies without starting again from scratch.



Input window



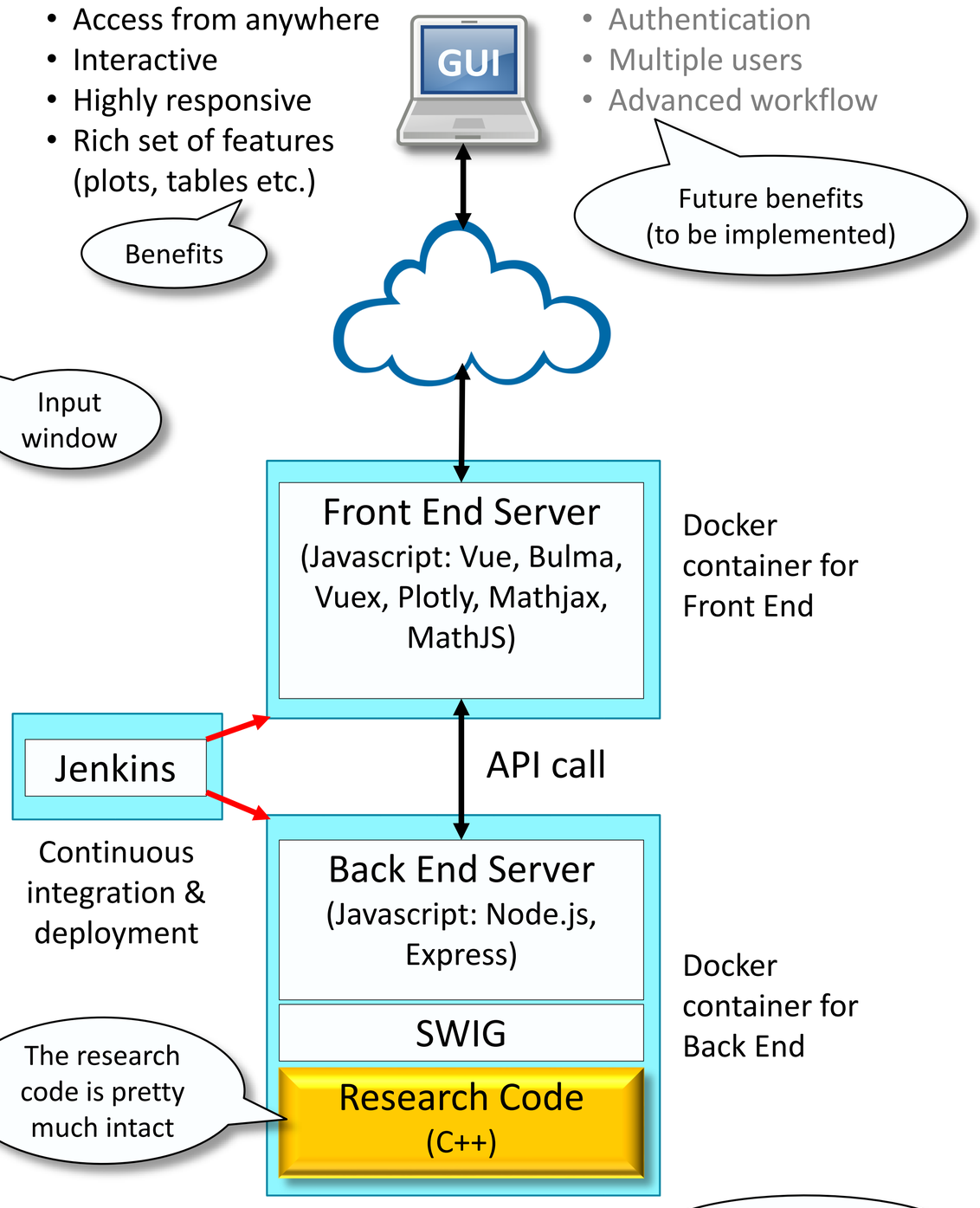
Output window

- Access from anywhere
- Interactive
- Highly responsive
- Rich set of features (plots, tables etc.)

Benefits

- Authentication
- Multiple users
- Advanced workflow

Future benefits (to be implemented)



The Current State

Research Code

Written in **C++** the base code is already mature and in production. Additional features are planned down the line

Infrastructure

Utilising microarchitecture technologies including: **Docker** (containerisation), **Kubernetes** (container orchestration) and **Jenkins** (continuous integration and deployment).

Back End

The interface between the research code and front end. Research code functionalities are being implemented as an API. Key technologies: **SWIG** (interface between C++ and Javascript), **Node.js**, **Express** (Javascript server solution)

Front End

Modularised GUI solution, using **Vue** and **Bulma** to compose reusable components. Key technologies: **Vuex** (state management), **Plotly**, **Mathjax** (symbols), **MathJS** (calculations)

From user click …

GUI

Calculate

API call from Front End to Back End

```
http://host_name:port_number/calc/dpdx
```

Javascript on the Back End

```
const funcMap = {
    'dpdx': dpDx,
…
};
…
result[i] = pipe.dPdx(model.suspension, parseFloat(x)) / 1000.0;
```

SWIG interface file

```
%module "pipetools“
%{
#include "include/Pipe.h“
%}
…
%include "include/Pipe.h“
```

C++ header file

```
virtual double dPdx(CSuspension *s, double v);
```

… to the research code

FOR FURTHER INFORMATION

- **Paulus Lahur**: paulus.lahur@csiro.au
- **Kieran Lomas**: kieran.lomas@csiro.au

ACKNOWLEDGEMENTS

- Research code: **Lionel Pullum** (Mineral Resources, Clayton VIC)
- Project manager: **Andrew Chryss** (Mineral Resources, Clayton VIC)
- Team lead on IMT side: **Daniel Collins** (IMT, Kensington WA)
- Front End: **Kieran Lomas** (IMT, Clayton VIC)
- Back End: **Paulus Lahur, Sam Moskwa** (IMT, Clayton VIC)
- Infrastructure: **Dylan Graham, Andrew Spiers, Sam Moskwa** (IMT, Clayton VIC)