Benchmarking In Google Cloud: Google Cloud HPC-Toolkit + Ramble

Doug Jacobsen - HPC Software Engineer

Google Cloud

Outline

- Application Benchmarking
    - Benchmarking at Google
    - Typical Workflow
    - Tools Introduction
- Ramble
- Workflow with Ramble
- Summary

Application Benchmarking

01

Google Cloud

Google + HPC

Our team consists of:

- HPC Focused Application Engineers
- General Software Engineers
- Scientific Domain Experts

Our team focuses on:

- Analysing and improving application performance
- Responding to Customer RFPs
- Developing tooling to support the two above tasks

Reproducibility and Productivity

It is desirable to make computational experiments as reproducible as possible.

Reproducible experiments allow:

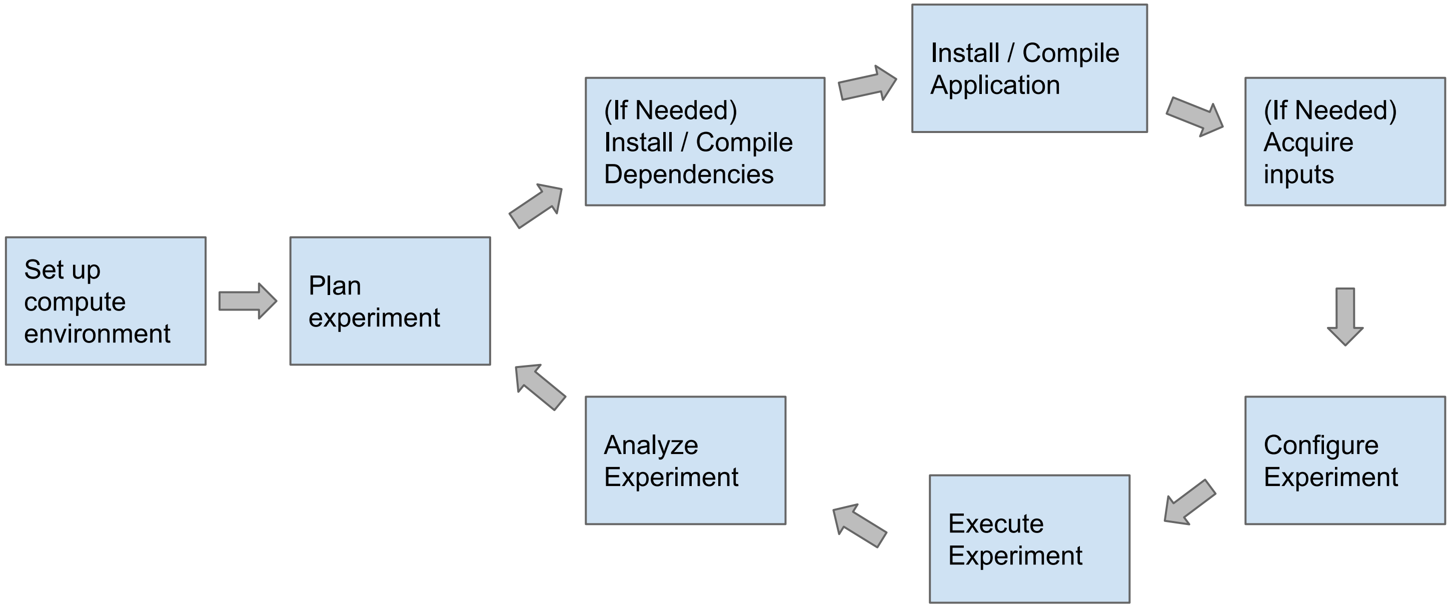
- Lower effort for re-running an experiment at a future time
- Easier knowledge transfer
- Increased productivity, by more rapidly exploring parameter spaces

Without reproducible experiments:

- Every environment requires the same effort
- Effort increases as a function of time between subsequent experiments
- Knowledge is lost when application experts change roles

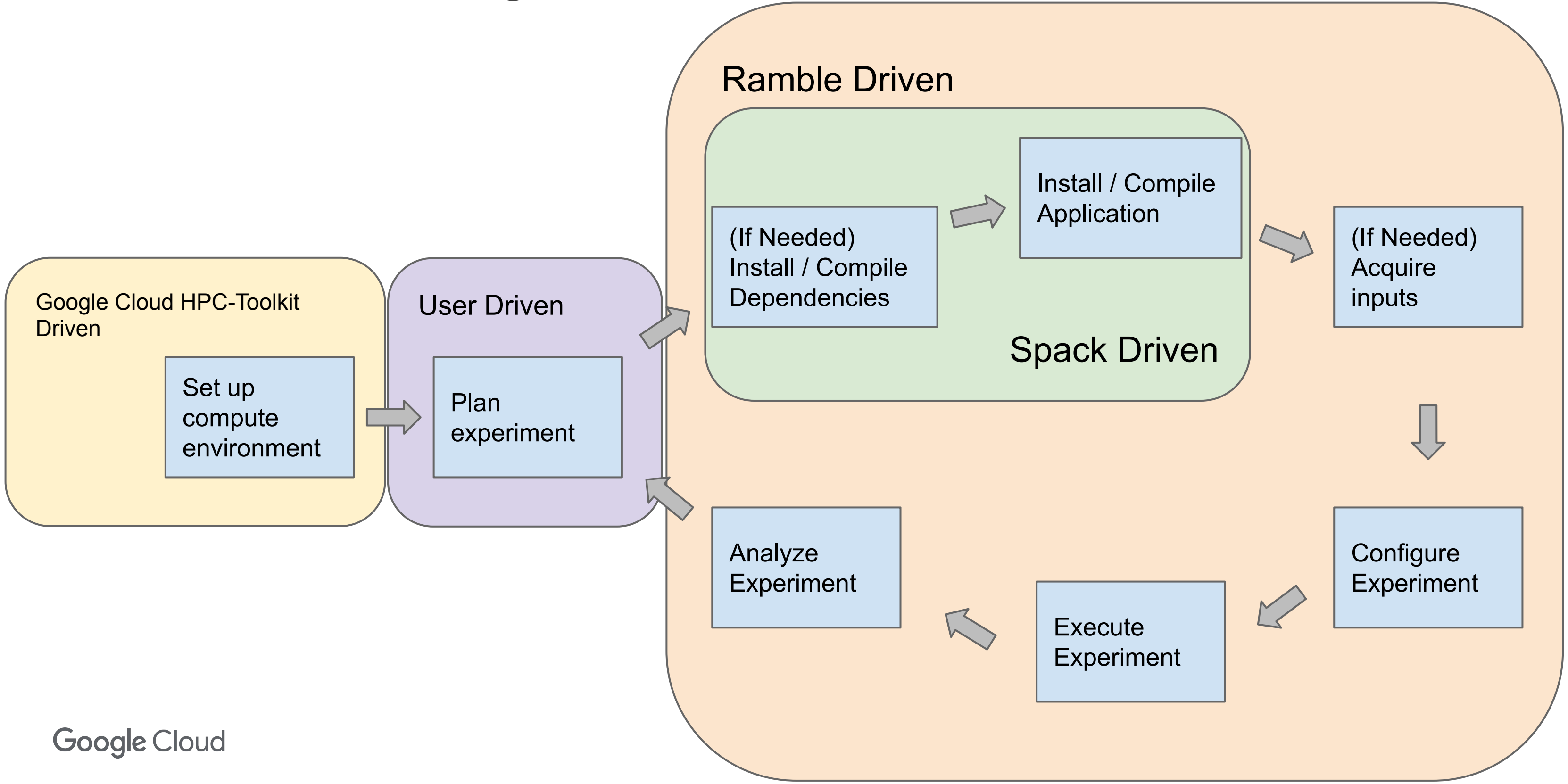
The goal of our tools is to improve reproducibility, regardless of the purpose of the experiment.
(i.e. performance analysis, scientific parameter study, platform evaluation, etc…)

Benchmarking Workflow



Set up compute environment
Plan experiment
(If Needed) Install / Compile Dependencies
Install / Compile Application
(If Needed) Acquire inputs
Configure Experiment
Execute Experiment
Analyze Experiment

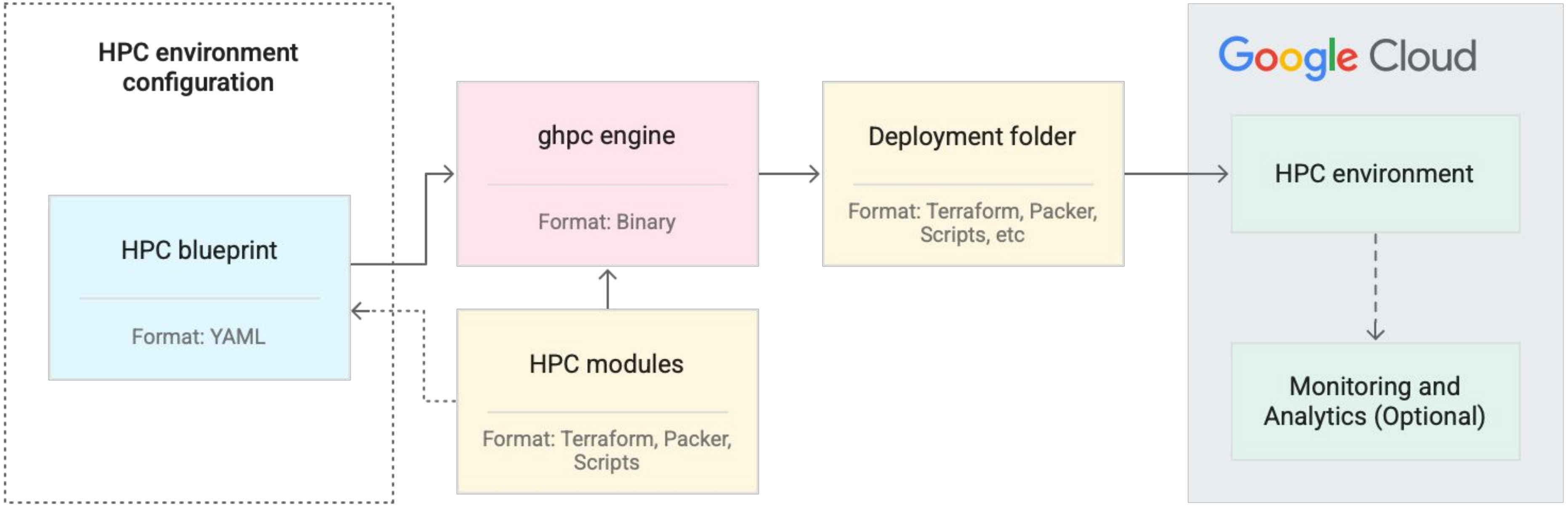
Benchmarking Workflow on GCP


Google Cloud HPC-Toolkit Driven
Set up compute environment
User Driven
Plan experiment
Ramble Driven
(If Needed) Install / Compile Dependencies
Install / Compile Application
Spack Driven
(If Needed) Acquire inputs
Configure Experiment
Execute Experiment
Analyze Experiment

Google Cloud

Cloud HPC Toolkit

https://cloud.google.com/hpc-toolkit


HPC environment configuration
HPC blueprint
Format: YAML
ghpc engine
Format: Binary
HPC modules
Format: Terraform, Packer, Scripts
Deployment folder
Format: Terraform, Packer, Scripts, etc
Google Cloud
HPC environment
Monitoring and Analytics (Optional)

02

Ramble

Google Cloud

What is Ramble?

Ramble is:

- Written in python
- Multi-platform
- Heavily based on Spack
- An experimentation framework
- Open Source: https://github.com/GoogleCloudPlatform/ramble

Ramble provides:

- A flexible templating engine
- A domain specific language for describing an application and its workloads
- Standardized definitions of how workloads are configured and executed

Why did we make a new framework?

Several options exist today as benchmarking frameworks, some examples are:

- ReFrame: (https://reframe-hpc.readthedocs.io/en/stable/)
- Buildtest: (https://buildtest.readthedocs.io/en/devel/)
- Pavilion: (https://github.com/lanl/Pavilion)
- Most compute centers use one of these, or have a custom testing harness

Ramble differs from these frameworks by:

- Abstracting at the experiment level rather than the system level
- Not specializing for system acceptance

Additionally, we developed Ramble to be an application / experimentation complement to Spack.

Standardized Application Definitions

As Ramble was heavily based off of Spack, both tools provide a domain specific language for standardizing their building blocks.

In Spack, these are package definitions.

In Ramble, these are application definitions.

Some of Ramble’s application language features include:

- input_file - Create reference to an input that the application can use
- executable - Define a command an experiment of this application might use
- figure_of_merit - Define a metric that will be extracted when analyzing experiments
- success_criteria - Define criteria to make an experiment as successful

Gromacs Example

```
from ramble.appkit import *

class Gromacs(SpackApplication):
    '''Define a Gromacs application'''
    name = 'gromacs'

    tags = ['molecular-dynamics']

    default_compiler('gcc9', base='gcc', version='9.3.0')
    mpi_library('impi2018', base='intel-mpi', version='2018.4.274')
    software_spec('gromacs', base='gromacs', version='2020.5', compiler='gcc9',
                  mpi='impi2018')

    executable('pre-process', 'gmx_mpi grompp ' +
               '-f {input_path}/{type}.mdp ' +
               '-c {input_path}/conf.gro ' +
               '-p {input_path}/topol.top ' +
               '-o exp_input.tpr', use_mpi=False)
    executable('execute-gen', 'gmx_mpi mdrun -notunepme -dlb yes ' +
               '-v -resethway -noconfout -nsteps 4000 ' +
               '-s exp_input.tpr', use_mpi=True)

    input_file('water_gmx50_bare',
               url='https://ftp.gromacs.org/pub/benchmarks/water_GMX50_bare.tar.gz',
               description='')
```

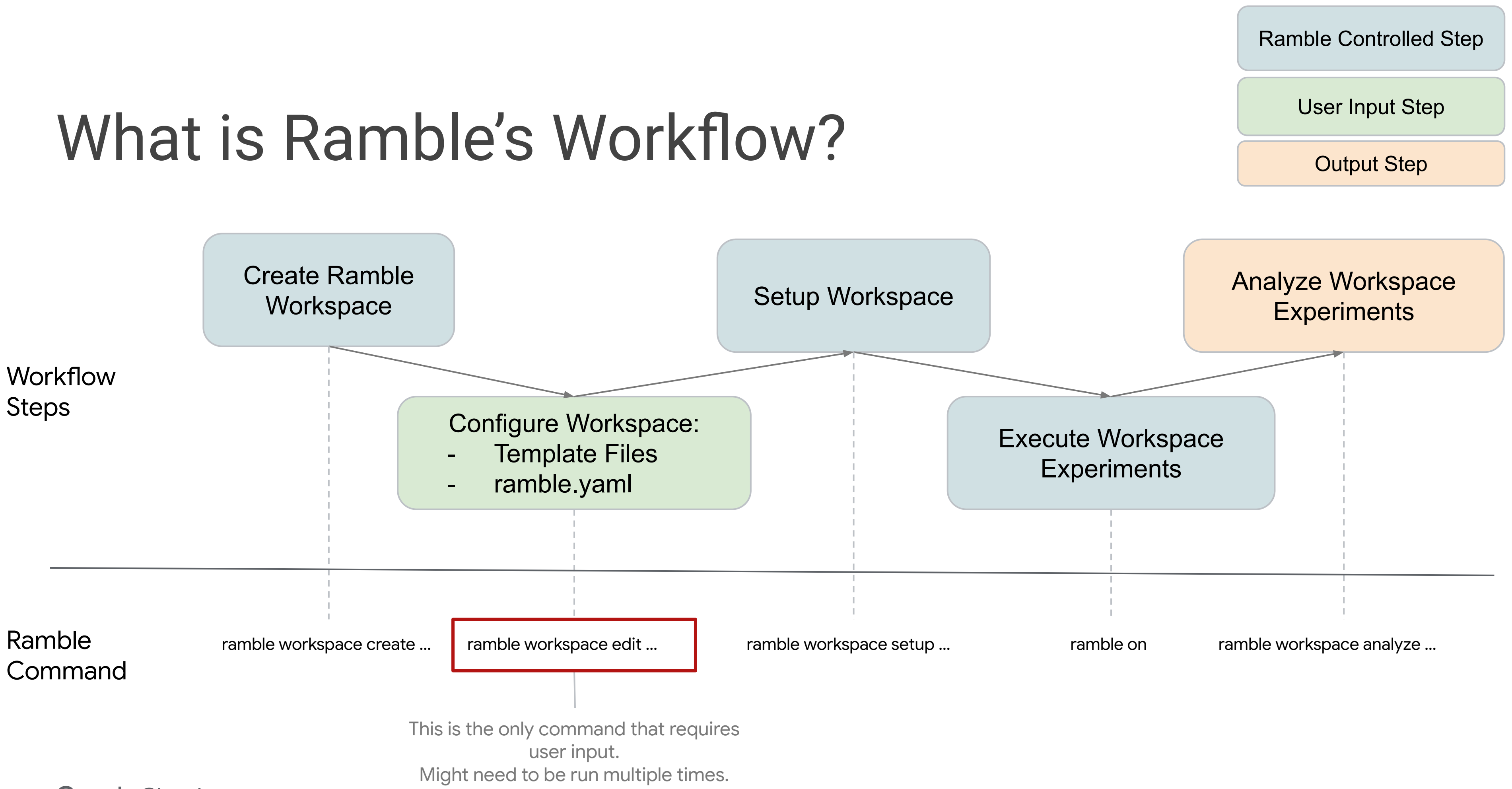
```
workload('water_gmx50', executables=['pre-process', 'execute-gen'],
         input='water_gmx50_bare')

workload_variable('size', default='1536',
                  description='Workload size',
                  workloads=['water_gmx50'])
workload_variable('type', default='pme',
                  description='Workload type. Valid values are ''pme'',''rf''',
                  workloads=['water_gmx50'])
workload_variable('input_path', default='{water_gmx50_bare}/{size}',
                  description='Input path for water GMX50',
                  workload='water_gmx50')

figure_of_merit('Core Time', log_file='{experiment_run_dir}/md.log',
                fom_regex=r'\s+Time:\s+(?P<core_time>[0-9]+\.[0-9]+).*',
                group_name='core_time', units='s')

figure_of_merit('Wall Time', log_file='{experiment_run_dir}/md.log',
                fom_regex=r'\s+Time:\s+[0-9]+\.[0-9]+\s+' +
                          r'(?P<wall_time>[0-9]+\.[0-9]+).*',
                group_name='wall_time', units='s')

figure_of_merit('Nanosecs per day', log_file='{experiment_run_dir}/md.log',
                fom_regex=r'Performance:\s+' +
                          r'(?P<ns_per_day>[0-9]+\.[0-9]+).*',
                group_name='ns_per_day', units='ns/day')
```

03

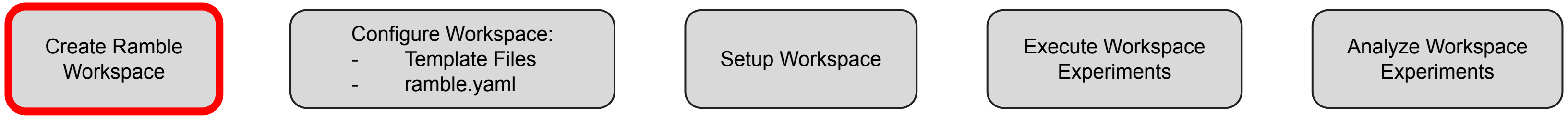
Workflow with Ramble

Google Cloud

What is Ramble's Workflow?


Ramble Controlled Step
User Input Step
Output Step
Workflow Steps
Create Ramble Workspace
Configure Workspace:
- Template Files
- ramble.yaml
Setup Workspace
Execute Workspace Experiments
Analyze Workspace Experiments
Ramble Command
ramble workspace create ...
ramble workspace edit ...
ramble workspace setup ...
ramble on
ramble workspace analyze ...
This is the only command that requires user input.
Might need to be run multiple times.

Google Cloud

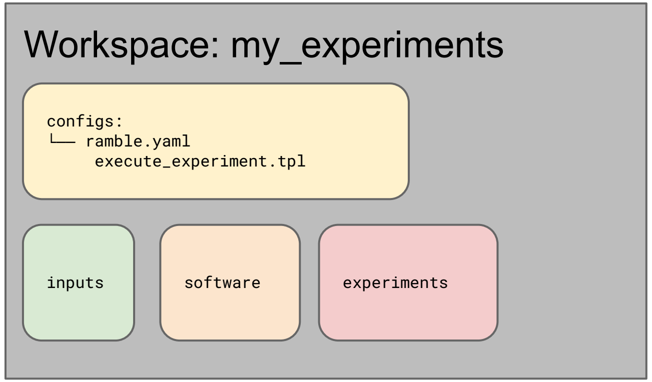


Ramble Workspace:

A workspace is a self contained directory, that contains:

- software environments
- input files
- experiments

Workspaces are independent, and should represent a set of independent experiments you want to execute.



Google Cloud

Create Ramble Workspace

Configure Workspace:
- Template Files
- ramble.yaml

Setup Workspace

Execute Workspace Experiments

Analyze Workspace Experiments

Workspace Config:

Ramble’s workspace config YAML can generate many experiments with only a little syntax.

```
applications:
  wrfv3:
    variables:
      processes_per_node: [30, 56]  } Define two vectors, of length 2.
      partition: [‘c2’, ‘c2d’]
      n_ranks: ‘{processes_per_node}*{n_nodes}’
    workloads:
      CONUS_2p5km:
        experiments:
          scaling_study_{partition}_{n_nodes}nodes:
            variables:
              n_nodes: [1, 2, 4, 8, 16, 32]} Define one vector, of length 6
            matrix:   } Define one matrix, using the n_nodes vector. Shape is 1x6.
            - n_nodes
```

- Vectors that are not used by a matrix are zipped together (and have to be the same length)
- Zip of vectors are crossed with any matrices
- Result is: 1x6 (matrix) x 2 (vectors) = 12, where each index is a 3-element tuple: (processes_per_node, partition, n_nodes)

Google Cloud

Create Ramble Workspace

Configure Workspace:
- Template Files
- ramble.yaml

Setup Workspace

Execute Workspace Experiments

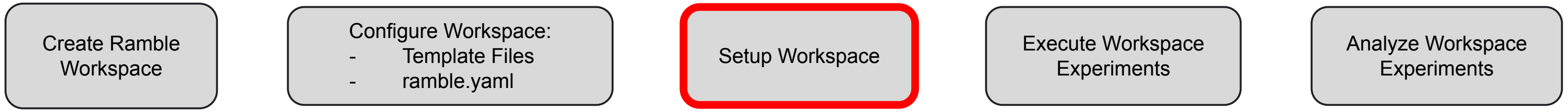
Analyze Workspace Experiments

Workspace Config:

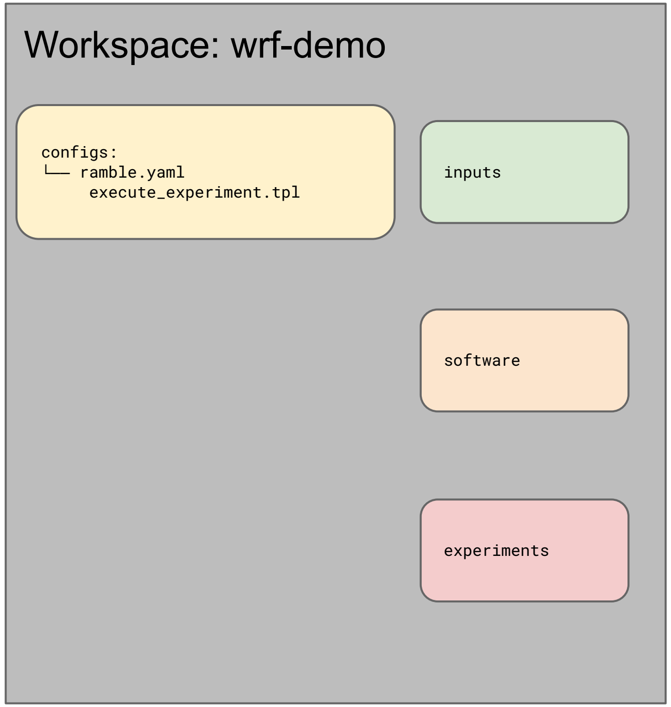
In addition to the YAML config, Ramble has a template engine to render scripts:

```
#!/bin/bash

{spack_setup}

{command}
```

```
#!/bin/bash

source <path/to/spack>
spack env activate <path/to/env>

cp <inputs>/* <experiment_dir>/.
mpirun -n <n_ranks> wrf.exe
```



Workspace Setup:

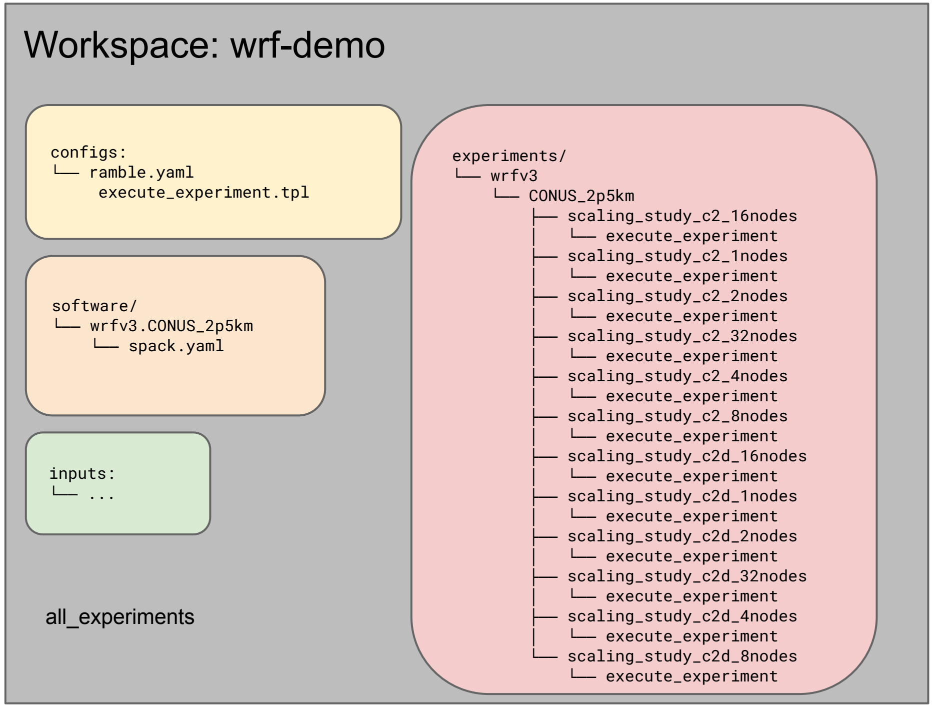


Execute input phases:
- Determine which inputs are necessary
- Download them
- Define variables for referring to these inputs

Execute software phases (Only used on SpackApplications currently):
- Install necessary compilers
- Create spack environments for the experiments
- Using spack, install the required software
- Define variables for referring to the spack environments

Execute experiment creation phases:
- Create experiment execution directories
- Define variables specific to the experiment
- Render any *.tpl files into the execution directories
- Append the experiment to $workspace/all_experiments

Execute Experiments:


Workspace: wrf-demo

```
configs:
└── ramble.yaml
     execute_experiment.tpl
```

```
software/
└── wrfv3.CONUS_2p5km
    └── spack.yaml
```

```
inputs:
└── ...
```

all_experiments

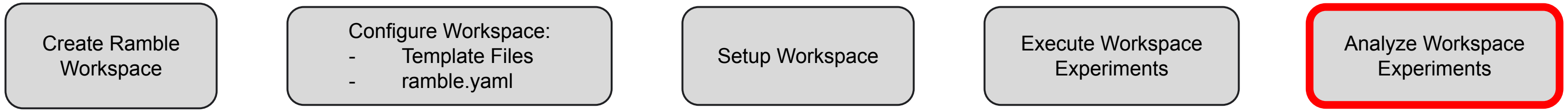
```
experiments/
└── wrfv3
    └── CONUS_2p5km
        ├── scaling_study_c2_16nodes
        │   └── execute_experiment
        ├── scaling_study_c2_1nodes
        │   └── execute_experiment
        ├── scaling_study_c2_2nodes
        │   └── execute_experiment
        ├── scaling_study_c2_32nodes
        │   └── execute_experiment
        ├── scaling_study_c2_4nodes
        │   └── execute_experiment
        ├── scaling_study_c2_8nodes
        │   └── execute_experiment
        ├── scaling_study_c2d_16nodes
        │   └── execute_experiment
        ├── scaling_study_c2d_1nodes
        │   └── execute_experiment
        ├── scaling_study_c2d_2nodes
        │   └── execute_experiment
        ├── scaling_study_c2d_32nodes
        │   └── execute_experiment
        ├── scaling_study_c2d_4nodes
        │   └── execute_experiment
        └── scaling_study_c2d_8nodes
            └── execute_experiment
```

After setting up a workspace, the experiments can be executed using:

- $workspace/all_experiments
- ramble on (with an activated workspace)

Depending on the ramble.yaml and execute_experiment.tpl this:

- Executes experiments sequentially
- Submits experiments to a workload manager


Create Ramble Workspace
Configure Workspace: - Template Files - ramble.yaml
Setup Workspace
Execute Workspace Experiments
Analyze Workspace Experiments

Analyze Workspace:

After experiments are executed, Ramble can extract their figures of merit.

This is done through:

```
ramble workspace analyze
```

Ramble will process the output files described in an experiment's application.py, and extract success criteria and figures of merit.

These are then written to a file (e.g. results.txt) in the provided format (can be text, yaml, or json).

Summary

04

Google Cloud

Summary:

Reproducibility is an important aspect of benchmarking. It helps lower the barrier to entry for new applications, and increase engineer / scientist productivity.

We rely on HPC-Toolkit to create reproducible HPC cluster on Google Cloud. This provides something of an Infrastructure as Code solution.

We developed and introduced Ramble to provide standardized, reproducible experiments.

Explore how useful it is for your workflow, and engage us on GitHub!

Thank you

https://github.com/GoogleCloudPlatform/ramble