A Unified Lab Notes Framework for Reproducibility on Exascale Systems
(An update)

R. Marshall

CUP-ECS Annual Review

Sep. 29, 2022

Motivation

- HPC systems are increasingly diverse, with various
  - hardware configurations
  - firmware versions
  - operating systems
  - installed software versions
  - communication media …
- By the time the results from an experiment can be published, some or all of the components of the environment could have changed.

Motivation (cont.)

- While a number of tools exist to aid in reproducibility, there is still a gap in experimental integrity that the researcher is often left to close manually:
  - Input files and runtime parameters
  - Output content and format
  - Method of connecting dependencies with configuration management and program output

Contributions

- Proposes a framework for managing experiments on large-scale HPC systems
- Enhances productivity for the research team
- Promotes experimental integrity and reproducibility
- Provides minimal infrastructure for greater flexibility

System Discovery

- Select Application
- Select Target System
- Collect System Info
- Login/Shell Access

Platform Establishment

- Get OS Details
- ID Runtime Systems
- Get Comm. Details
- Get Compiler Details
- Update Env. Variables

Application Build

- Read Exp. Config File
- Obtain Source Code
- Resolve Dependencies
- Build the Application

Deploy/Run Experiment

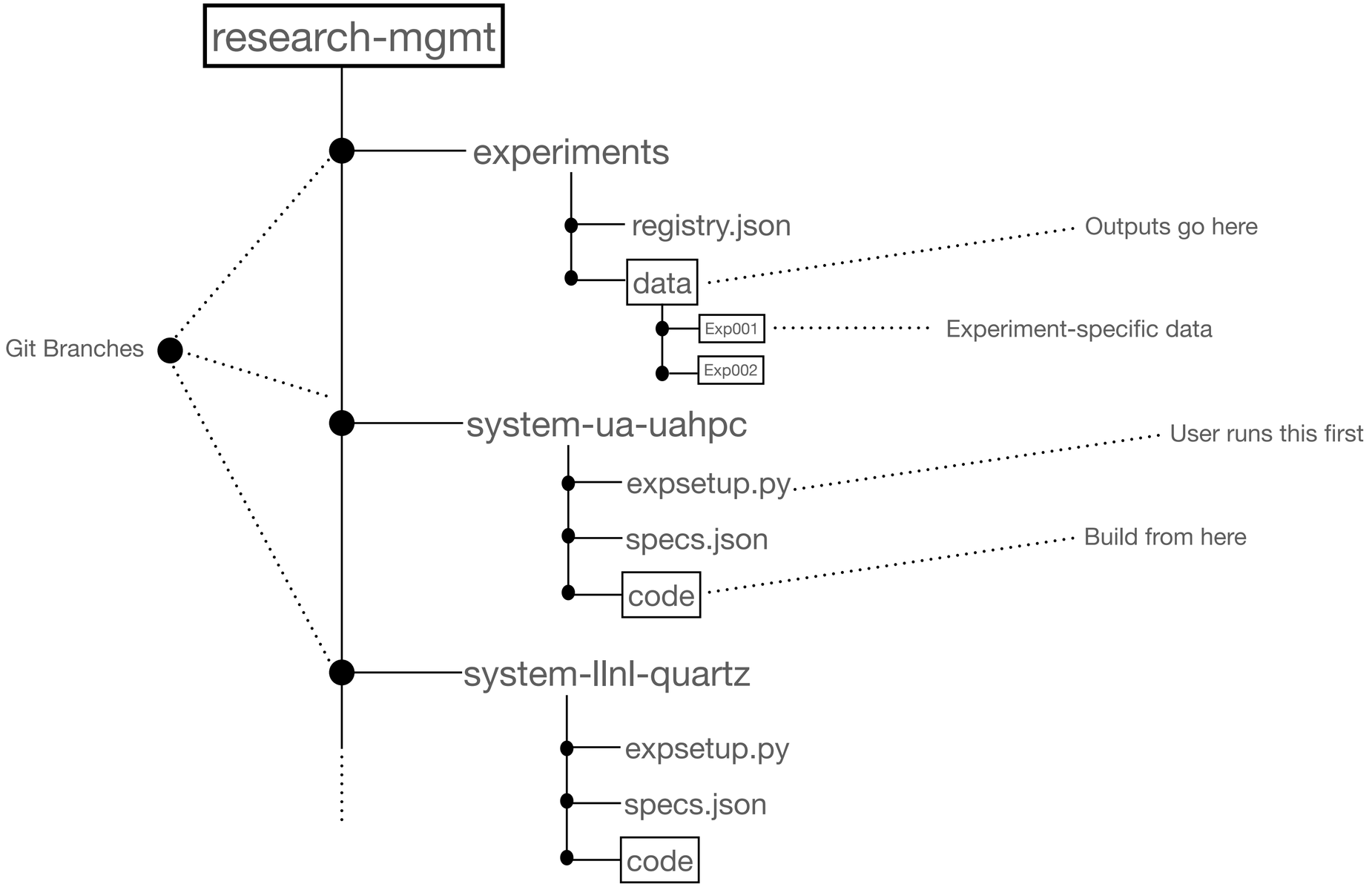
- Update Env. Variables
- Get Input/Exec. Params
- Choose Input Data Srcs.
- Generate Job Script
- Submit Job

Process Exp. Results

- Choose Output Staging Loc.
- Collect Results
- Analyze Results
- Preserve Results and Analysis

Efforts Since Feb. Update

- “Implementation by necessity” of two use cases
  - CLAMR (modified branch) for multiple systems
    - LLNL Lassen
    - LLNL Quartz
  - HOSS for LANL systems
    - LANL Darwin
    - CPU versus various GPU versions via OpenACC
- Development of repository layout (workflow and data)
- Publication targeted for IPDPS 2023


research-mgmt
experiments
registry.json
Outputs go here
data
Exp001
Experiment-specific data
Exp002
Git Branches
system-ua-uahpc
User runs this first
expsetup.py
specs.json
Build from here
code
system-llnl-quartz
expsetup.py
specs.json
code

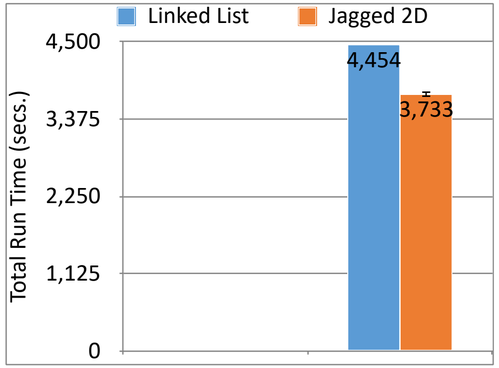
CLAMR Workflow

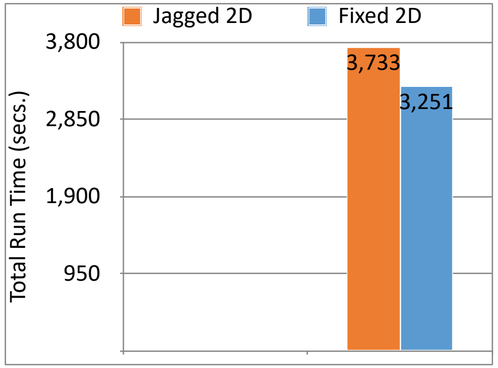
- Basic build and run scenario, but for multiple systems
- Discovery
  - System info by python script -> json -> reframe-settings.py
- Platform
  - Modules needed are stored in env. variables -> reframe
  - *lrun* for lassen, *srun* for quartz
- Build
  - Clone source repository
  - Use spack to check dependencies
  - Invoke compilers from env. variables and reframe-settings.py
- Deploy
  - Submit job scripts generated by reframe
- Results
  - Use python script to collect from stdout
  - Write to output directory and git commit (push optional)

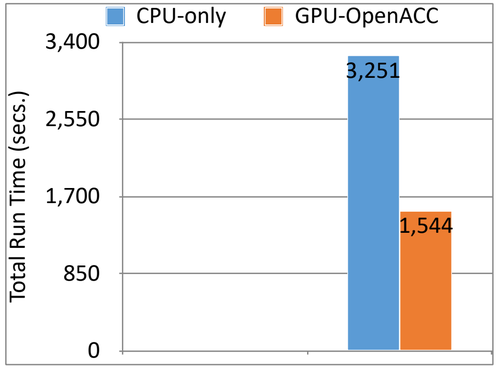
HOSS Workflow

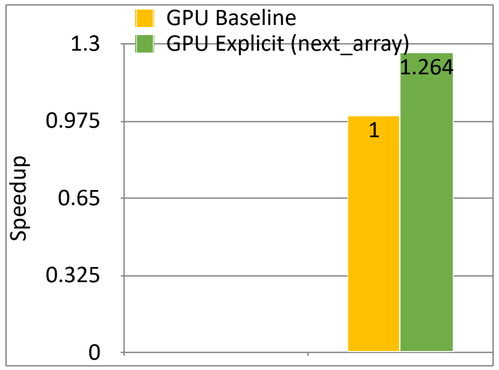
- Make minor changes to source code, test performance
- Discovery
  - System info by python script -> write to json -> reframe-settings.py
- Platform
  - Modules needed are stored in env. variables from previous stage -> collect via reframe
- Build
  - Clone source repository **(or fetch from local)**
  - Load modules, invoke compilers from env. variables and reframe-settings.py
- Deploy
  - **Change to input directory**
  - Submit job scripts generated by reframe
- Results
  - Use shell script to move output files to output directory
  - **Use python script to validate results (integrate with reframe)**
  - **Collect performance data from SLURM (email or job script)**

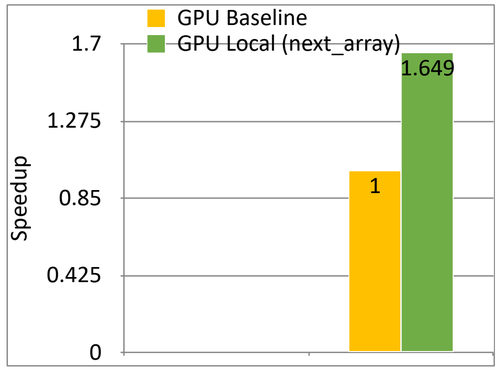
HOSS Optimizations - Results (context-free examples)











Lessons Learned

- Must consider export control / classification level when implementing framework
- Framework scripts/data must exist completely separate from exp. program build/exec structure.
- Force version numbers when using module systems.

Remaining Work

- Clear HOSS-related scripts for limited access
- Set repository for release, long form documentation
- Stay with paper through publication
- Introduce new team member?