Modern Flight Software CI for Diverse Platforms:

A Demo Using Slurm on the Georgia Tech CRNCH Rogues Gallery Testbed

Sterling Peet · Research Technologist II
Jeffrey Young, PhD · Rogues Gallery Director, Principal Research Scientist
Center for Research into Novel Computing Hierarchies (CRNCH)

Funded by NSF Rogues Gallery (CNS-2016701)

Georgia Tech

Overview

- Motivation
- Rogues Gallery Testbed
  - Achievements and Successes
  - Unique Capabilities
- CI/CD Workflow
  - GitHub Actions
  - Slurm scheduler
  - Example Workflow
- Demo CI/CD Build Process
  - Heterogeneous CI workflows
  - Additional Tools and Utilities
- Questions

Georgia Tech

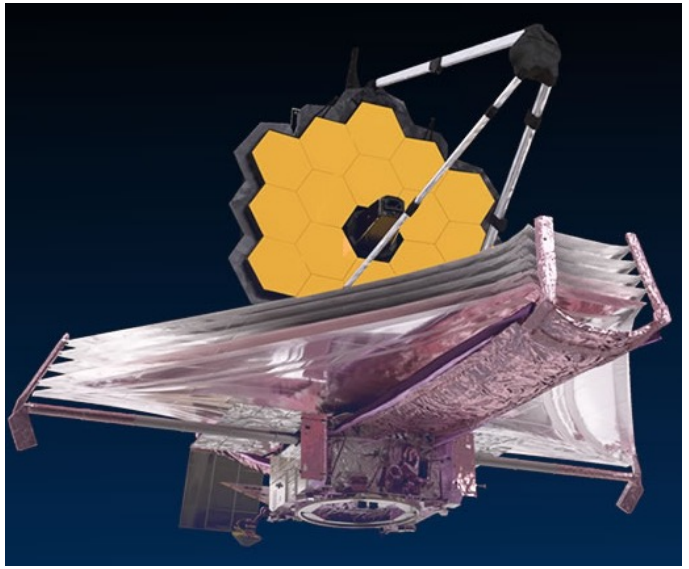
Motivation – The Future of Flight Software

Flight software is the core of current and future space missions

- Frameworks like F Prime and cFS are open-source platforms that largely target similar, embedded style processors

However, the explosion in small-format hardware offers both opportunities and challenges for flight software design

- The latest Arm and RISC-V processors and FPGAs offer customizable designs to further mission capabilities



Georgia Tech

Motivation – The Future of Flight Software

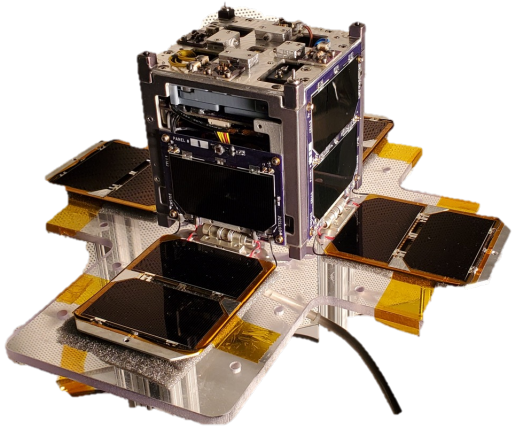
Designing flight software frameworks for next-generation hardware requires:

- Access to a variety of hardware targets that are suitable for space missions

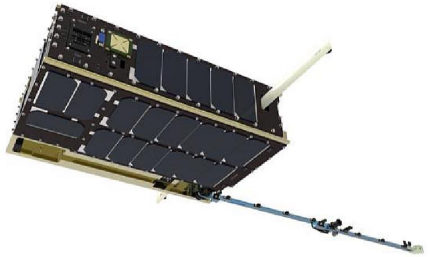
  → **The CRNCH Rogues Gallery testbed provides a low-effort environment to investigate these platforms**

- A robust framework for developing and testing flight software on this next hardware

  → **The usage of Slurm and continuous integration (CI/CD) testing for novel hardware enables a productive software development and testing environment**



GT-1



Dellingr

Georgia Tech

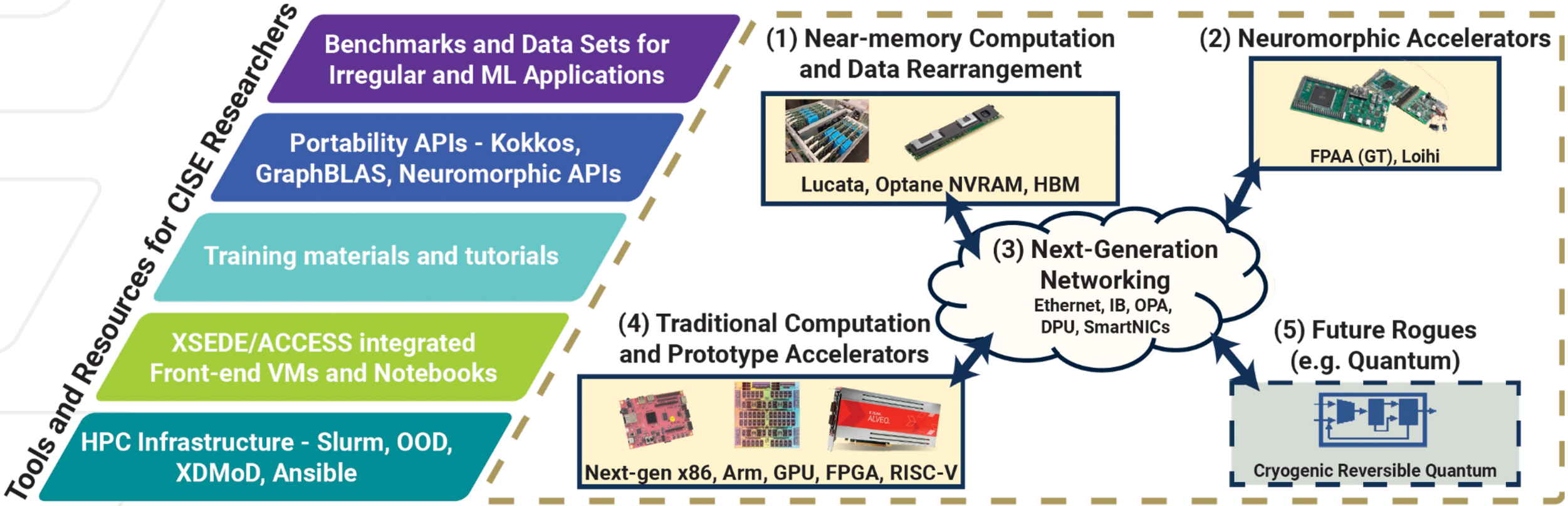
The CRNCH Rogues Gallery

Rogues Gallery is an NSF funded post-Moore testbed for CISE researchers and the community.

- The Gallery contains 40 servers and 40+ development boards – Intel CLX, SKL, ICX; AMD/NVIDIA GPUs; Arm; RISC-V; Xilinx
- ***Extreme heterogeneity*** with GPU, FPGAs, FPAAs, Optane Memory, InfiniBand, OmniPath, and Ethernet networking

Georgia Tech

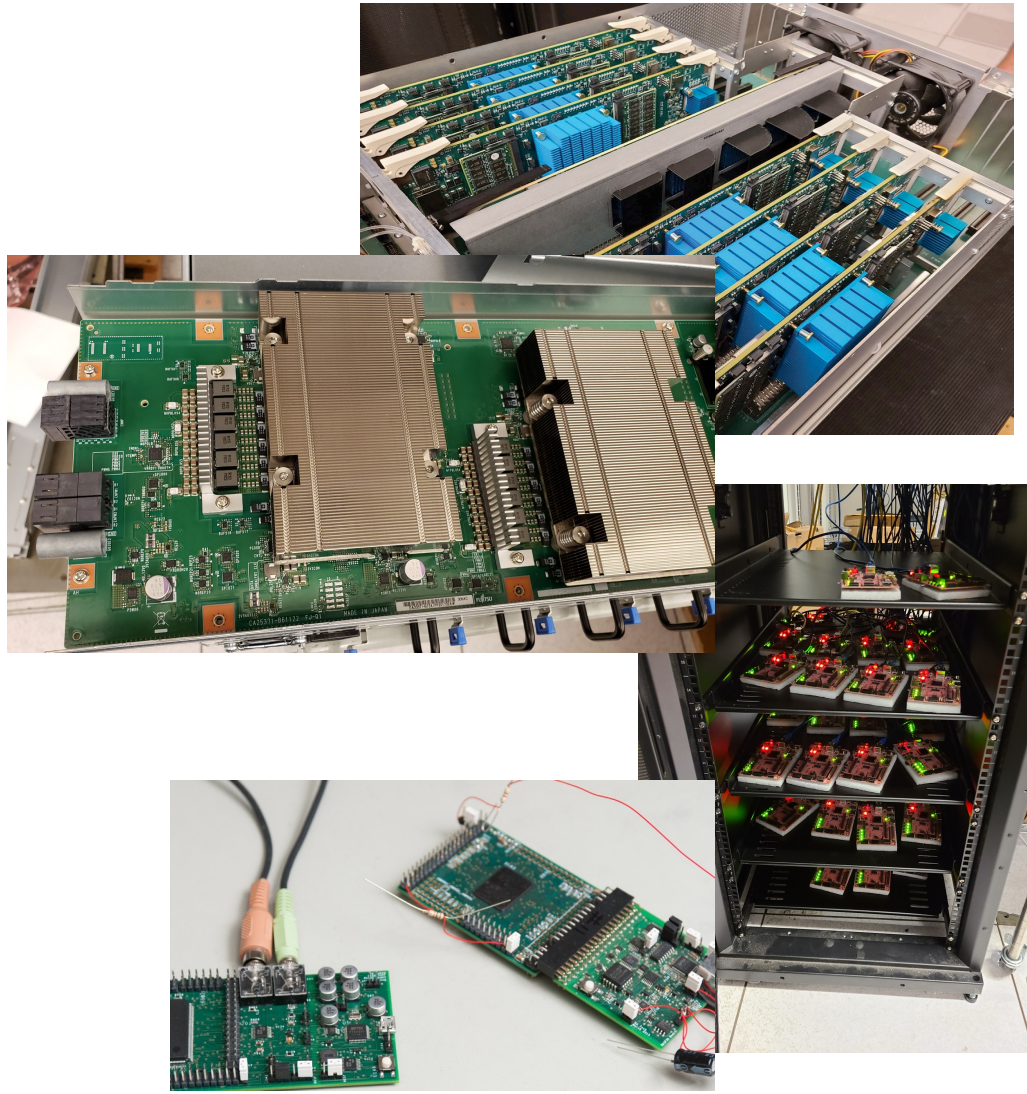
The Rogues Gallery



Georgia Tech

Rogues Gallery Highlights

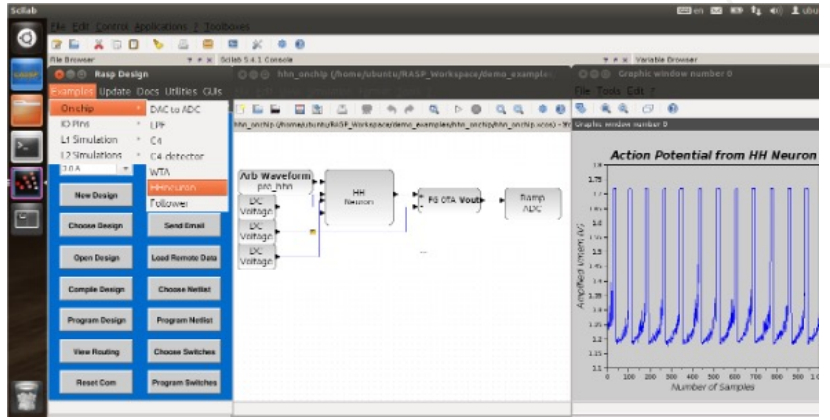
- First deployment of A64FX with Open OnDemand, later adopted by RIKEN for Fugaku
- Largest public instance of Lucata Pathfinder - #211 on 2021 Graph500 and #46 on GreenGraph500 rankings
- Support for over 180 researchers and 60-70 external users across multiple areas
- Support for 80-130 students each year with Pynq boards and similar infrastructure



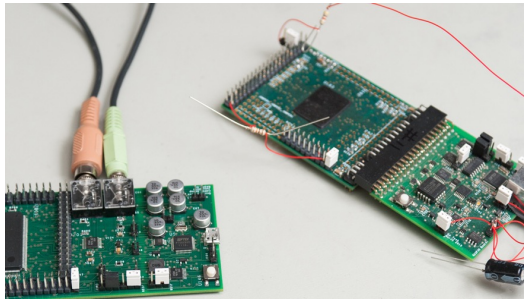
Georgia Tech

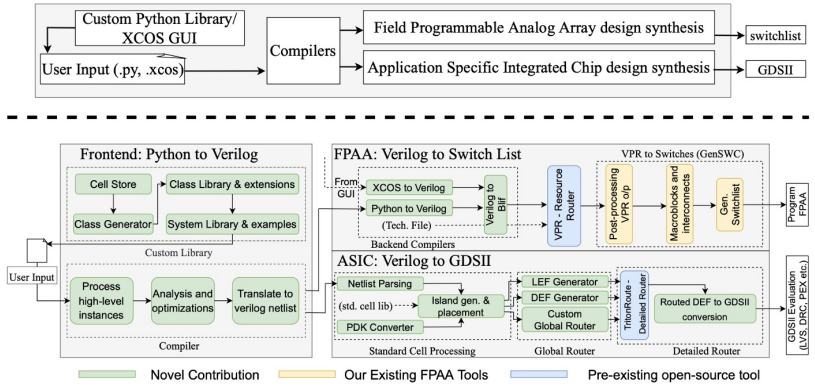
Unique Rogues Gallery Capabilities

- Support for investigating graph analytics accelerators, neuromorphic hardware, and smart networking
- Integration with local educational mission
  - Provides valuable testing of novel architecture
  - Ex: FPAA Python workflow, PYNQ cluster
- Growing interactions with software development best practices
  - CI/CD, continuous benchmarking



Next generation tools for FPAA development: graphical with Xcos interface (above), full open-source analog FPAA workflow (below)





Georgia Tech

Why Use Rogues Gallery?

- Programmatic/Technology Pathfinding activities
- Access to heterogeneous resources for testing, validation, benchmarking
- Avoid administration of a prototype hardware testbed while maintaining regression testing confidence
- Stop-gap hardware access, decoupling supply chain timelines from software development
- Enables evaluation of open source libraries or frameworks on uncommon hardware

Rogues Gallery exists to enable new and unique research projects targeting novel hardware – come talk with us!

Georgia Tech

CI/CD with Modern Tools

Modern Flight Software requires an updated approach to using software engineering tools

- Software management
  - GitHub, GitLab, Jira
- Automated build and testing
  - GitHub Actions, Jenkins, etc.
  - Unit testing
  - Static analysis
  - Code coverage and formatting

Most freely available management solutions primarily support x86 CPU CI/CD

- *We can use the Rogues Gallery and Slurm to enable consistent testing for new architectures!*

Georgia Tech

Fig credit: Prosymbols from Flaticon.com

Slurm on the Rogues Gallery

The Slurm HPC scheduler allows for scheduling jobs across widely heterogeneous resources within the larger testbed. Slurm provides:

Resource management for scarce hardware resources

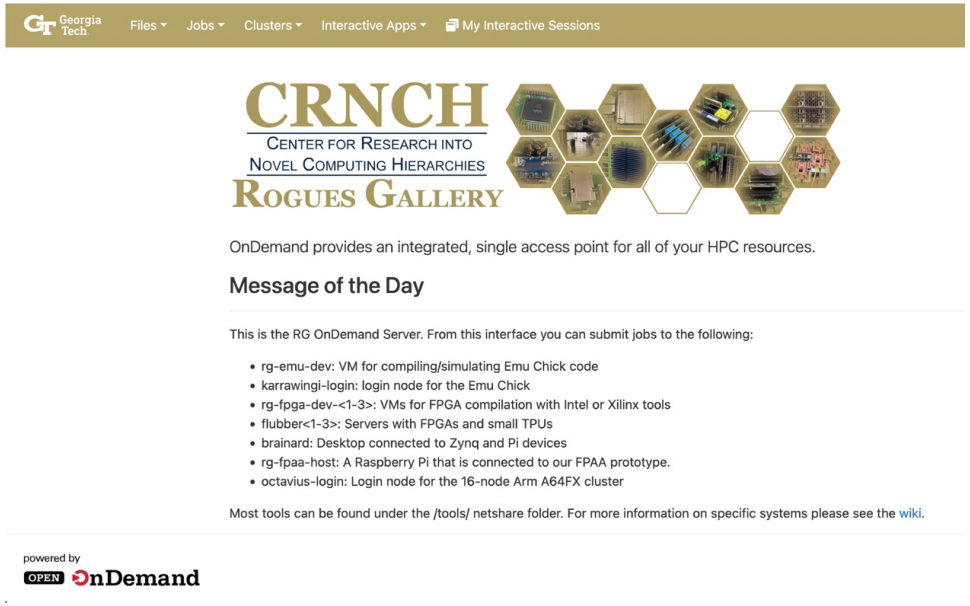
- Slurm allows for fairly sharing the latest GPUs

Partitioning and sharing different types of hardware within the same interface

- AMD CPUs coexist with FPGAs under the same Slurm cluster
- Anything that runs Linux (i.e., a Raspberry Pi or devboard) can also run a Slurm daemon!

Accounting and priority for specific resources

- Researchers with a near-term deadline can preempt other jobs
- Slurm accounting tracks all jobs and related statistics in a database for later analysis



Related tools like Open OnDemand allow for Slurm scheduling of Jupyter notebooks and other tools for testing software in detail

Leveraging Slurm for CI/CD

- Slurm jobs can use an interactive session or a “batch file” to run applications
- Additionally, you can specify features to target a particular architecture, e.g., *riscv* or *aarch64*
- Batchfiles and feature constraints can be combined to automate CI runners for different platforms

```
#Use sinfo to list the nodename (%10n) and its features (%40f)
gburdell:~$ sinfo -o "%10n %40f" | grep riscv
johnny-rv5 riscv,u740,ubuntu

#Check for Arm nodes that use the Ampere Q80-30 CPU
gburdell:~$ sinfo -o "%10n %40f" | grep ampereq8030
kingpin1   aarch64,ampereq8030,ubuntu,nvidia-gpu
kingpin2   aarch64,ampereq8030,ubuntu,nvidia-gpu

#Request an Ampere node for a job using feature constraints
salloc -t 01:00:00 -p rg-nextgen-hpc -Criscv
 ... Slurm job launch output elided  ...
gburdell9@johnny-rv5-1:~$
```

Selecting Arm or RISC-V features for jobs

```
#!/bin/bash
#SBATCH -Jspatter-ci-cuda                          # Job name
#SBATCH -N1 --cpus-per-task=4                      # Number of nodes and CPUs per node required
#SBATCH --mem-per-cpu=4G                           # Memory per core
#SBATCH -t 00:30:00                                # Duration of the job (Ex: 30 mins)
#SBATCH -p rg-nextgen-hpc                          # Partition Name
#SBATCH -o /ci-reports/spatter-cuda-test-%j.out    # Combined output and error messages file
#SBATCH --gres gpu:A100:1                          # Request a A100 GPU on any available node
#SBATCH -W                                         # Do not exit until the submitted job terminates.

##Add commands here to build and execute
cd $GITHUB_WORKSPACE

#Load NVHPC SDK, which includes the latest CUDA support
module load nvhpc
#Use cmake to build the CUDA version of the code
cmake -DBACKEND=cuda -DCOMPILER=nvcc -B build_cuda_workflow -S .
#Build the codebase and related tests
make -C build_cuda_workflow
cd build_cuda_workflow
#Run tests with CTest
make test
```

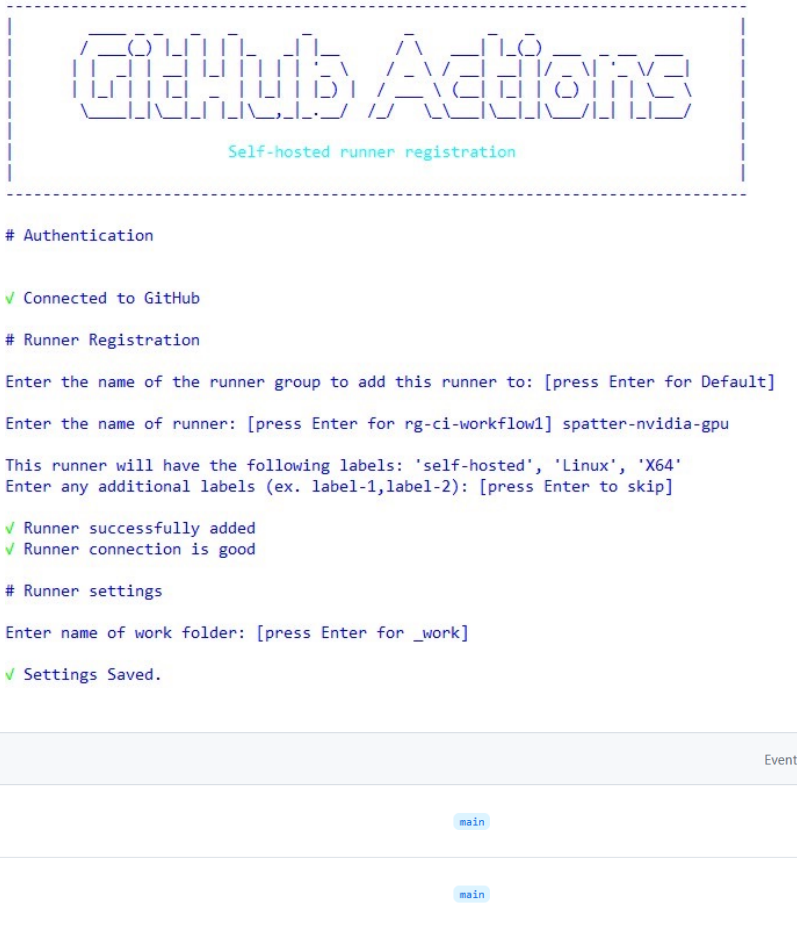
Example of a Slurm batchfile used for a CI runner ; note the GRES constraint requesting a GPU!

Georgia Tech

CI/CD Support on the Rogues Gallery

CRNCH RG provides Slurm-oriented infrastructure to schedule runners for novel architectures (RISC-V, Arm, Intel GPU, FPGA)

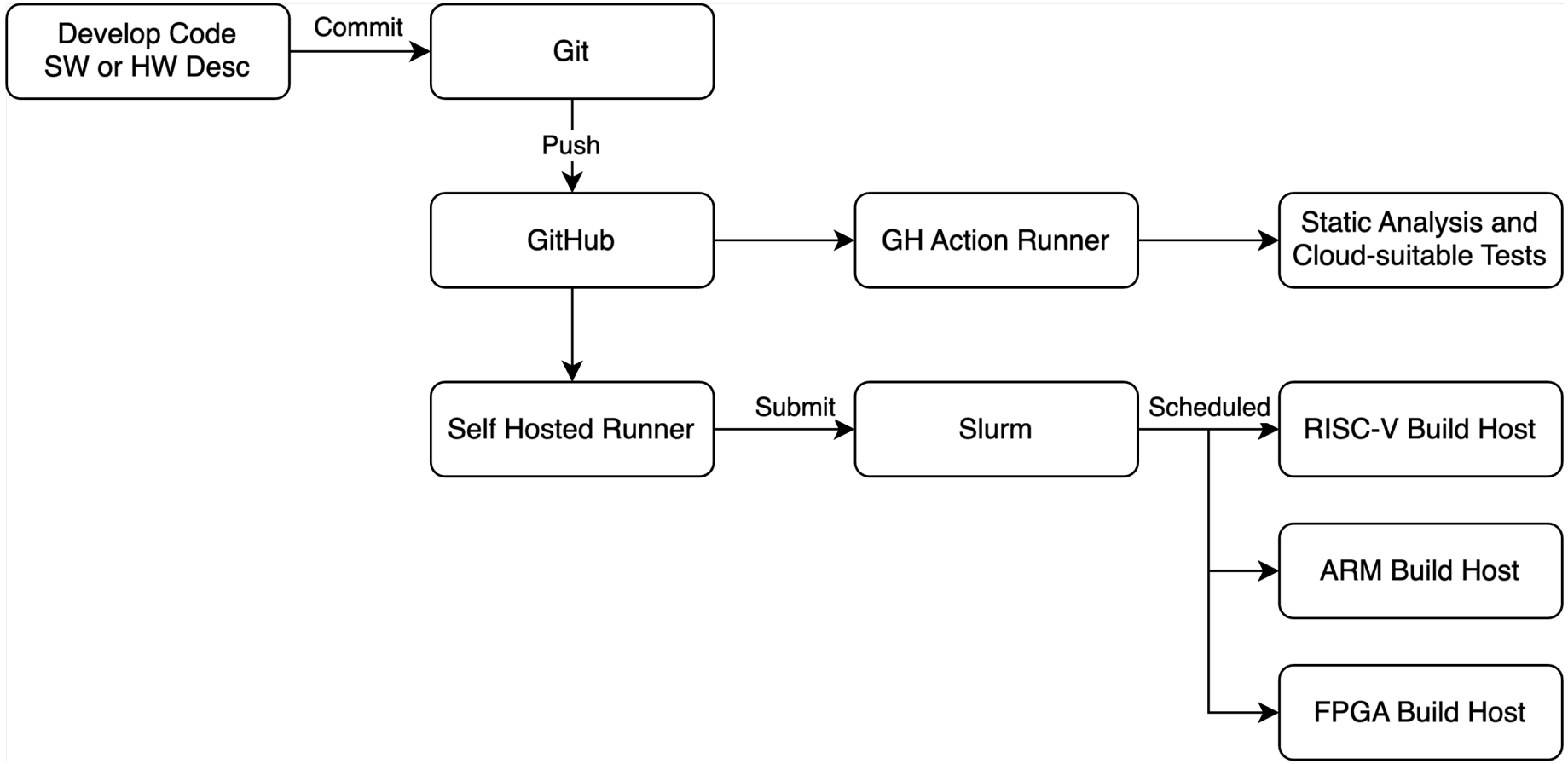
- Dedicated VM and secure setup process allows for persistent runners



Learn More: https://gt-crnch-rg.readthedocs.io/en/main/general/ci-runners.html

Georgia Tech

CI/CD support – Workflow



Georgia Tech

CI/CD Support – Self Hosted Runners with Slurm

Why we recommend self-hosted runners (with Slurm)

- A self-hosted CI runner is one that runs on local hardware rather than on Github cloud instances

Easier to support with RG testbed

- It's difficult for admins to test and validate actions hosted by external users

Self-hosted runners can be tested in phases

1. Run tests locally
2. Run on the Rogues Gallery with a Slurm job
3. Set up a Github action and download runner to RG
4. Test runner in your home directory
5. Install as a service, if desired

Georgia Tech

CI/CD Support – Installing a New Runner

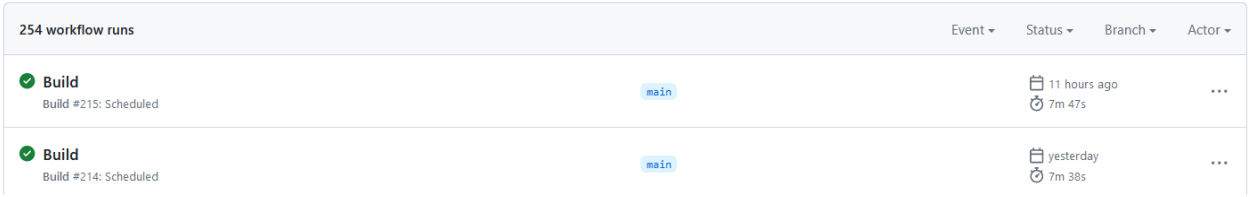
- Get GH Runner Token from Repo or GH Org
- Use rg-ci-workflow1 virtual machine within RG as the runner host
  - Run svc.sh from your user home directory
  - Note: Make sure that your runner “checks in” at least once a week
- Runner can be active in your home directory
  - Simple test cases: use nohup/tmux
  - For advanced usage, admins can help you install as a system service

Georgia Tech

Demo

- Hello World Project
- F Prime Deployment Build
- cFS Mission Project Build

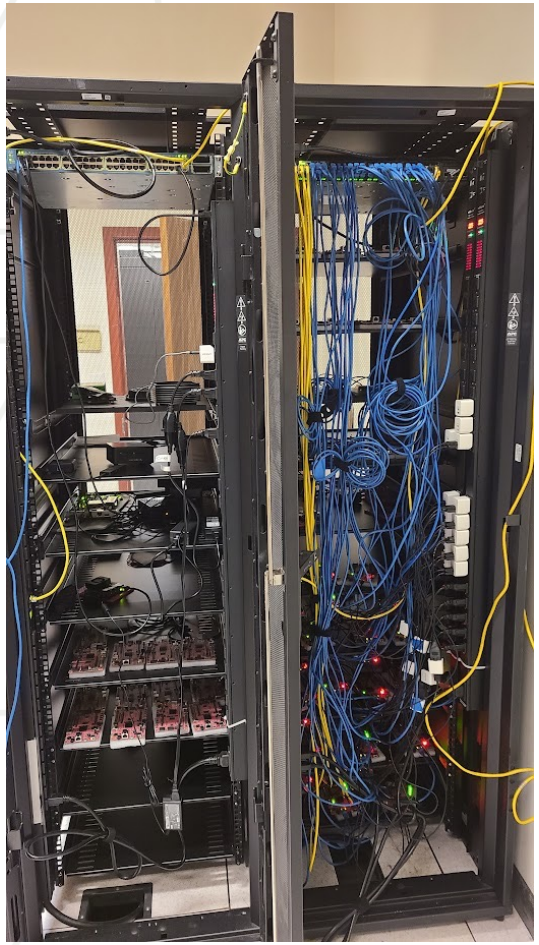

```
# Authentication

√ Connected to GitHub

# Runner Registration

Enter the name of the runner group to add this runner to: [press Enter for Default]

Enter the name of runner: [press Enter for rg-ci-workflow1] spatter-nvidia-gpu

This runner will have the following labels: 'self-hosted', 'Linux', 'X64'
Enter any additional labels (ex. label-1,label-2): [press Enter to skip]

√ Runner successfully added
√ Runner connection is good

# Runner settings

Enter name of work folder: [press Enter for _work]

√ Settings Saved.
```

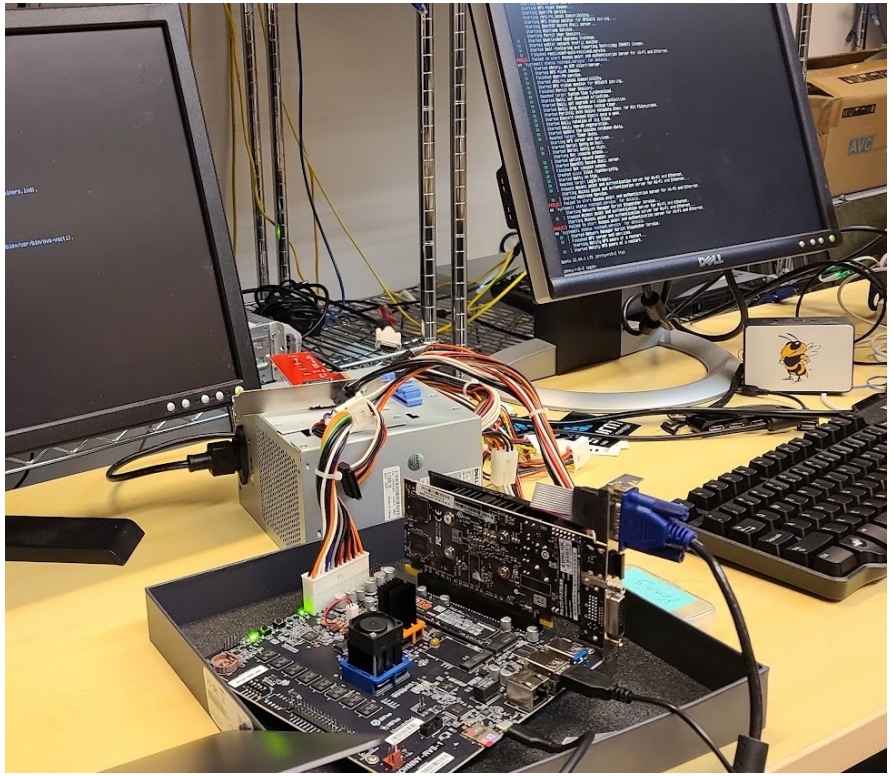


Learn More: https://gt-crnch-rg.readthedocs.io/en/main/general/ci-runners.html

Georgia Tech

Questions?





Project site: crnch-rg.cc.gatech.edu
GH Presence: https://github.com/gt-crnch-rg

Apply for access: https://crnch-rg.cc.gatech.edu/request-rogues-gallery-access/

Georgia Tech