A Unified Lab Notes Framework for Experimental Reproducibility in HPC Systems

R. Marshall and P. Bangalore

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.

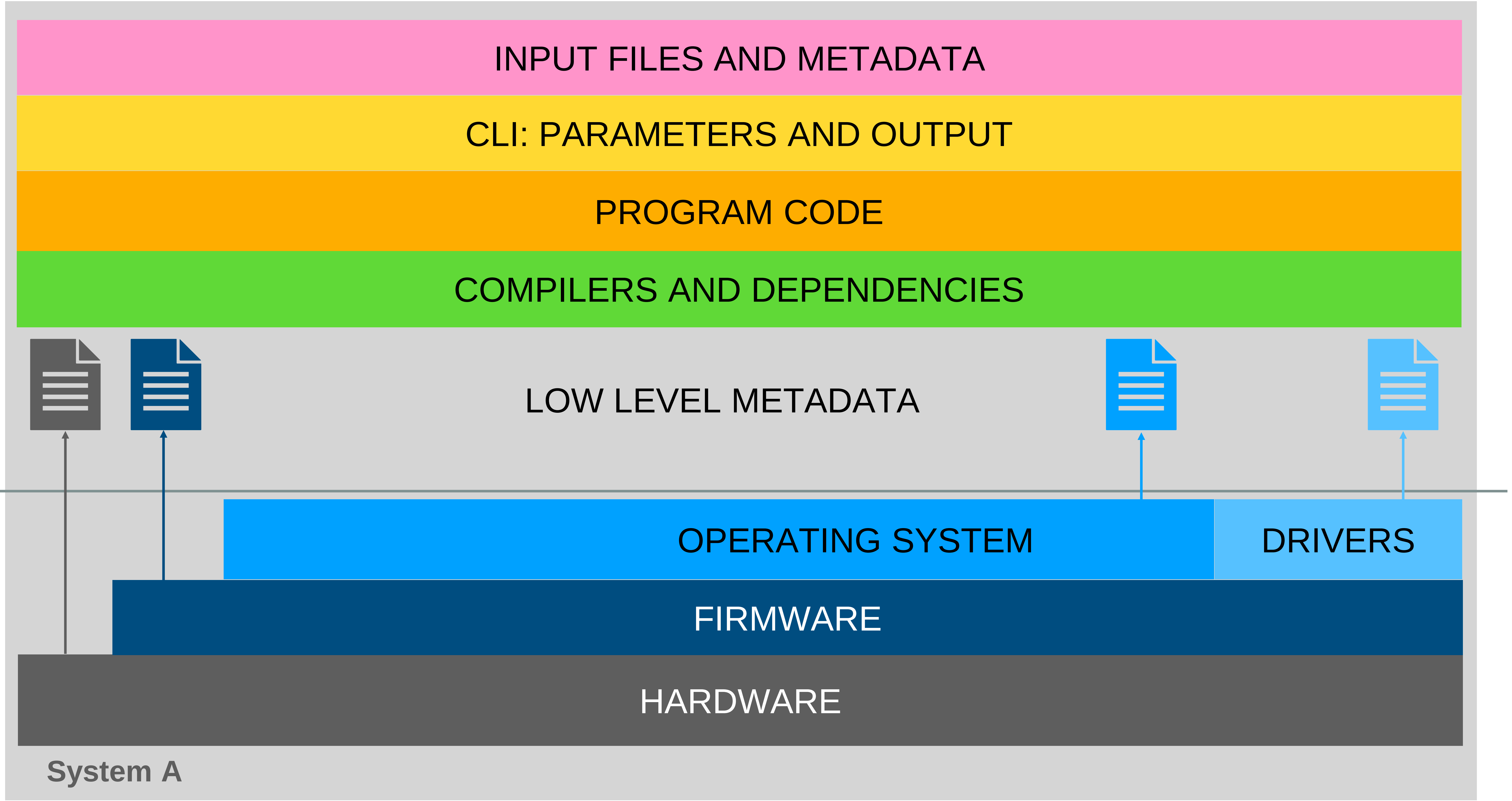
Motivation (cont.)

- 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

Trackable components

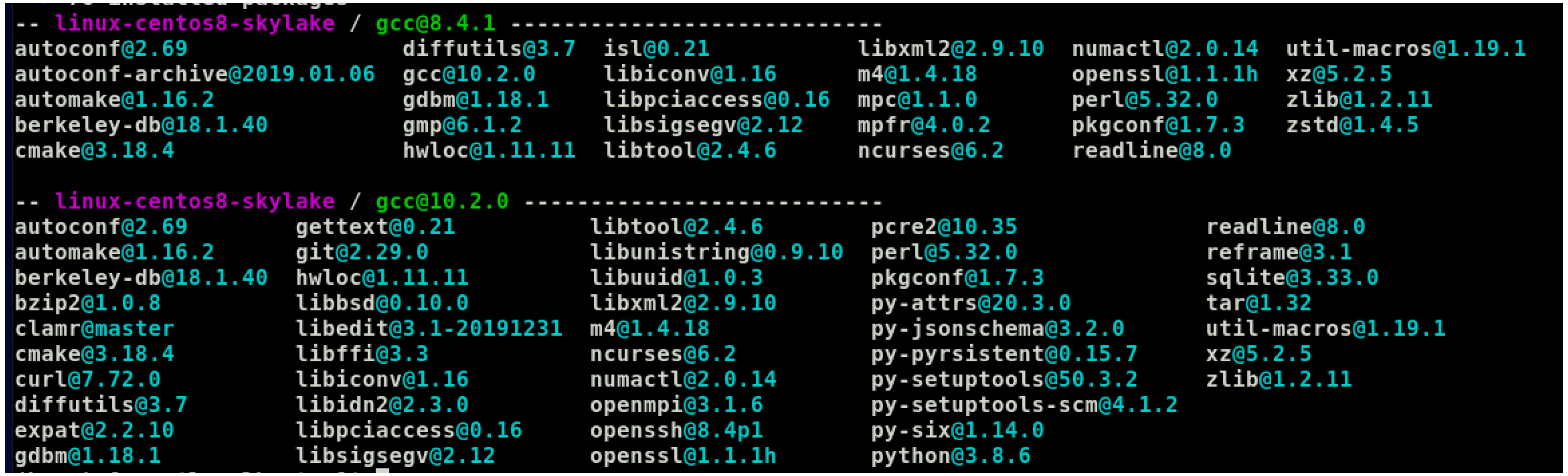
By breaking down an experiment into trackable components, we can ensure all aspects of a published experiment can be reproduced.

Components below this line are beyond scope to manage directly, though collection of metadata to record is OK.



Experimental Integrity: Compilers and Dependencies

- For full reproducibility of an experimental application that uses binary executables, the compiler and linked libraries should also be reproduced.
- Systems like Spack are useful for tracking and managing dependencies.



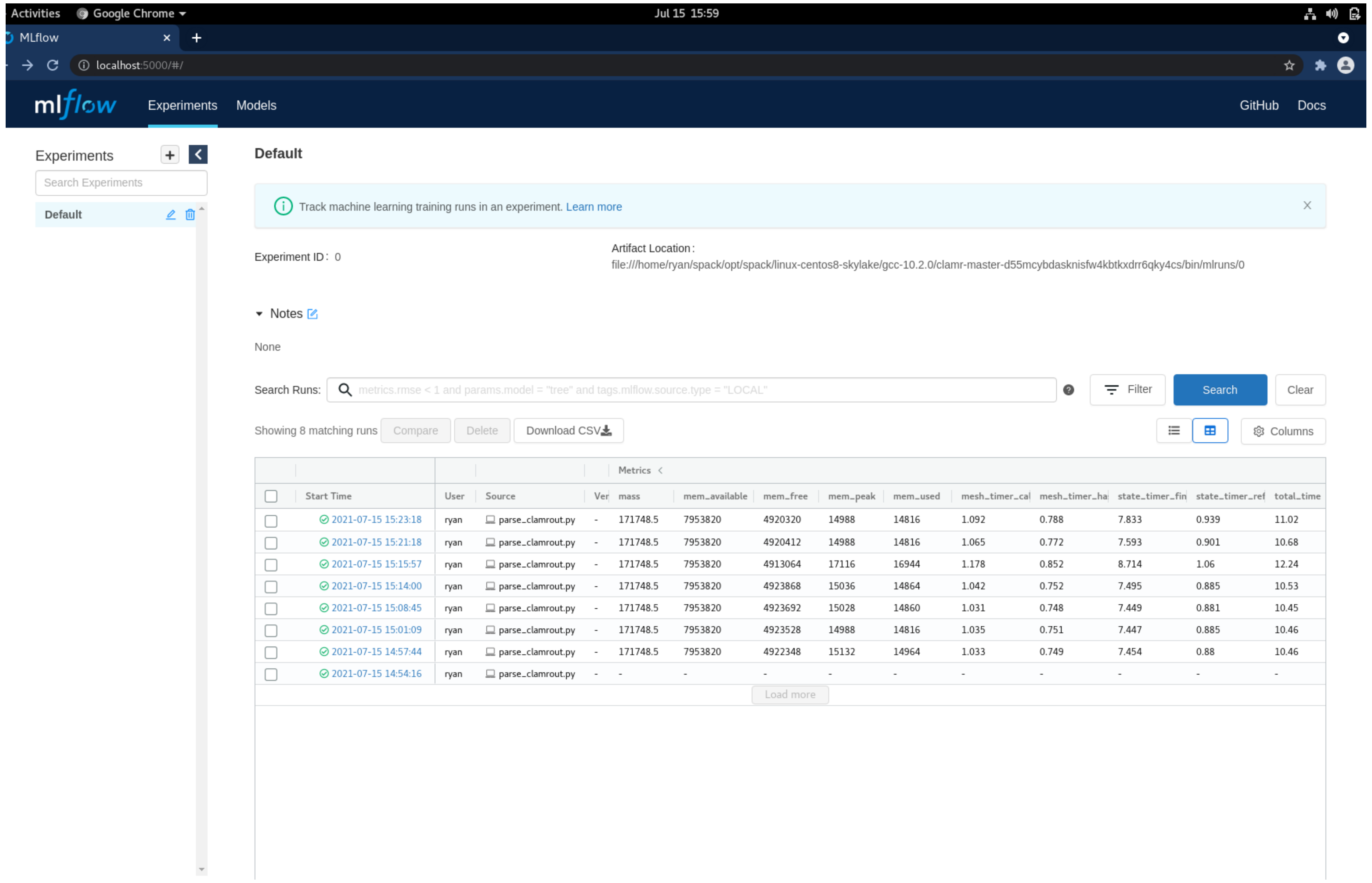
```
-- linux-centos8-skylake / gcc@8.4.1 --------------------------
autoconf@2.69               diffutils@3.7 isl@0.21          libxml2@2.9.10 numactl@2.0.14 util-macros@1.19.1
autoconf-archive@2019.01.06 gcc@10.2.0    libiconv@1.16     m4@1.4.18      openssl@1.1.1h xz@5.2.5
automake@1.16.2             gdbm@1.18.1   libpciaccess@0.16 mpc@1.1.0      perl@5.32.0    zlib@1.2.11
berkeley-db@18.1.40         gmp@6.1.2     libsigsegv@2.12   mpfr@4.0.2     pkgconf@1.7.3  zstd@1.4.5
cmake@3.18.4                hwloc@1.11.11 libtool@2.4.6     ncurses@6.2    readline@8.0

-- linux-centos8-skylake / gcc@10.2.0 -------------------------
autoconf@2.69       gettext@0.21         libtool@2.4.6       pcre2@10.35             readline@8.0
automake@1.16.2     git@2.29.0           libunistring@0.9.10 perl@5.32.0             reframe@3.1
berkeley-db@18.1.40 hwloc@1.11.11        libuuid@1.0.3       pkgconf@1.7.3           sqlite@3.33.0
bzip2@1.0.8         libbsd@0.10.0        libxml2@2.9.10      py-attrs@20.3.0         tar@1.32
clamr@master        libedit@3.1-20191231 m4@1.4.18           py-jsonschema@3.2.0     util-macros@1.19.1
cmake@3.18.4        libffi@3.3           ncurses@6.2         py-pyrsistent@0.15.7    xz@5.2.5
curl@7.72.0         libiconv@1.16        numactl@2.0.14      py-setuptools@50.3.2    zlib@1.2.11
diffutils@3.7       libidn2@2.3.0        openmpi@3.1.6       py-setuptools-scm@4.1.2
expat@2.2.10        libpciaccess@0.16    openssh@8.4p1       py-six@1.14.0
gdbm@1.18.1         libsigsegv@2.12      openssl@1.1.1h      python@3.8.6
```

Output of ‘spack find’

Experimental Integrity: User Input/Output

- The following should be tracked and stored:
  - user configuration files
  - program input files/metadata
  - command line parameters
  - program output
- Storing in a common format helps ensure experimental integrity
- Tools/systems of interest:
  - MLFlow
  - ReFrame



Default

Track machine learning training runs in an experiment. Learn more

Experiment ID: 0

Artifact Location: file:///home/ryan/spack/opt/spack/linux-centos8-skylake/gcc-10.2.0/clamr-master-d55mcybdasknisfw4kbtkxdrr6qky4cs/bin/mlruns/0

Notes

None

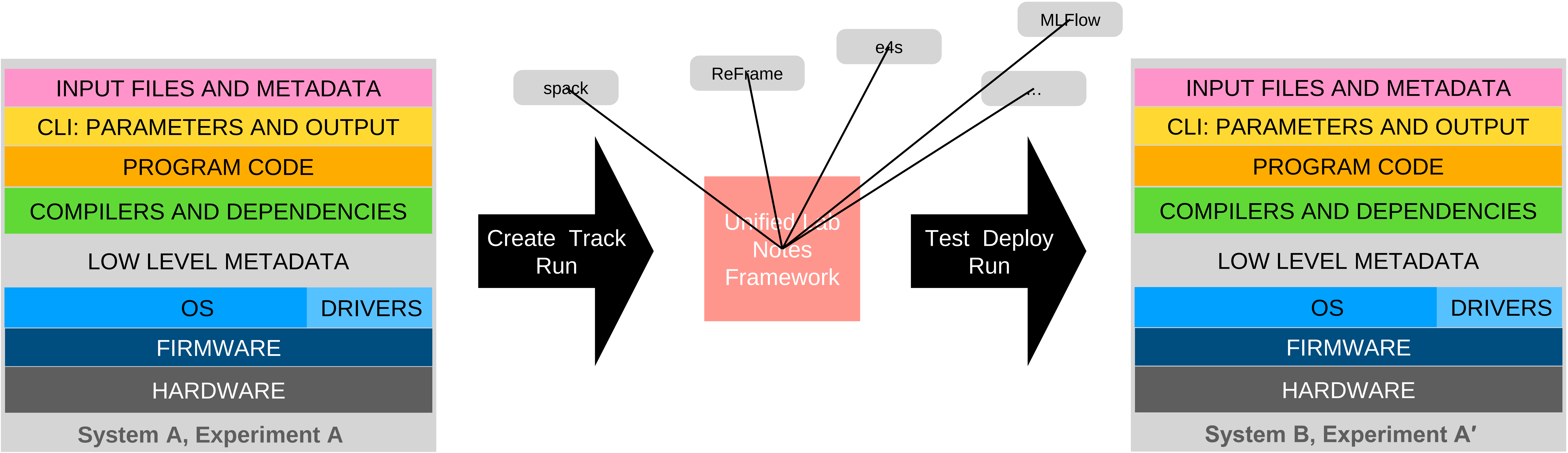
Search Runs: metrics.rmse < 1 and params.model = "tree" and tags.mlflow.source.type = "LOCAL"

Showing 8 matching runs

					Metrics									
□	Start Time	User	Source	Ver	mass	mem_available	mem_free	mem_peak	mem_used	mesh_timer_cal	mesh_timer_ha	state_timer_fin	state_timer_ref	total_time
□	2021-07-15 15:23:18	ryan	parse_clamrout.py	-	171748.5	7953820	4920320	14988	14816	1.092	0.788	7.833	0.939	11.02
□	2021-07-15 15:21:18	ryan	parse_clamrout.py	-	171748.5	7953820	4920412	14988	14816	1.065	0.772	7.593	0.901	10.68
□	2021-07-15 15:15:57	ryan	parse_clamrout.py	-	171748.5	7953820	4913064	17116	16944	1.178	0.852	8.714	1.06	12.24
□	2021-07-15 15:14:00	ryan	parse_clamrout.py	-	171748.5	7953820	4923868	15036	14864	1.042	0.752	7.495	0.885	10.53
□	2021-07-15 15:08:45	ryan	parse_clamrout.py	-	171748.5	7953820	4923692	15028	14860	1.031	0.748	7.449	0.881	10.45
□	2021-07-15 15:01:09	ryan	parse_clamrout.py	-	171748.5	7953820	4923528	14988	14816	1.035	0.751	7.447	0.885	10.46
□	2021-07-15 14:57:44	ryan	parse_clamrout.py	-	171748.5	7953820	4922348	15132	14964	1.033	0.749	7.454	0.88	10.46
□	2021-07-15 14:54:16	ryan	parse_clamrout.py	-	-	-	-	-	-	-	-	-	-	-

Output from CLAMR runs in MLFlow

A Unified Lab Notes Framework



INPUT FILES AND METADATA
CLI: PARAMETERS AND OUTPUT
PROGRAM CODE
COMPILERS AND DEPENDENCIES
LOW LEVEL METADATA
OS DRIVERS
FIRMWARE
HARDWARE

System A, Experiment A

Create Track Run

spack
ReFrame
e4s
MLFlow
…

Unified Lab Notes Framework

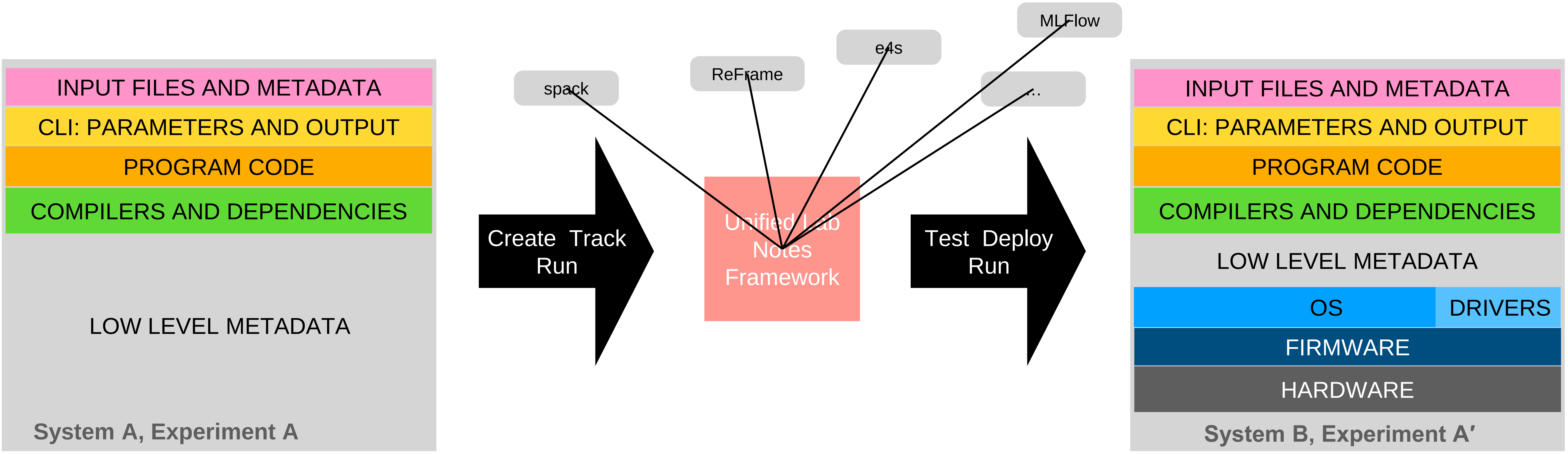
Test Deploy Run

INPUT FILES AND METADATA
CLI: PARAMETERS AND OUTPUT
PROGRAM CODE
COMPILERS AND DEPENDENCIES
LOW LEVEL METADATA
OS DRIVERS
FIRMWARE
HARDWARE

System B, Experiment A′

1. **Create Experiment A with framework**
2. **Run Experiment A on System A**
3. **Generate Experiment A′ with framework (based on Experiment A)**
4. **Deploy, test Experiment A′ on System B**
5. **Run Experiment A′ on System B**

A Unified Lab Notes Framework



INPUT FILES AND METADATA
CLI: PARAMETERS AND OUTPUT
PROGRAM CODE
COMPILERS AND DEPENDENCIES
LOW LEVEL METADATA
System A, Experiment A

Create Track Run

spack
ReFrame
e4s
MLFlow
...

Unified Lab Notes Framework

Test Deploy Run

INPUT FILES AND METADATA
CLI: PARAMETERS AND OUTPUT
PROGRAM CODE
COMPILERS AND DEPENDENCIES
LOW LEVEL METADATA
OS DRIVERS
FIRMWARE
HARDWARE
System B, Experiment A′

1. Create Experiment A with framework
2. Run Experiment A on System A
3. Generate Experiment A′ with framework (based on Experiment A)
4. Deploy, test Experiment A′ on System B
5. Run Experiment A′ on System B

Contributions

- Productivity:
  - new team members can easily get up to speed on existing experiments
  - simpler and more accurate handoff
- Reproducibility:
  - development of standards
  - as a regression test
- Ongoing work:
  - software product/toolkit for release
  - technical documentation (reproducibility standards)
  - conference publication (1 or more)

Thanks for attending

- Open for questions