FireSim

Running a FireSim Simulation: Password Cracking on a RISC-V SoC with SHA-3 Accelerators and Linux

https://fires.im

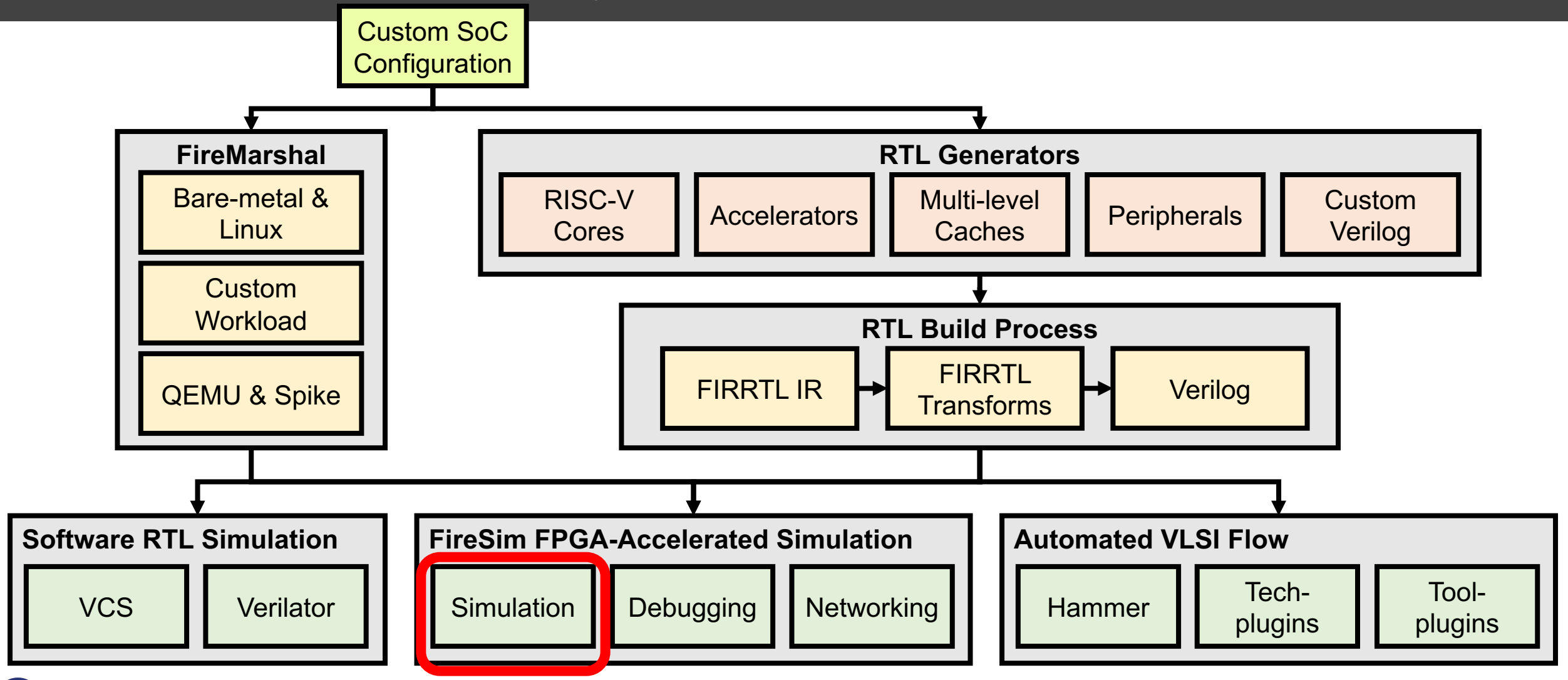
@firesimproject

ASPLOS Tutorial 2022

Abraham Gonzalez

Berkeley Architecture Research

Tutorial Roadmap



Agenda

- Configure and launch a simulation runfarm
- Boot Linux interactively on the target hardware
- Deploy new automated workloads
- Stress the SHA-3 accelerator with a complex Linux application (John the Ripper)

Prerequisites

- Interactive shell commands intended to be run during the tutorial are highlighted in blue blocks (prefixed by "`$`")
- Some simplifying assumptions about the shell environment:
  - We have set `$FDIR` and `$CDIR` variables refering to FireSim's and Chipyard's top directories.
  - We have already sourced the `sourceme-f1-manager.sh` setup script

```
$ export FDIR=~/chipyard-afternoon/sims/firesim
$ source $FDIR/sourceme-f1-manager.sh
```

Optional Prefetching

- We will later be using a special hardware config that instantiates the SHA3 accelerator
- To hide setup latency, start the RTL elaboration in another shell
  - This requires having integrated the `sha3` generator as instructed earlier
  - A prebuilt AGFI has been prepared, but this step will build the simulation drivers

```
$ cd $FDIR/sim
$ make DESIGN=FireSim \
    TARGET_CONFIG=DDR3FRFCFSLLC4MB_WithDefaultFireSimBridges_ ↩
       WithFireSimHighPerfConfigTweaks_chipyard.Sha3RocketConfig \
    PLATFORM_CONFIG=F30MHz_BaseF1Config \
    replace-rtl
```

- The `TARGET_CONFIG` variable is all one word

Prefetching

- We will later be launching and setting up simulations
- To hide setup latency, lets do the following:
  - Edit `$FDIR/deploy/config_runtime.ini` to match the following

```
[runfarm]
f1_16xlarges=0
f1_2xlarges=1

[targetconfig]
topology=no_net_config
no_net_num_nodes=1

defaulthwconfig=firesim-rocket-singlecore-no-nic-l2-lbp
```

Prefetching

- We will later be launch and setting up simulations
- To hide setup latency, lets do the following:
  - Add this hardware entry to `config_hwdb.ini`:

Note: Make sure there are no duplicate entries

```
$ cd $FDIR/deploy
$ cat built-hwdb-entries/firesim-rocket-singlecore-no-nic-l2-lbp >> ↩
  config_hwdb.ini
```

- Verify that it follows this format (with a unique AGFI ID):

```
[firesim-rocket-singlecore-no-nic-l2-lbp]
agfi=agfi-05d8ed7cc486ace80
deploytripletoverride=None
customruntimeconfig=None
```

Note: “l2” and not “12”

A pre-populated entry is provided for you to use! Otherwise, you would have to run `firesim buildafi`

Prefetching

- We will later be launching and setting up simulations
- To hide setup latency, lets do the following:
  - Run the following commands

```
$ tmux new -s sim-area
$ firesim launchrunfarm && firesim infrasetup
```

- Verify that you aren't in another `tmux` session.
- Can exit out of prior `tmux` session using `CTRL+b` then `d`

- Once complete exit out of `tmux` session using `CTRL+b` then `d`

What did we just do?

Runtime Configuration

What to simulate and *what infrastructure is required* is controlled by `$FDIR/deploy/config_runtime.ini`

- Target-level: Assemble a simulated system from components
  - FPGA images of SoC hardware designs
  - Network topology
  - Workload definition
- Host-level: Specify which EC2 instances to use

config_runtime.ini

The `[runfarm]` section specifies the number, type, and other launch parameters of instances to be managed

```
[runfarm]
runfarmtag=mainrunfarm
always_expand_runfarm=yes

f1_16xlarges=1
m4_16xlarges=0
f1_4xlarges=0
f1_2xlarges=0

launch_instances_timeout_minutes=60

runinstancemarket=ondemand
spotinterruptionbehavior=terminate
spotmaxprice=ondemand
```

config_runtime.ini

The `[targetconfig]` section specifies the high-level configuration of the system to simulate

```
[targetconfig]
topology=example_8config
no_net_num_nodes=2
linklatency=6405
switchinglatency=10
netbandwidth=200
profileinterval=-1

defaulthwconfig=firesim-rocket-quadcore-nic-l2-llc4mb-ddr3
```

`defaulthwconfig` references an entry from `config_hwdb.ini`

config_runtime.ini

The `[workload]` section specifies the software to be executed on the simulated nodes

```
[workload]
workloadname=linux-uniform.json
terminateoncompletion=no
suffixtag=
```

Other sections (`tracing`, `autocounter`, `hostdebug`, `synthprint`) will be explained further during the debugging session.

Testing the new AGFI

- By now, the `buildafi` run that you started at the very beginning of this tutorial should have finished!
- Add your hardware entry to `config_hwdb.ini`:

Note: Make sure to remove the duplicate entry

```
$ cd $FDIR/deploy
$ cat built-hwdb-entries/firesim-rocket-singlecore-no-nic-l2-lbp >> ↩
  config_hwdb.ini
```

- Verify that it follows this format (with a unique AGFI ID):

```
[firesim-rocket-singlecore-no-nic-l2-lbp]
agfi=agfi-05d8ed7cc486ace80
deploytripletoverride=None
customruntimeconfig=None
```

Note: “l2” and not “12”

Didn’t work? A pre-populated entry is provided for you to use! Otherwise, you would have to run `firesim buildafi`

Single-Node Simulation

What did we modify in `config_runtime.ini` earlier:

```
[runfarm]
f1_16xlarges=0
f1_2xlarges=1

[targetconfig]
topology=no_net_config
no_net_num_nodes=1

defaulthwconfig=firesim-rocket-singlecore-no-nic-l2-lbp
```

- Use a smaller f1.2xlarge instance (1 FPGA)
- Simulate one non-networked node without a switch model
- Load the single-core Rocket design without a NIC

Launching Simulation Instances

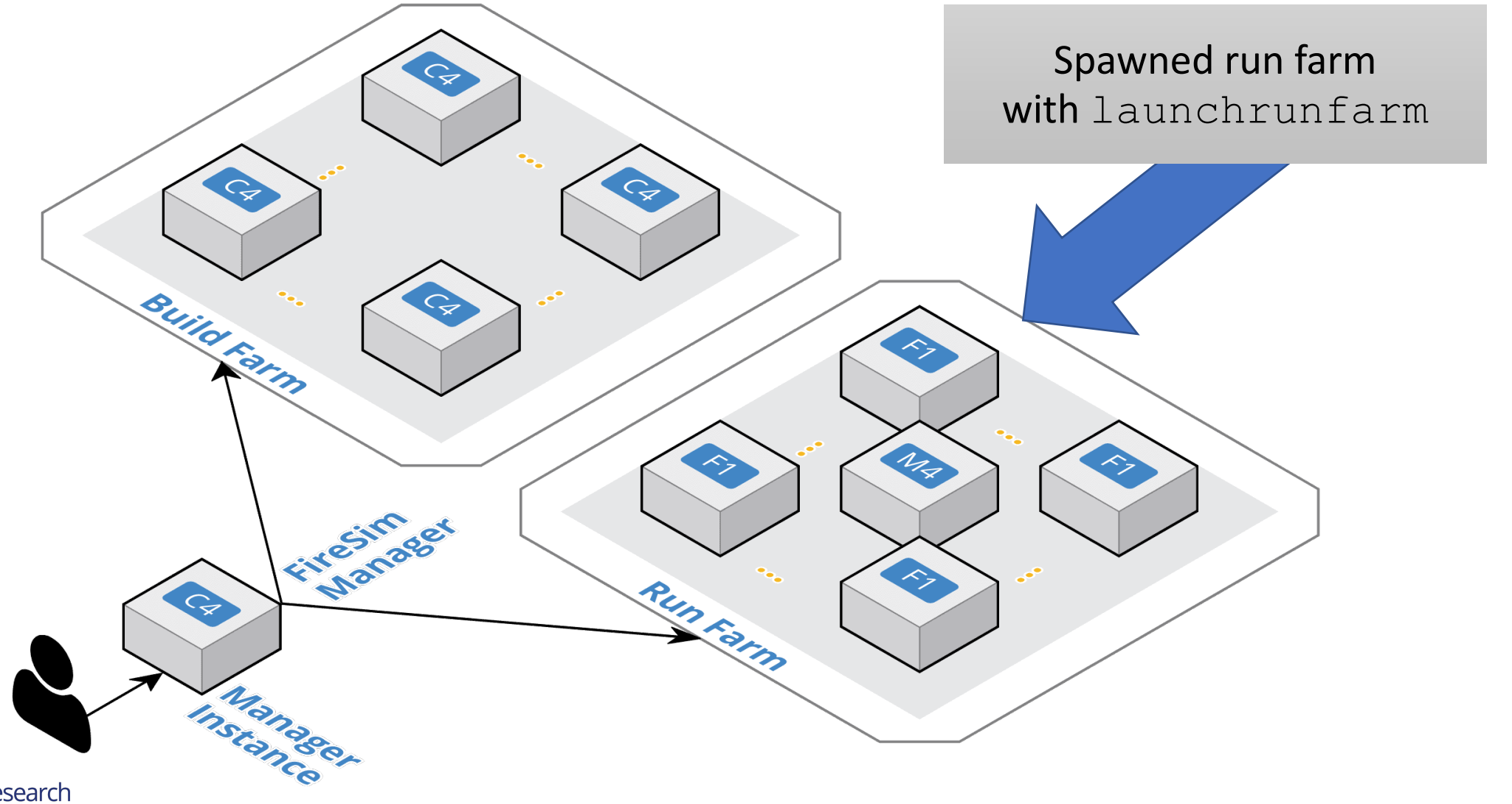
```
$ firesim launchrunfarm
```

```
FireSim Manager. Docs: http://docs.fires.im
Running: launchrunfarm

Waiting for instance boots: 0 f1.16xlarges
Waiting for instance boots: 0 f1.4xlarges
Waiting for instance boots: 0 m4.16xlarges
Waiting for instance boots: 1 f1.2xlarges
i-0a42dfd6edd081d10 booted!
The full log of this run is:
/home/centos/chipyard-afternoon/sims/firesim/deploy/logs/2019-10-07--05-14-31-
launchrunfarm-JNWXEVMP49H036E7.log
```

- Running in separate `tmux` session already!
- Go ahead and re-attach with
  - `tmux a -t sim-area`

Launching Simulation Instances



Deploying Simulation Infrastructure

```
$ firesim infrasetup
```

- Running already!

This deploys various software prerequisites:

- Builds host-side simulation drivers for the specific build triplet
- Builds the switch model executable (if enabled)
- Collects information about simulation instances and transfers files
- Programs the FPGAs with the desired AGFIs

Deploying Simulation Infrastructure

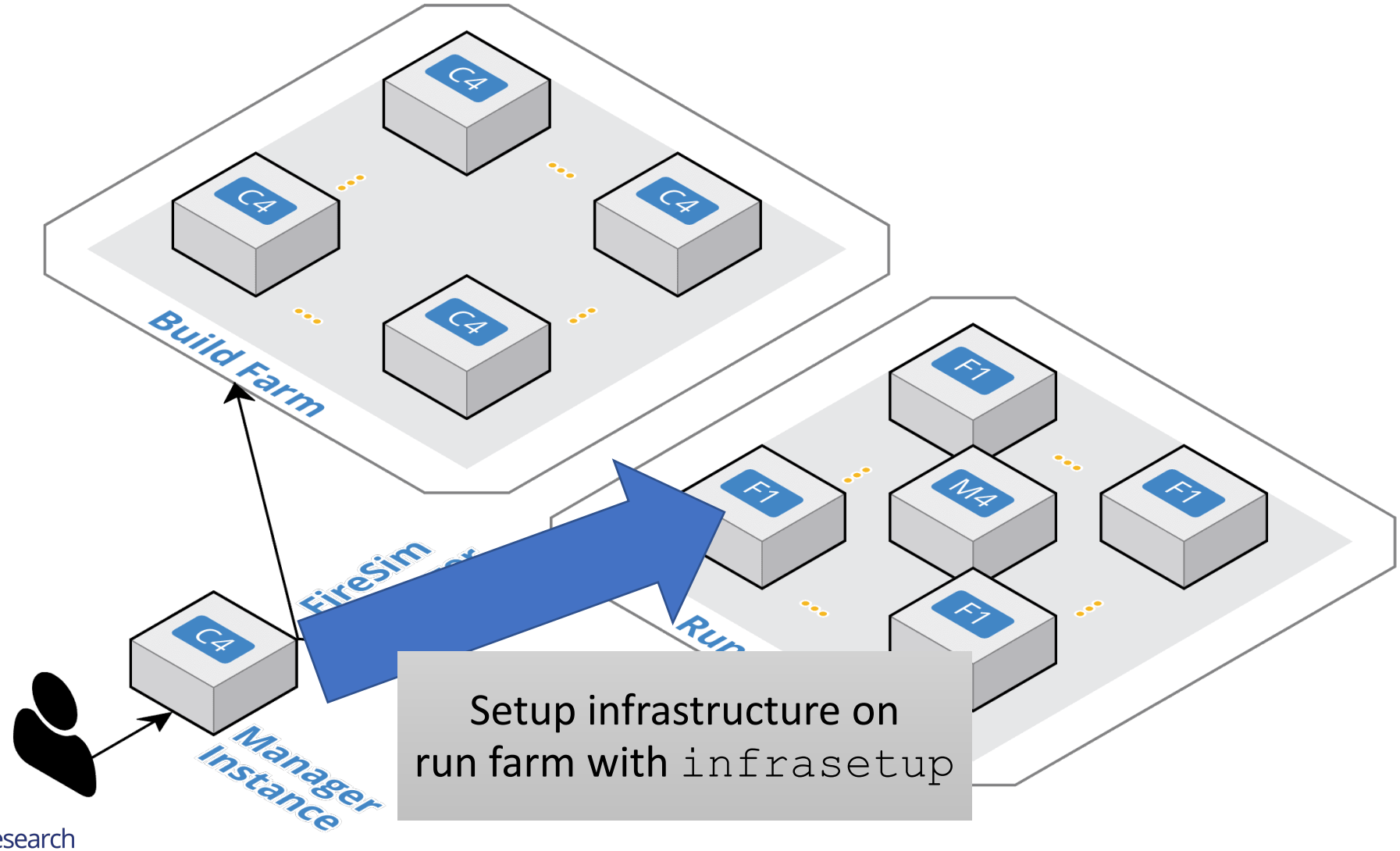
```
$ firesim infrasetup
```

- Running already!

```
FireSim Manager. Docs: http://docs.fires.im
Running: infrasetup

Building FPGA software driver for FireSim-
WithDefaultFireSimBridges_WithFireSimHighPerfConfigTweaks_chipyard.RocketConfig-F30MHz_BaseF1Config
[192.168.3.39] Executing task 'instance_liveness'
[192.168.3.39] Checking if host instance is up...
[192.168.3.39] Executing task 'infrasetup_node_wrapper'
[192.168.3.39] Copying FPGA simulation infrastructure for slot: 0.
...
[192.168.3.39] Clearing FPGA Slot 0.
[192.168.3.39] Checking for Cleared FPGA Slot 0.
[192.168.3.39] Flashing FPGA Slot: 0 with agfi: agfi-05d8ed7cc486ace80.
[192.168.3.39] Checking for Flashed FPGA Slot: 0 with agfi: agfi-05d8ed7cc486ace80.
[192.168.3.39] Unloading XDMA Driver Kernel Module.
[192.168.3.39] Loading XDMA Driver Kernel Module.
[192.168.3.39] Starting Vivado hw_server.
[192.168.3.39] Starting Vivado virtual JTAG.
The full log of this run is:
/home/centos/chipyard-afternoon/sims/firesim/deploy/logs/2022-02-27--00-11-08-infrasetup-HIYASQAGEEY5K5D4.log
```

Deploying Simulation Infrastructure



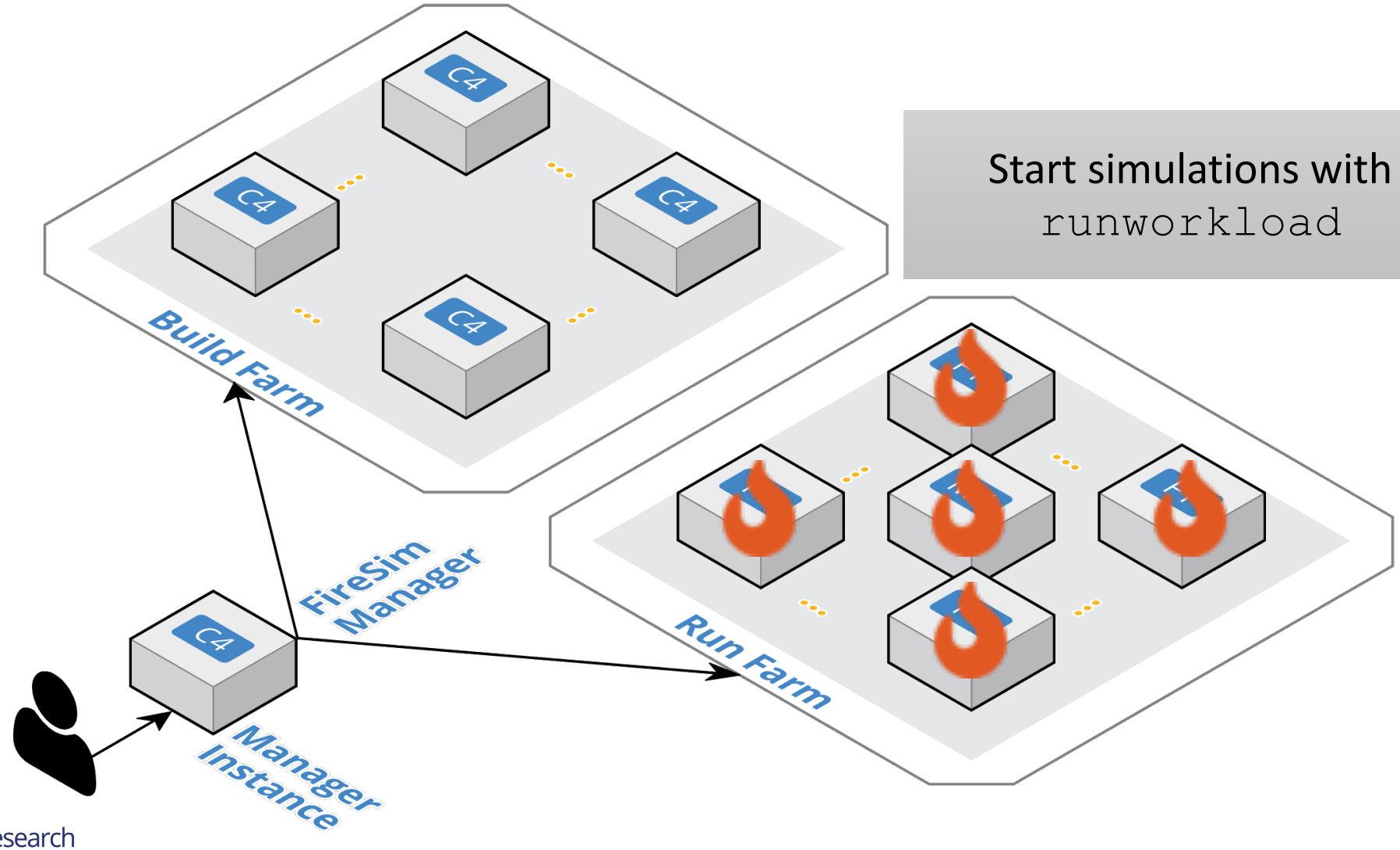
Running the Simulation

```
$ firesim runworkload
```

```
FireSim Manager. Docs: http://docs.fires.im
Running: runworkload

Creating the directory: /home/centos/chipyard-afternoon/sims/firesim/deploy/results-
workload/2019-10-07--05-35-00-linux-uniform/
[192.168.3.142] Executing task 'instance_liveness'
[192.168.3.142] Checking if host instance is up...
[192.168.3.142] Executing task 'boot_switch_wrapper'
[192.168.3.142] Executing task 'boot_simulation_wrapper'
[192.168.3.142] Starting FPGA simulation for slot: 0.
[192.168.3.142] Executing task 'monitor_jobs_wrapper'
```

Running the Simulation



Monitoring the Simulation

You should see a live status report that refreshes periodically:

```
FireSim Simulation Status @ 2022-02-27 00:23:20.253671
--------------------------------------------------------------------------------
This workload's output is located in:
/home/centos/chipyard-afternoon/sims/firesim/deploy/results-workload/2022-02-27--00-22-53-linux-uniform/
This run's log is located in:
/home/centos/chipyard-afternoon/sims/firesim/deploy/logs/2022-02-27--00-22-53-runworkload-
S1D6GL76CDJXZ1PK.log
This status will update every 10s.
--------------------------------------------------------------------------------
Instances
--------------------------------------------------------------------------------
Instance IP:   192.168.3.39 | Terminated: False
--------------------------------------------------------------------------------
Simulated Switches
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Simulated Nodes/Jobs
--------------------------------------------------------------------------------
Instance IP:   192.168.3.39 | Job: linux-uniform0 | Sim running: True
--------------------------------------------------------------------------------
Summary
--------------------------------------------------------------------------------
1/1 instances are still running.
1/1 simulations are still running.
--------------------------------------------------------------------------------
```

Interacting with the Simulation

Look for the instance's IP address in the status:

```
FireSim Simulation Status @ 2022-02-27 00:23:20.253671
--------------------------------------------------------------------------------
This workload's output is located in:
/home/centos/chipyard-afternoon/sims/firesim/deploy/results-workload/2022-02-27--00-22-53-linux-uniform/
This run's log is located in:
/home/centos/chipyard-afternoon/sims/firesim/deploy/logs/2022-02-27--00-22-53-runworkload-
S1D6GL76CDJXZ1PK.log
This status will update every 10s.
--------------------------------------------------------------------------------
Instances
--------------------------------------------------------------------------------
Instance IP:   192.168.3.39 | Terminated: False
--------------------------------------------------------------------------------
Simulated Switches
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Simulated Nodes/Jobs
--------------------------------------------------------------------------------
Instance IP:   192.168.3.39 | Job: linux-uniform0 | Sim running: True
--------------------------------------------------------------------------------
Summary
--------------------------------------------------------------------------------
1/1 instances are still running.
1/1 simulations are still running.
--------------------------------------------------------------------------------
```

Interacting with the Simulation

- On the *manager* instance, `ssh` into the run farm instance:

```
$ ssh 192.168.3.39
```

Detach from `tmux` or open a new `ssh` window

```
 ___ ___ ___   _     ___ _____  __    _   __ __ ___
| __| _ \/ __| /_\   |   \ __\ \ / /   /_\ |  \/  |_ _|
| _||  _/ (_ |/ _ \  | |) | _| \ V /   / _ \| |\/| || |
|_| |_|  \___/_/ \_\ |___/|___| \_/   /_/ \_\_|  |_|___|
AMI Version:         1.11.1
Xilinx Version:      2021.1
Readme:              /home/centos/src/README.md
AMI Release Notes:   /home/centos/src/RELEASE_NOTES.md
GUI/Cluster setup:   https://github.com/aws/aws-fpga/blob/master/developer_resources
```

- Then attach to the console of the simulated node:

```
$ screen -r fsim0
```

Logging Into the Simulated System

- Once Linux boots, the login prompt should appear over the console
- Log in as “root” with password “firesim” (password does not echo)

```
[    0.085714] EXT4-fs (iceblk): re-mounted. Opts: (null)
Starting syslogd: OK
Starting klogd: OK
Starting mdev... done.
Starting dropbear sshd: OK

Welcome to Buildroot
buildroot login: root
Password:
#
```

Logging Into the Simulated System

- Feel free to experiment with shell commands

```
# uname -a
# cat /proc/cpuinfo
# free -m
# vim
```

- When done, shut down the system and return to the manager node

```
# poweroff -f
```

Open a new `ssh` window or `CRTL+d` to exit

- This will also end the simulation

Custom FireSim Workloads

- *Workload*: Series of jobs (software configurations) assigned to run on individual simulations

- Two types of workloads:

  Uniform: Homogenous job run by all nodes in a simulated cluster

  Non-uniform: Each node is assigned a different job

  - Client/server configurations
  - Benchmark suites (SPEC17)

Workload Definitions

- Our previous example used “linux-uniform” as the simulated workload
- These JSON files live in `$FDIR/deploy/workloads/*.json`

```
{
    "benchmark_name" : "linux-uniform",
    "common_bootbinary" : "br-base-bin",
    "common_rootfs"     : "br-base.img",
    "common_outputs"    : ["/etc/os-release"],
    "common_simulation_outputs" : ["uartlog", "memory_stats.csv"]
}
```

- `$FDIR/deploy/workloads/linux-uniform/br-base{-bin,.img}` are symlinks to the FireMarshal-generated images

SPEC CPU2017

- 10 jobs – one per benchmark in the SPECrate Integer suite
- No time in this tutorial, but the general procedure is:
  - Build/install the SPEC17 target w/ FireMarshal in `$CDIR/software/spec2017`
  - Setup the `config_runtime.ini`
    - Set `f1_2xlarges=10`
    - Set `topology=no_net_config` and `no_net_num_nodes=10`
    - Set `workloadname=spec17-intrate.json`
    - Sselect the hardware config to benchmark, then `firesim launchrunfarm/infrasetup/runworkload!`

```
{
  "common_bootbinary" : "bbl-vmlinux",
  "benchmark_name" : "spec17-intrate",
  "deliver_dir" : "spec17-intrate",
  "common_args" : ["--copies 4"],
  "common_files" : ["intrate.sh"],
  "common_outputs" : ["/output"],
  "common_simulation_outputs" : ["uartlog"],
  "workloads" : [
    {
      "name": "500.perlbench_r",
      "files": ["500.perlbench_r"],
      "command": "cd /spec17-intrate && ./intrate.sh 500.perlbench_r",
      "simulation_outputs": [],
      "outputs": []
    },
    {
      "name": "502.gcc_r",
      "files": ["502.gcc_r"],
      "command": "cd /spec17-intrate && ./intrate.sh 502.gcc_r",
      "simulation_outputs": [],
      "outputs": []
    },
    …
  ]
}
```

John the Ripper

- Open-source password checking software
- Our customized version adds support for two more hash formats:
  - **Raw-SHA3-256**: pure software implementation using generic Keccak code
  - **Raw-SHA3-256-rocc**: RoCC accelerator offload

- github.com/ucb-bar/JohnTheRipper/blob/riscv/src/sha3_256_rocc_fmt_plug.c
  - The `crypt_all()` function performs the actual hashing
- Minor Linux kernel patches to facilitate accelerator context switching

Changing Workloads

- Generate the FireSim workload definition for “sha3-linux-jtr-test”:

```
$ cd $CDIR/generators/sha3/software
$ marshal install marshal-configs/sha3-linux-jtr-test.yaml
```

- Update `$FDIR/deploy/config_runtime.ini` accordingly:

```
defaulthwconfig=firesim-rocket-singlecore-sha3-no-nic-l2-llc4mb-ddr3
workloadname=sha3-linux-jtr-test.json
```

- Then start another simulation:

```
$ firesim infrasetup && firesim runworkload
```

Basic Benchmarking

- The workload first runs John the Ripper's low-level self-tests and benchmarks to measure raw hash performance
  - Passwords constitute a less optimal input for the accelerator
  - Many unrelated messages much shorter than the block size (1088 bits)
- "Crypts per second" (C/s) metric
  - *Real*: elapsed real time
  - *Virtual*: total CPU time

```
Benchmarking: Raw-SHA3-256 [SHA3 256 32/64]... DONE
Raw:    164928 c/s real, 164928 c/s virtual

Benchmarking: Raw-SHA3-256-rocc [SHA3 256 32/64]... DONE
Raw:    10171K c/s real, 10222K c/s virtual
```

Password Solving

- In the second half of the workload, the SHA-3 accelerator is used to attack sample hashes from the default wordlist
- `john` is given input files of one hash per line, unsalted for simplicity:

```
hash_0:be87f99a67e48ec4ec9f05b565f6ca531e24b9c71a62cfd3a58f54ebc60115ea
hash_1:f706280cdf972ed4af636d540e7d2ea2ff3e9f91e63bc389b2aa0fa288c486a9
hash_2:2cd81e6887b1618af765e2bc127f68b563e6a1b4abd397331b759f878eb8515e
hash_3:9cdc6b9ff3d0d0a90cb8670fb972debc08947697c6b63903458abbaaa0fe93c9
```

- The companion “sha3-linux-jtr-crack” workload includes a more challenging scenario that tests the incremental (brute-force) mode

Capturing Results

- Once the workload terminates automatically, the results are copied to the manager instance:

```
FireSim Simulation Exited Successfully. See results in:
/home/centos/chipyard/sims/firesim/deploy/results-workload/2019-10-07--08-13-35-sha3-linux-jtr-test/
```

- Console output recorded in `sha3-linux-jtr-test0/uartlog`
- HW configuration in `sha3-linux-jtr-test0/HW_CFG_SUMMARY`