FireMarshal:

Software Workload Management

Abraham Gonzalez

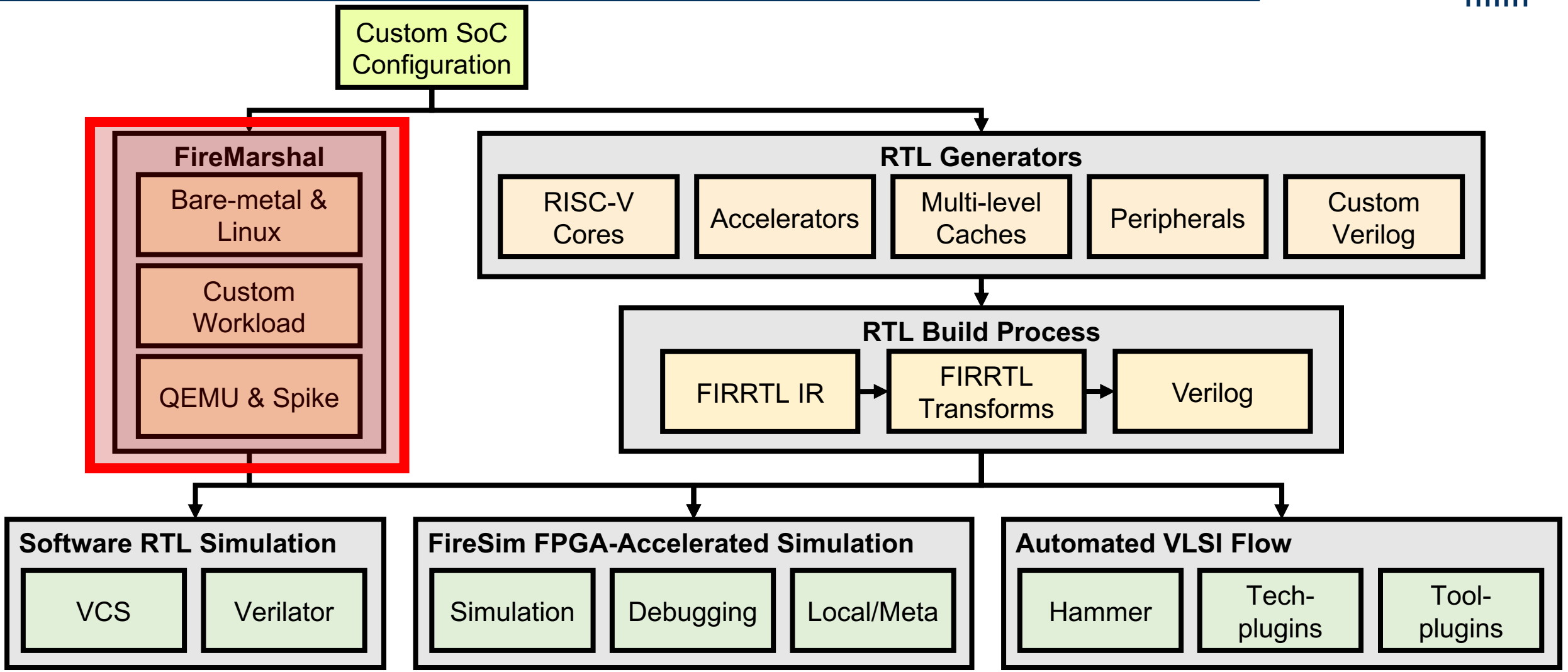
UC Berkeley

abe.gonzalez@eecs.berkeley.edu

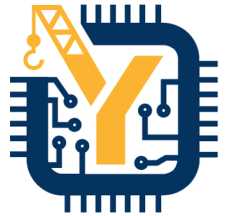
CHIPYARD

Berkeley Architecture Research

Tutorial Roadmap



In parallel…



Build Spike + SHA3 functional model

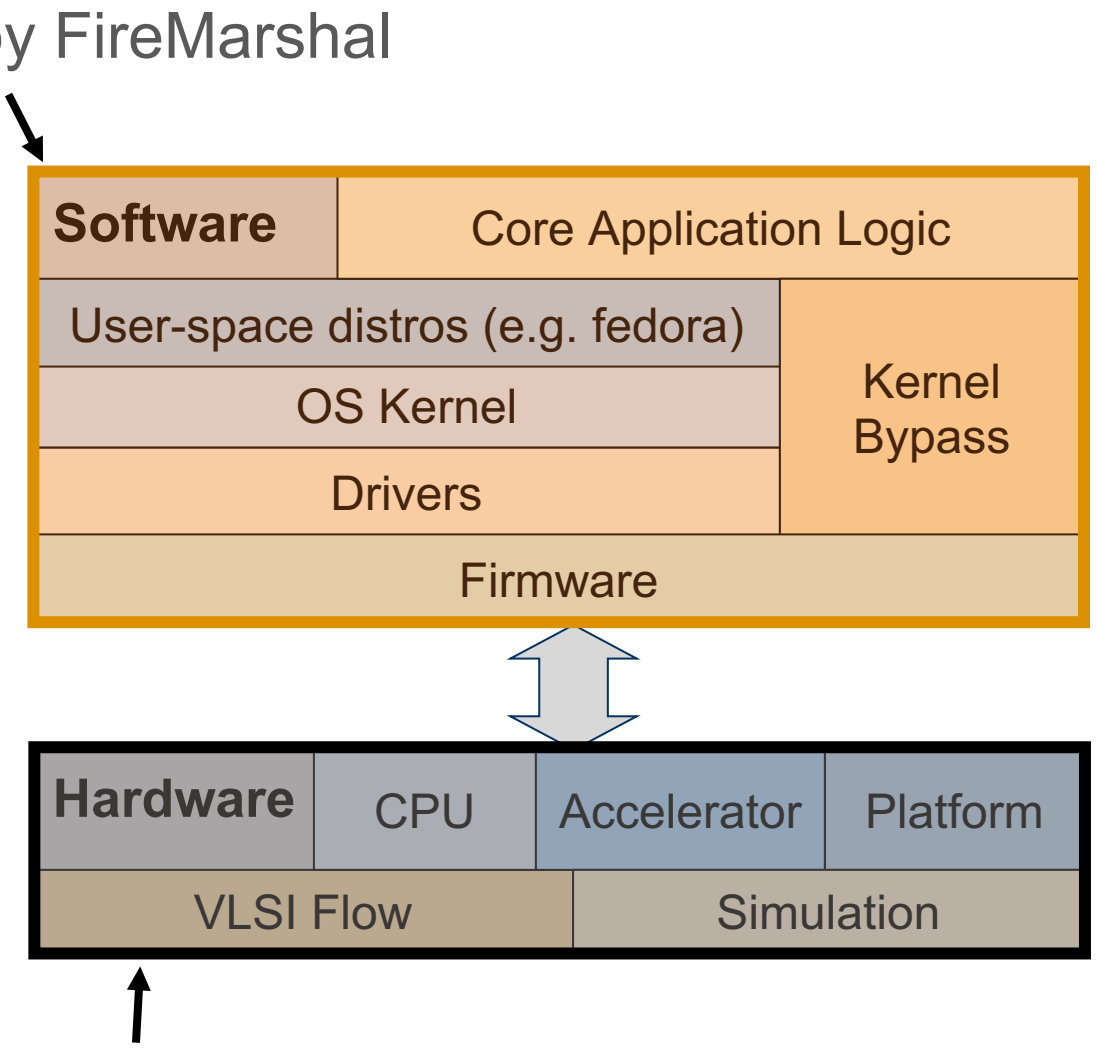
```
$ cd ~
$ wget tinyurl.com/cyfsim-esp-install
$ source cyfsim-esp-install


$ cd ~/chipyard-afternoon
$ cd generators/sha3/software
```

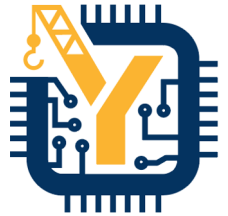
Software Workload Management

Workload Management Tasks:

- Building Binaries and Filesystems
- Experiment Management
  - Inputs and outputs
  - Repeatable execution
  - Multiple levels of simulation
- Reproducibility and Reusability
  - Share workloads with the community



Hardware/Software Co-Design Flow



Write Spec

super_cool_accelerator.md

Functional Model

$Spike_{sca}$

Write HW/SW

sca.chisel

sca.img/bin

Evaluate

FireSim

Requirements

Workload Management Tool Challenges

Write Spec

super_cool_accelerator.md

Functional Model

$Spike_{sca}$

Write HW/SW

sca.chisel

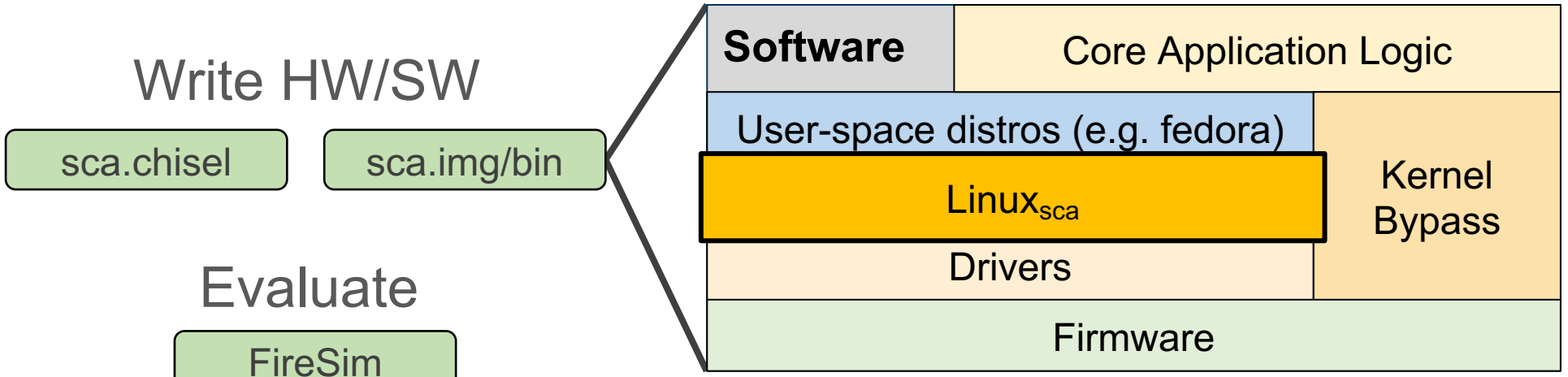
sca.img/bin

Evaluate

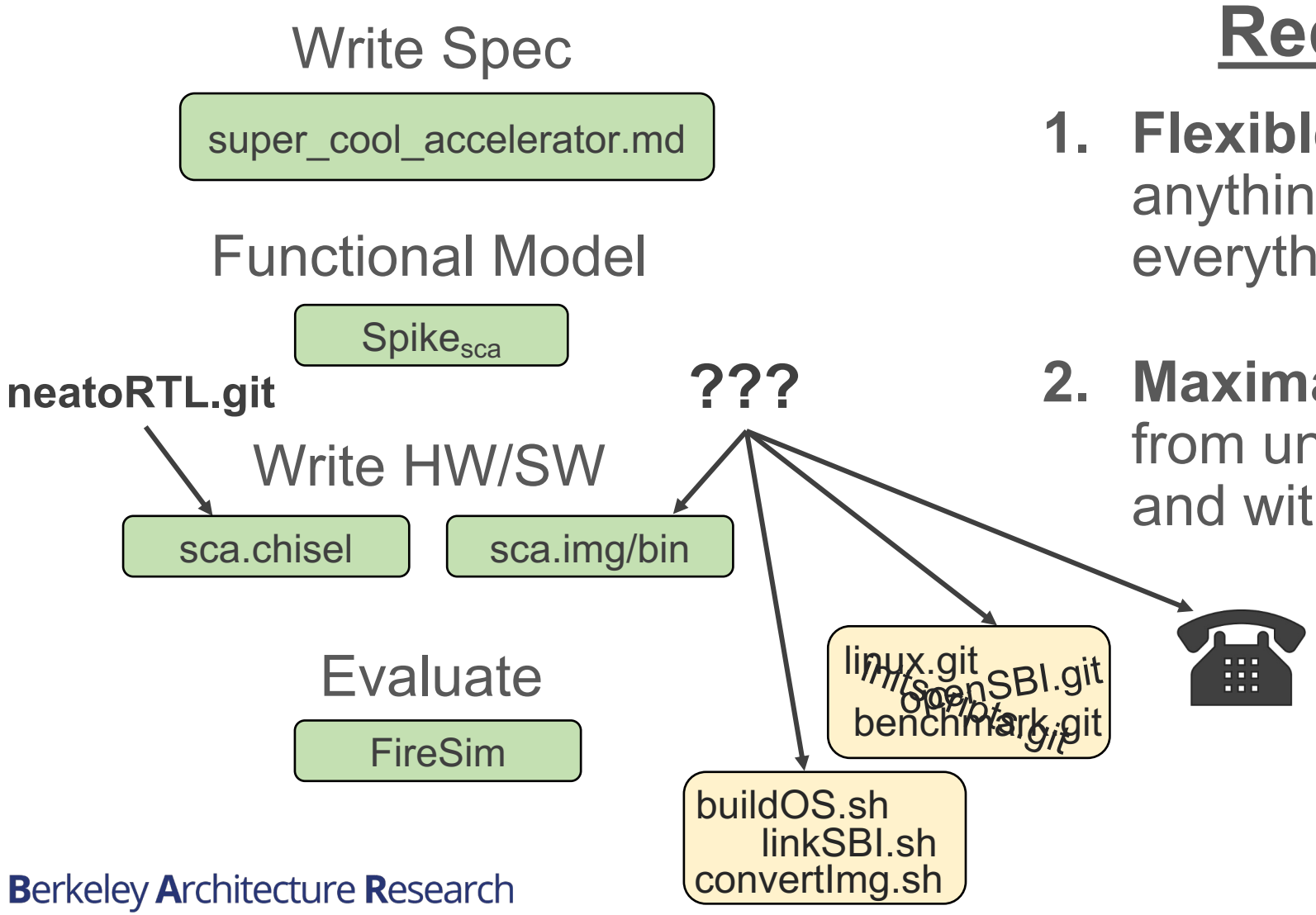
FireSim

Requirements

1. **Flexible Design:** Can change anything, without changing everything



Workload Management Tool Challenges

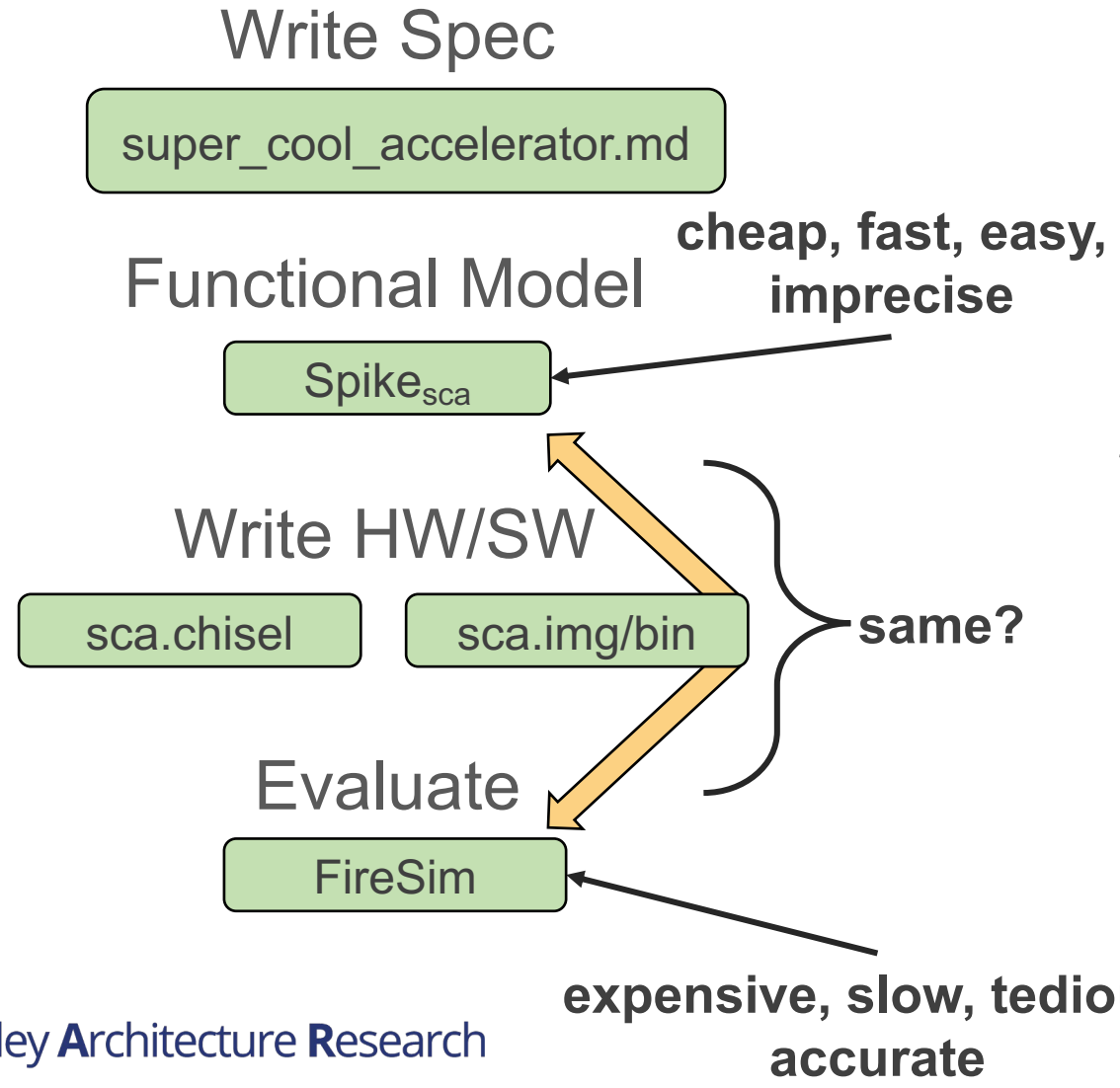


Requirements

1. **Flexible Design:** Can change anything, without changing everything
2. **Maximal Reuse:** Can rebuild from unambiguous description and without inside knowledge

Alice: How big was your image?
Bob: lol, idk. Like 256? maybe?
Alice: I'll just try something

Workload Management Tool Challenges

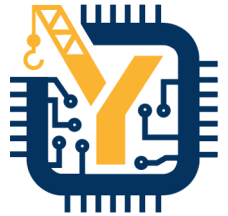


Requirements

1. **Flexible Design:** Can change anything, without changing everything
2. **Maximal Reuse:** Can rebuild from unambiguous description and without inside knowledge
3. **Flexible Simulation:** Minimize SW differences across simulation levels

FireMarshal Workflow

FireMarshal Workflow

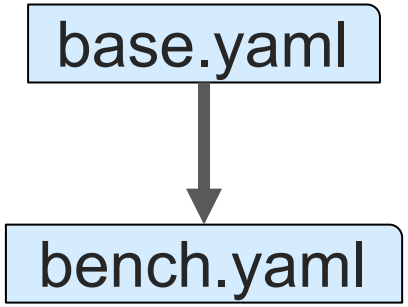


Specify	Build	Launch/Install	Test

Berkeley Architecture Research

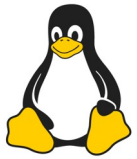
FireMarshal Workflow

Specify	Build	Launch/Install	Test



What's in a Workload?

Custom Sources



QEMU

OpenSBI

Inputs/Outputs

Overlays

```
/etc
  /init.d
    /S20foo
```

File Lists

```
["/res.csv",
 "/cfg.json]
```

Serial Console

```
# vda mounted
# run /init
experiment start
time: 50s
# poweroff
```

User Scripts

```
host_setup.sh
```

```
target_setup.sh
```

```
on_boot.sh
```

```
after_run.sh
```

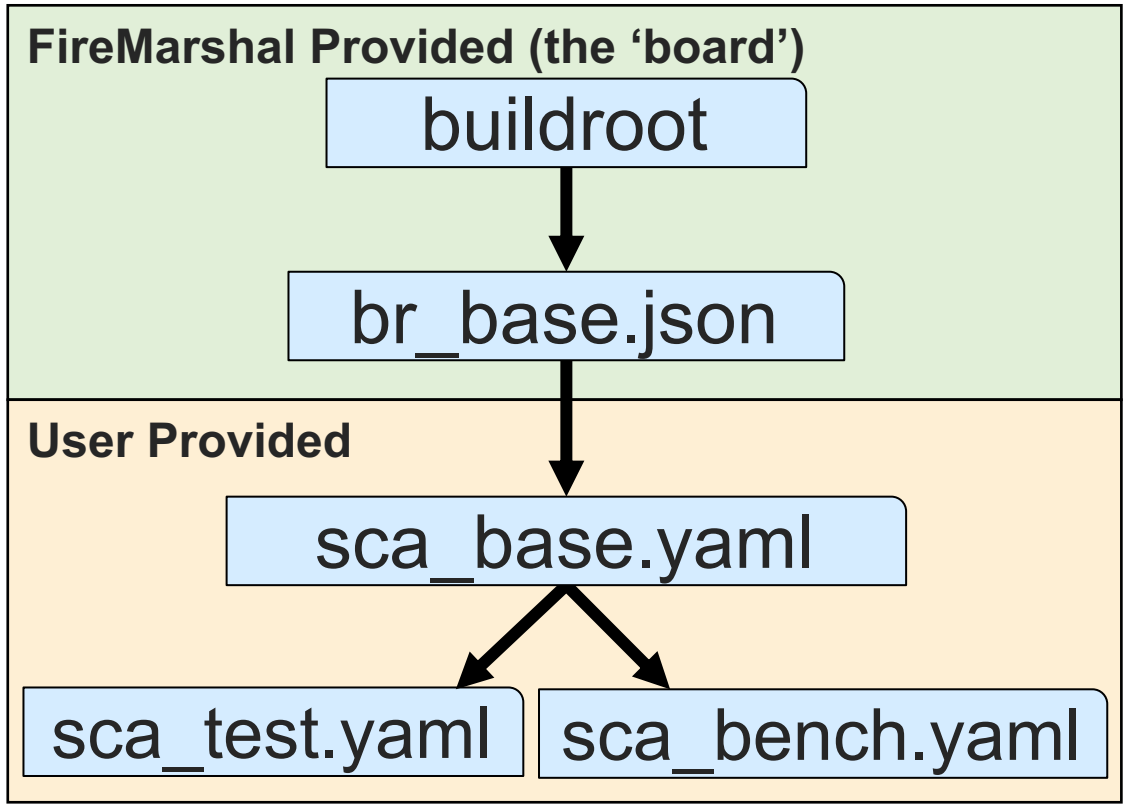
Miscellaneous

Option	Description
base	Start from a pre-existing workload - inherit all options unless explicitly overridden
overlay/files	Files to include in the image (e.g. utilities, benchmarks, config files, etc…)
host-init	Script to run before building (e.g. cross-compile)
guest-init	Script to run once on guest (e.g. install packages)
run/command	Script to run every time the image boots (e.g. default experiment)
outputs	Files to copy out of the image after an experiment
post-run-hook	Script to run on the output of the experiment (parse or format results)
linux	Linux customization options including Linux source directory, kernel configuration options to modify, as well as any needed kernel module sources
firmware	Firmware-related options including choice of firmware, and build options.
spike	custom Spike binary to use
spike/qemu-args	Additional arguments to pass to functional simulators
jobs	Additional, related images to build (e.g. each node of a networked workload)

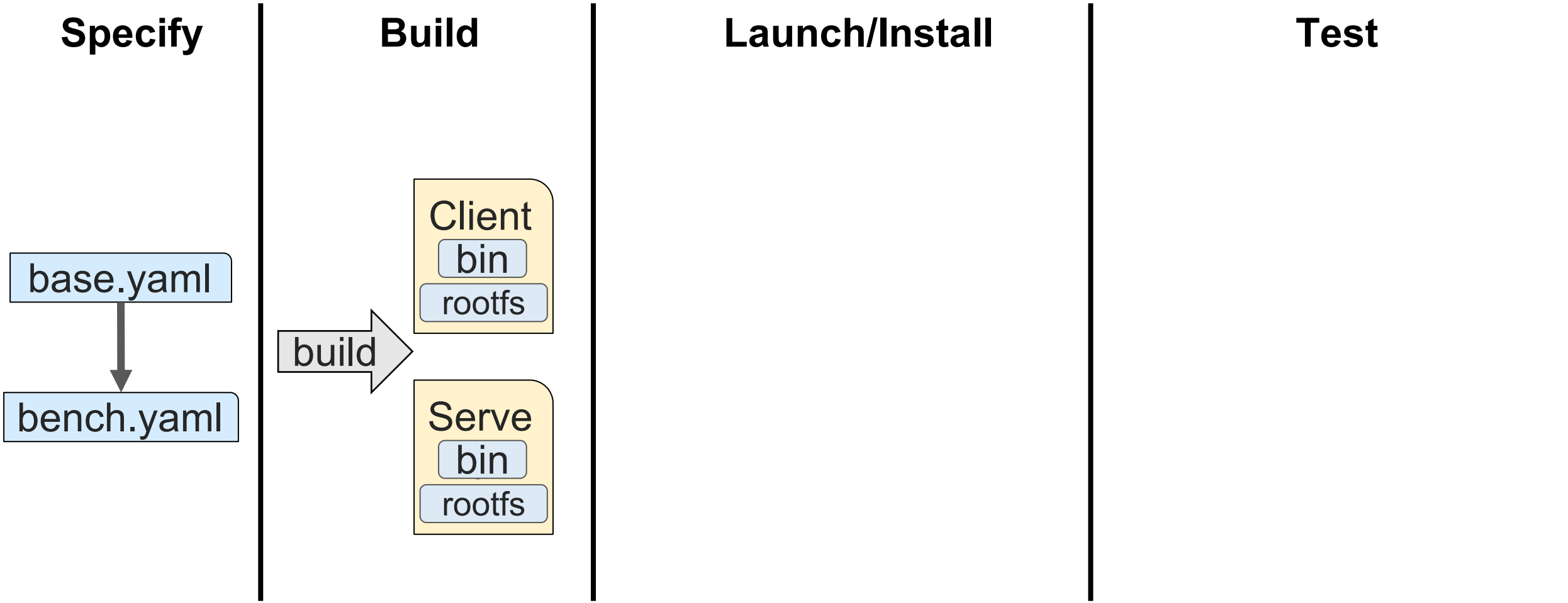
■ ■ ■

Enabling Reuse

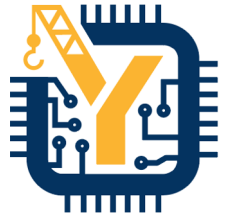
- **Inheritance:** Workloads are relative to a “base”
  - Inheritance rules vary by option
- **The “Board”:** Sane defaults for a hardware platform
- **User Hierarchies:** arbitrary inheritance tree



FireMarshal Workflow



Build

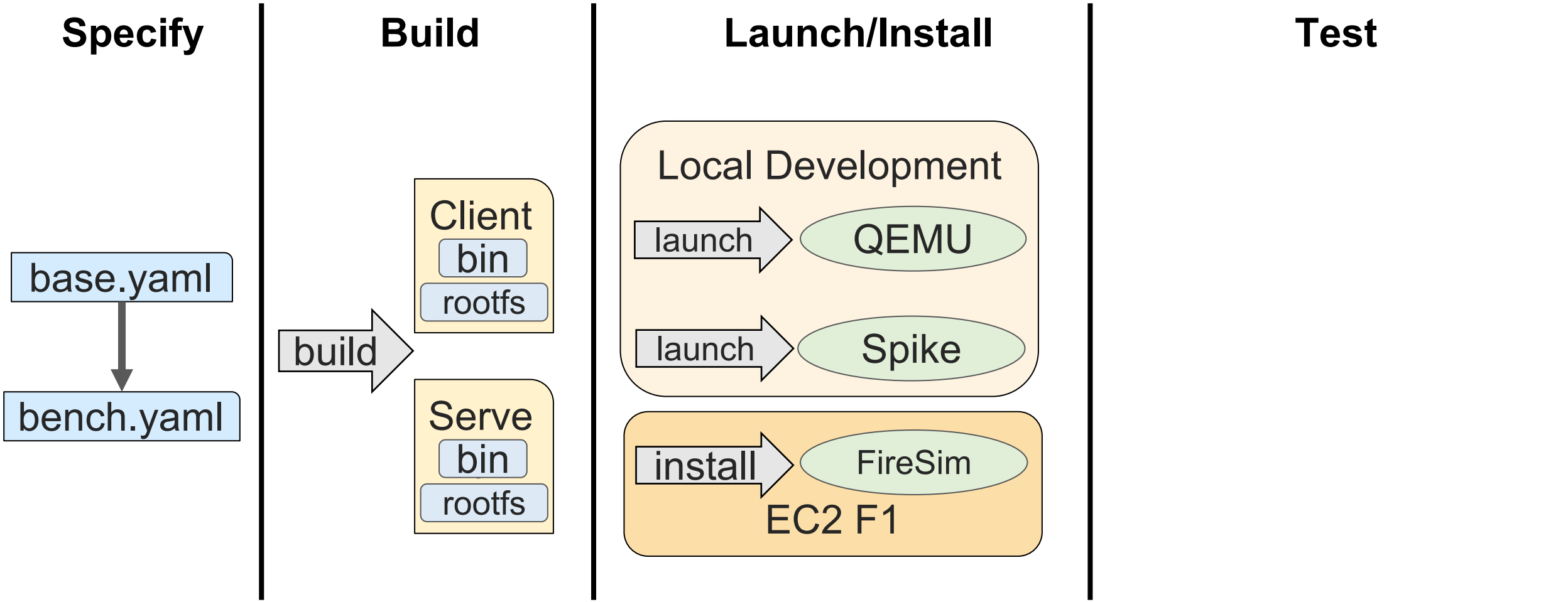


- Construct software artifacts
  - Make-like dependency tracking

- Steps
  1. Run workload setup script (host-init)
  2. Recursively construct parent images
  3. Compile Linux kernel and link firmware
  4. Copy parent image and modify as needed

```
$ marshal build example.json
. build.sh
Applying host-init: build.sh
-- BuildBusybox
. br-base/host-init.sh
Applying host-init: br-base/host-init.sh
-- br.04dc.img
. calc_br-base_dep
-- br-base.img
-- parent-bin
. calc_parent_dep
-- parent.img
. example-bin
. calc_example-test_dep
. example.img
Applying file list: ...
Applying run command: /root/test.sh
Log available at: logs/example-build.log
```

FireMarshal Workflow

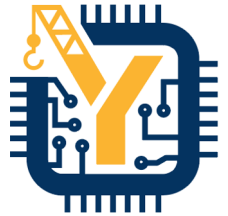


Launch

- Run workload in functional simulation
  - Currently supports QEMU and Spike
- Steps
  1. Boot Linux
  2. Automatically run benchmark
  3. Return serial and file outputs

```
$ marshal launch example.json
Linux version 5.7.0-rc3
earlycon: sbi0 at I/O port 0x0
printk: bootconsole [sbi0] enabled
...
launching workload run/command
Start basic test 1.
output[0]:203 ==? results[0]:203
output[1]:52 ==? results[1]:52
output[2]:27 ==? results[2]:27
Success!
execution took 8 cycles
reboot: Power down
Workload outputs available at:
output/example-launch/
Log available at: logs/example-launch.log
```

Install

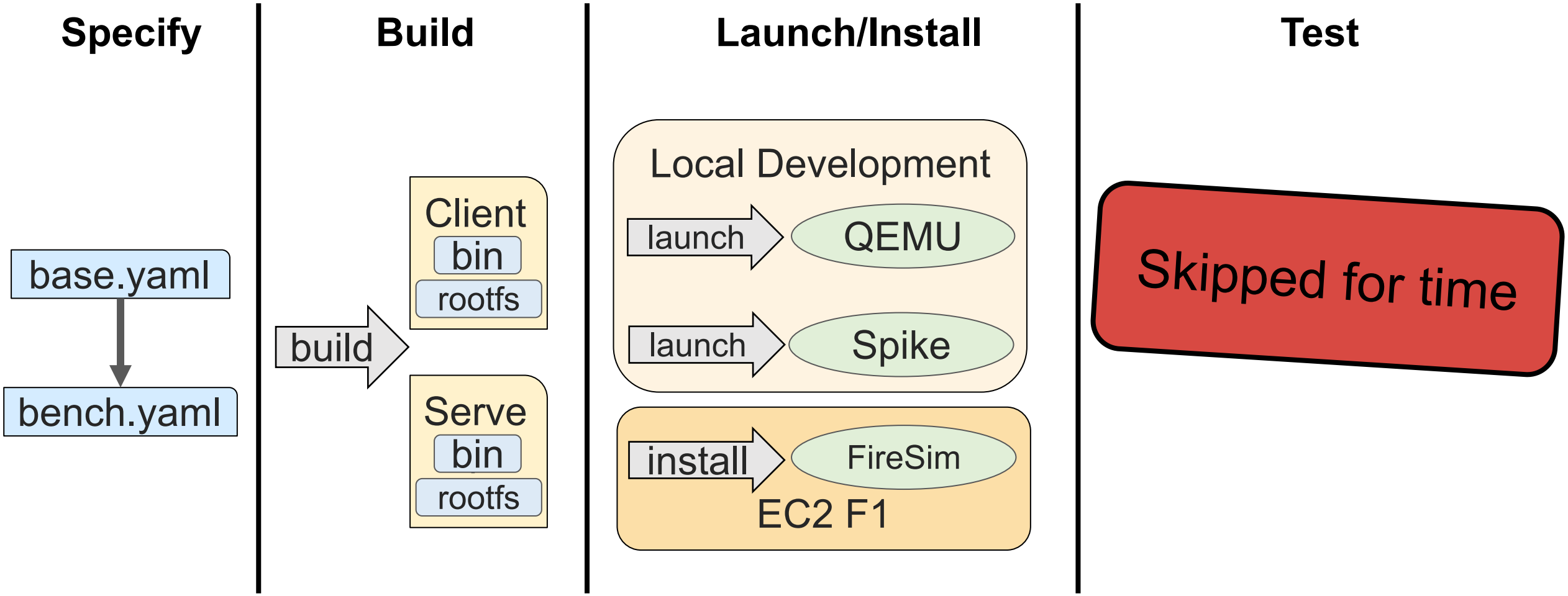


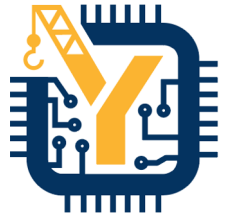
- Export workload to external simulator (e.g. FireSim)
  - Converts FireMarshal workload configuration to a FireSim configuration
- Does *not* rebuild workload: The same workload runs on all platforms
  - Simplify debugging
  - Ensure consistency

```
$ marshal install example.json

Workload installed to FireSim at
  firesim/example.json
Log available at:
  logs/example-install.log
```

FireMarshal Workflow





SHA3 Example Workloads

generators/sha3/software

Example Workload: SHA3 Workload Directory

$FDIR/generators/sha3/software

marshal-configs/

- FireMarshal configuration files

test-reference/

- Reference outputs for FireMarshal workloads

tests/

- unit tests for bare-metal and Linux

jtr/

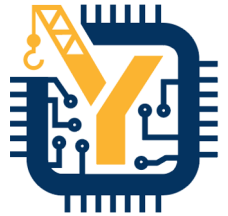
- An end-to-end benchmark

linux/

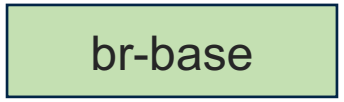
- A custom Linux kernel for our accelerator

```
[chipyard]$ pwd
/data/repos/chipyard/generators/sha3
[chipyard]$
[chipyard]$ ls software/marshal-configs/
sha3-bare-rocc.yaml        sha3-linux-jtr-test.yaml  sha3-linux.yaml
sha3-bare-sw.yaml          sha3-linux-jtr.yaml
sha3-linux-jtr-crack.yaml  sha3-linux-test.yaml
[chipyard]$
[chipyard]$ ls software/tests
bare  common.mk  linux  src
[chipyard]$
[chipyard]$ ls software/jtr/
bench.sh  buildroot-config  buildroot-external  crack.sh  overlay
[chipyard]$
[chipyard]$ ls software/linux/
arch           drivers  kernel          Module.symvers  System.map
block          fs       lib             net             tools
certs          include  LICENSES        README          usr
COPYING        init     MAINTAINERS     samples         virt
CREDITS        ipc      Makefile        scripts         vmlinux
crypto         Kbuild   mm              security        vmlinux.o
Documentation  Kconfig  modules.builtin  sound
[chipyard]$
```

Workload Inheritance



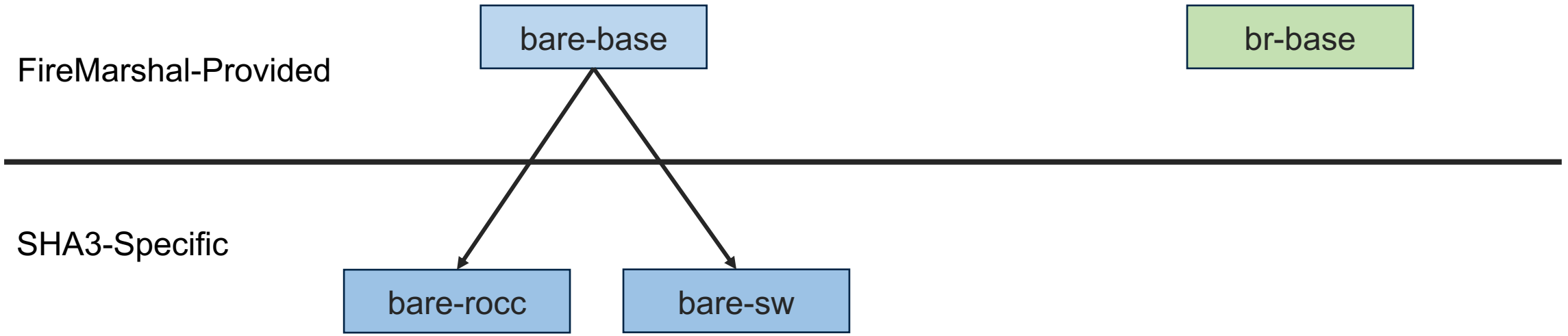
FireMarshal-Provided



SHA3-Specific

- FireMarshal provides several “base” workloads to act as starting points for user workloads.
- Today, we will use:
  - “br-base”: Interactive Buildroot-based Linux workload. This is used for FireSim’s “linux-uniform” default workload.
  - “bare”: A trivial workload used for bare-metal experiments

Workload Inheritance



Bare-metal unit tests

Example Workload:
SHA3 Bare-Metal Unit Test

sha3-bare-rocc.yaml

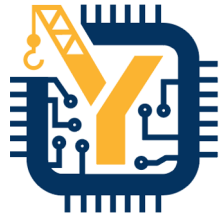
```
{
"name" : "sha3-bare-rocc",
"workdir" : "..",
"base" : "bare-base.json",
"host-init" : "build.sh",
"bin" : "tests/bare/sha3-rocc.riscv",
"spike" : "spike-local/bin/spike",
"spike-args" : "--extension=sha3"
}
```

Script to run when building this workload (build.sh cross-compiles the unit test)

Hard-coded binary to use (produced by build.sh)

Custom Spike boot arguments (to enable the sha3 functional model)

Example Workload:
SHA3 Bare-Metal Unit Test

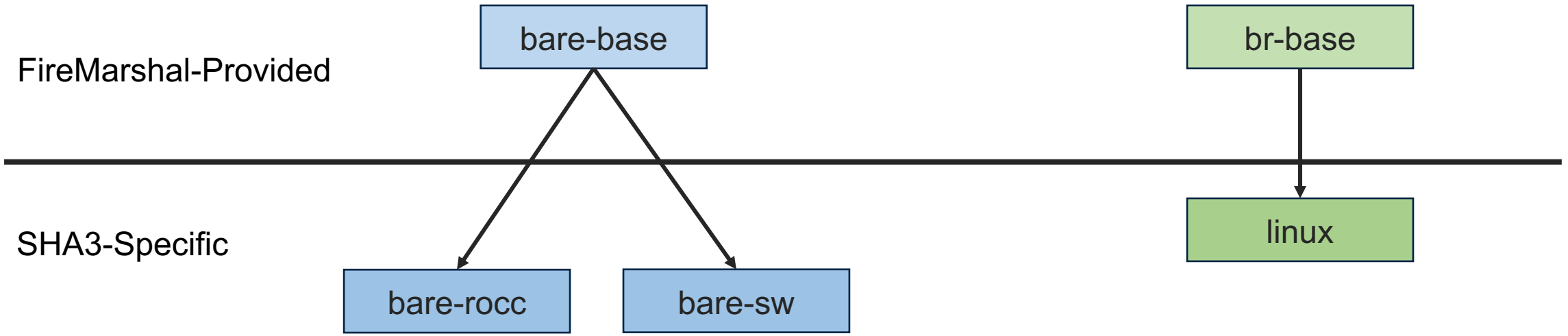


```
(base) [xarc@xarc0 firesim-software]$ ./marshal build workloads/sha3-bare.json
To check on progress, either call marshal with '-v' or see the live output at:
/data/repos/firesim-software/logs/sha3-bare-build-2019-08-29--00-24-07-67ARZD490JUNSSAX.log
Applying host-init: /data/repos/firesim-software/workloads/sha3/build.sh
.  /data/repos/firesim-software/workloads/sha3/bare/sha3-rocc.riscv
Log available at: /data/repos/firesim-software/logs/sha3-bare-build-2019-08-29--00-24-07-67ARZD490JUNSSAX.log
(base) [xarc@xarc0 firesim-software]$ ./marshal launch -s workloads/sha3-bare.json
To check on progress, either call marshal with '-v' or see the live output at:
/data/repos/firesim-software/logs/sha3-bare-launch-2019-08-29--00-24-16-E3KF36KFH8ZFUYNV.log
Running: /data/repos/firesim-software/workloads/sha3/spike-local/bin/spike --extension=sha3 -p4 -m16384 /data/repos/firesim-software/workloads/sha3/bare/sha3-rocc.riscv
start basic test 1.
output[0]:221 ==? results[0]:221
output[1]:204 ==? results[1]:204
output[2]:157 ==? results[2]:157
output[3]:217 ==? results[3]:217
output[4]:67 ==? results[4]:67
```

```
$ cd ~/chipyard-afternoon
$ cd generators/sha3/software/marshal-configs
$ marshal -v build sha3-bare-rocc.yaml
$ marshal launch -s sha3-bare-rocc.yaml
```

```
output[25]:61 ==? results[25]:61
output[26]:3 ==? results[26]:3
output[27]:149 ==? results[27]:149
output[28]:137 ==? results[28]:137
output[29]:42 ==? results[29]:42
output[30]:57 ==? results[30]:57
output[31]:238 ==? results[31]:238
success!
```

Workload Inheritance



Interactive Linux base workload

- Specifies all common settings for sha3 workloads

Example Workload:
SHA3 on Linux (interactive base workload)

sha3-linux.yaml

```
{
"name" : "sha3-linux",
"base" : "br-base.json",
"workdir" : "..",
"host-init" : "build.sh",
"files" : [
  ["tests/linux/sha3-sw.rv", "/root/sha3-sw"],
  ["tests/linux/sha3-rocc.rv","/root/sha3-rocc"],
],
"linux-src" : "linux",
"spike" : "spike-local/bin/spike",
"spike-args" : "--extension=sha3"
}
```

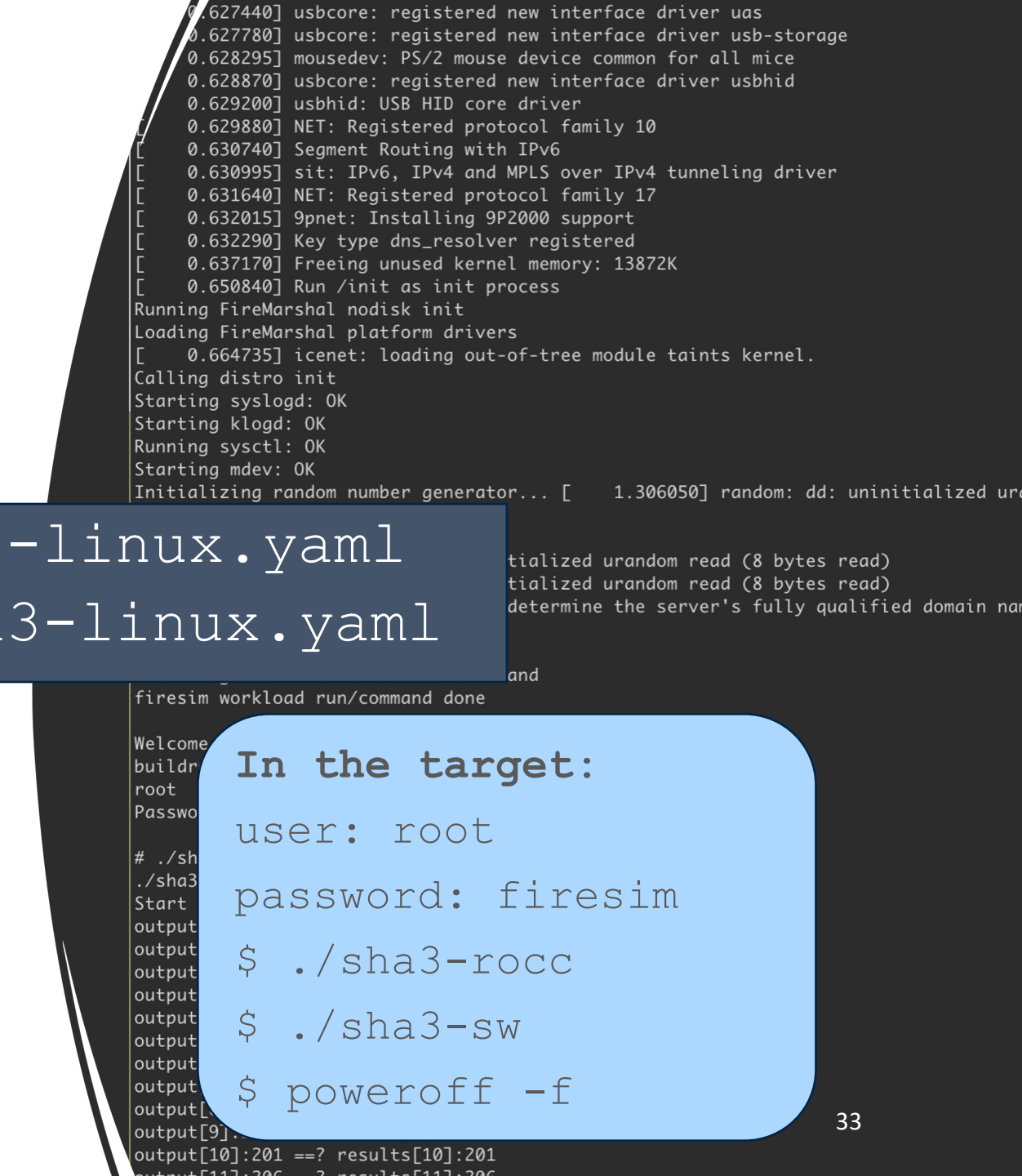
Run by Marshal at build time (cross-compiles the Linux benchmarks)

Files to copy into the guest root filesystem (the pre-compiled benchmarks in this case)

Custom Linux source to compile (needed in this case to manage the accelerator)

Example Workload: SHA3 on Linux

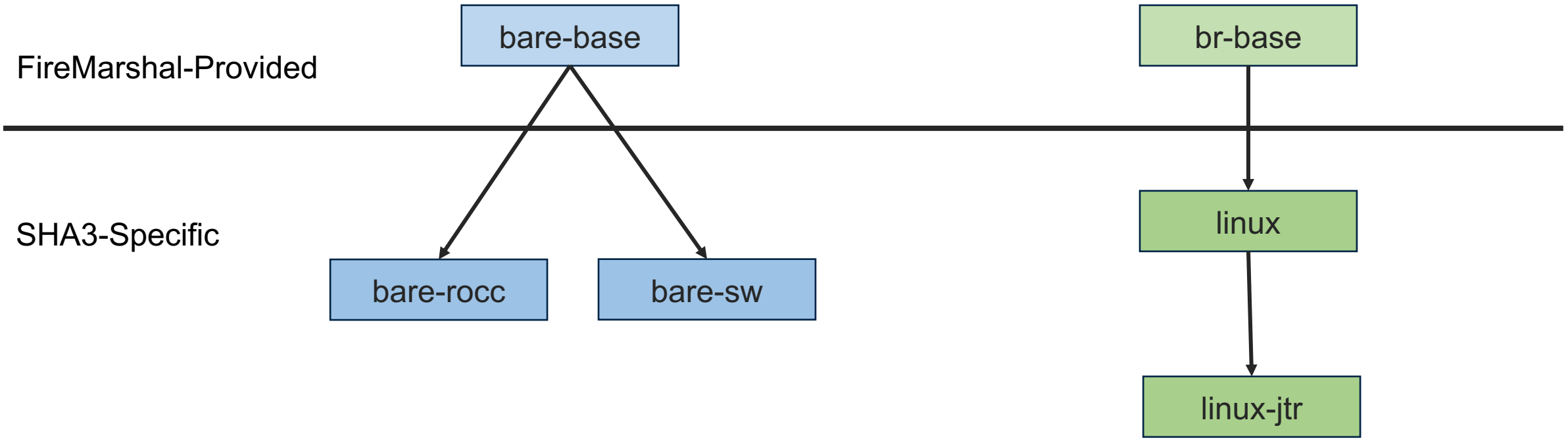
```
$ marshal -d -v build sha3-linux.yaml
$ marshal -d launch -s sha3-linux.yaml
```



In the target:

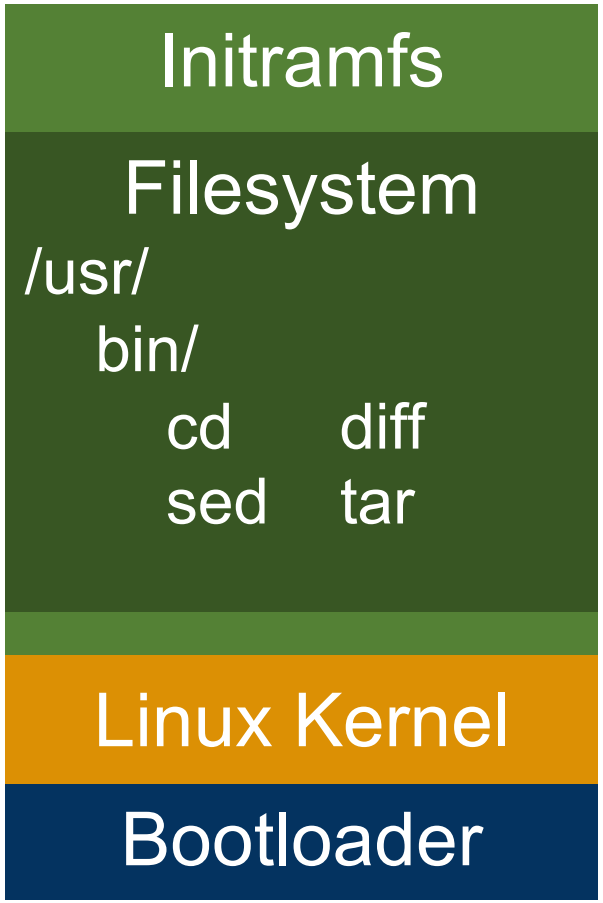
```
user: root
password: firesim
$ ./sha3-rocc
$ ./sha3-sw
$ poweroff -f
```

Workload Inheritance



End-to-end Benchmark: John The Ripper

Linux Build Internals:
Overlays

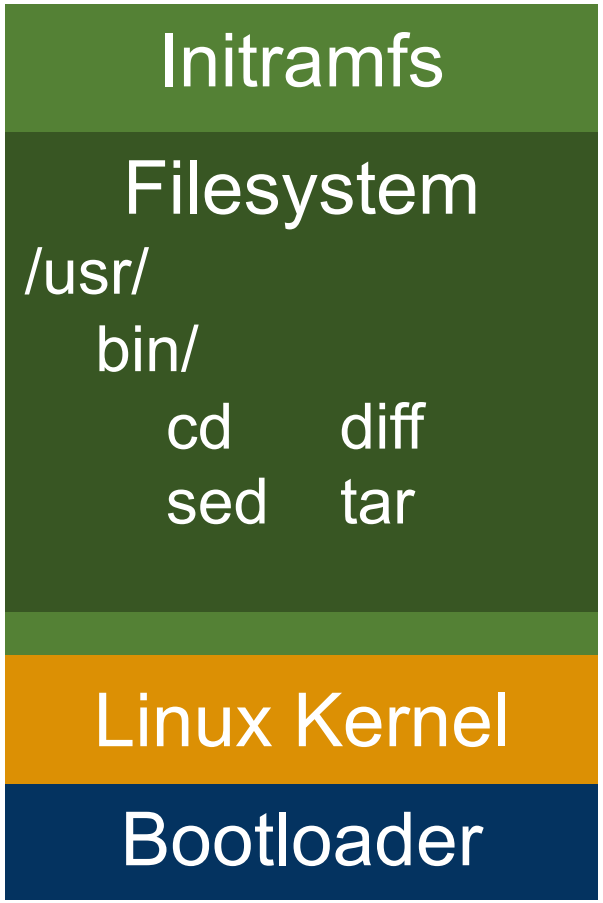


+

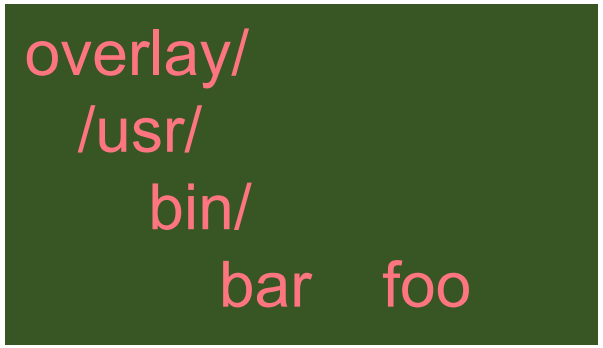
- Problem: How to specify many files on workload binary?

```
{
"name" : "overlay-example",
"files" : [
  ["tests/example/foo.rv", "/usr/bin/foo"],
  ["tests/example/bar.rv","/usr/bin/bar"], ],
}
```

Linux Build Internals:
Overlays



+

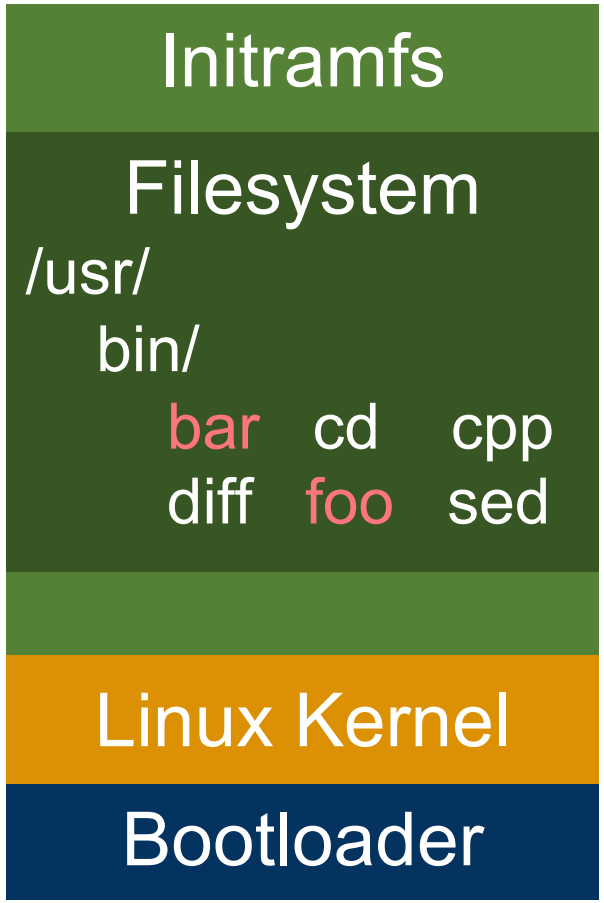


- Problem: How to specify many files on workload binary?

```
{
"name" : "overlay-example",
"overlay" : "test/overlay"
}
```

- Solution: Provide an overlay directory with all necessary files.

Linux Build Internals:
Overlays



- Problem: How to specify many files on workload binary?

```
{
"name" : "overlay-example",
"overlay" : "test/overlay"
}
```

- Solution: Provide an overlay directory with all necessary files.

Example Workload:
Linux-based Benchmark – John the Ripper

```
{
  "name" : "sha3-linux-jtr",
  "base" : "sha3-linux.yaml",
  "workdir" : "..",
  "distro" : {
    "name": "br",
    "opts": {
      "configs" : [ "jtr/buildroot-config" ],
      "environment" : {
        "BR2_EXTERNAL":
"${SHA3_LINUX_JTR_PATH}/jtr/buildroot-
external"
      }
    }
  },
  "overlay" : "jtr/overlay"
}
```

Inherit from sha3-linux again. Only need to specify that stuff once.

Add John The Ripper to our underlying buildroot distro

John The Ripper must be installed to work correctly. The overlay allows us to specify a complex directory structure.

Example Workl
End-to-end Benchmark

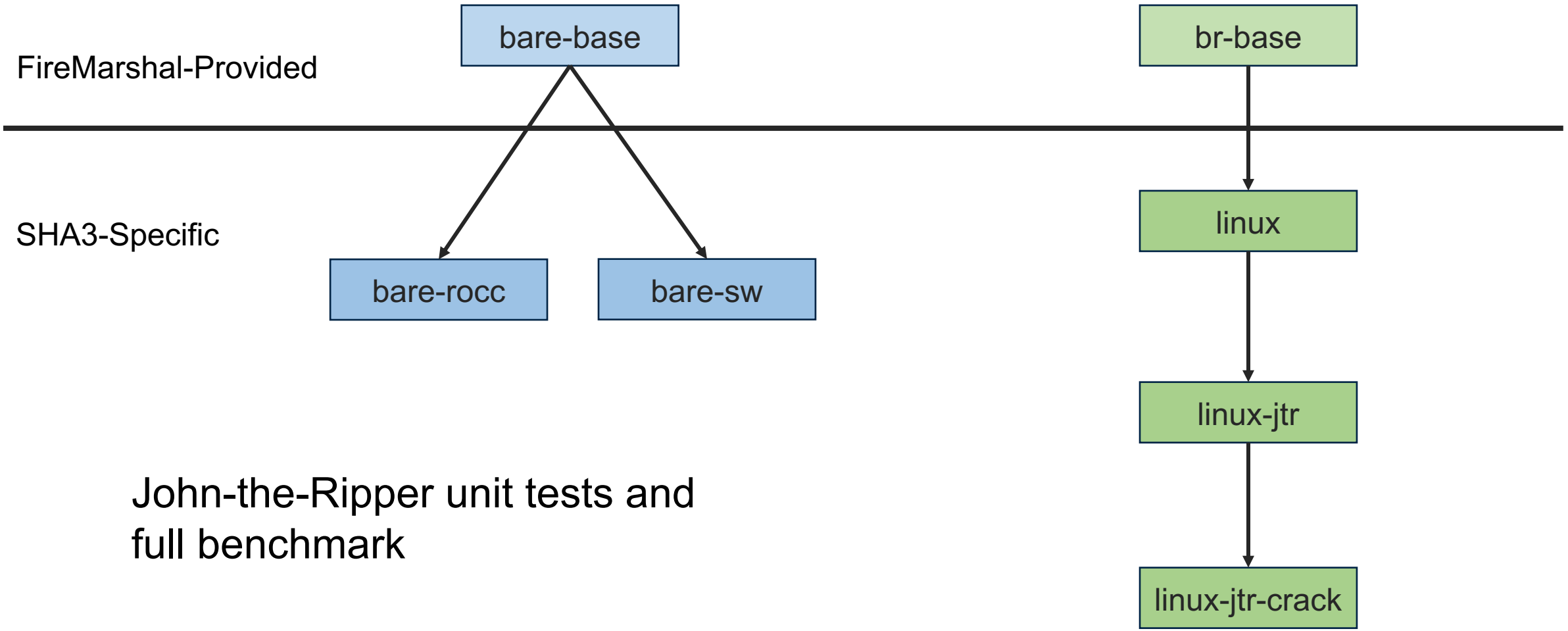
Skipped for Time
(try it out yourself!)

In the target:

```
user: root
password: firesim
$ cd sha3
$ john --format=Raw-SHA3-256-rocc short.txt
$ poweroff -f
```

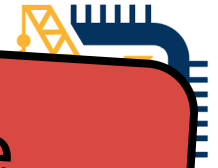
```
$ marshal -d build sha3-linux-jtr.yaml
$ marshal -d launch -s sha3-linux-jtr.yaml
```

Workload Inheritance



John-the-Ripper unit tests and full benchmark

Example Workload:
Linux-based Benchmark – John the Ripper



Skipped for Time
(try it out yourself!)

sha3-linux-jtr-crack.yaml

```
{
  "name" : "sha3-linux-jtr-crack",
  "base" : "sha3-linux-jtr.yaml",
  "workdir" : "..",
  "run" : "jtr/crack.sh"
}
```

Run the full benchmark

```
marshal -d launch
-s sha3-linux-jtr-crack.yaml
```

More Complex Use-Cases

Multi-Node Workloads
(“jobs”)

job-example.yaml

```
{
  "name" : "job-example",
  "base" : "br-base.json",
  "jobs" : [
    { "name" : "node0",
      "command" : "ping -c 1 172.16.0.3",
    },
    { "name" : "node1",
      "command" : "ping -c 1 172.16.0.2",
    }
  ]
}
```

- Each job runs on a single node in multi-node simulations.
- Described the same as any workload
  - implicitly ‘base’d on the enclosing workload
- Runs in parallel in SW simulation in new Firemarshal release
  - FireSim already supported a network

Native Initialization
(“guest-init”)

guest-init-example.yaml

```
{
  ”name” : “guest-init-example”,
  “base” : “fedora-base.json”,
  “guest-init” : “init.sh”
}
```

init.sh

```
#!/bin/bash
yum install -y blas python3 …

cd cafe2_src/
make
```

- “guest-init” script is run once on the guest during build
  - Run in QEMU
  - Can access internet
- Useful for installing packages and/or natively compiling benchmarks

Automatic Results Processing
(“post-run-hook”)

results-example.yaml

```
{
  ”name” : “results-example”,
  “base” : “mytest.yaml”,
  “outputs” : [“/root/res.csv”],
  “post-run-hook” : “results.py”
}
```

“post-run-hook” executed on the host after every run

- Good for post-processing of more complex experiments

results.py

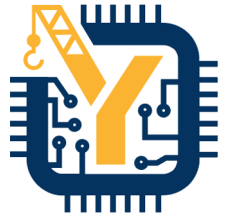
```
#!/usr/bin/env python
from pathlib import Path
import csv

resultPath = Path(sys.argv[1]) /
  ‘results-example’ / ‘res.csv’

processResult(resultPath)
```

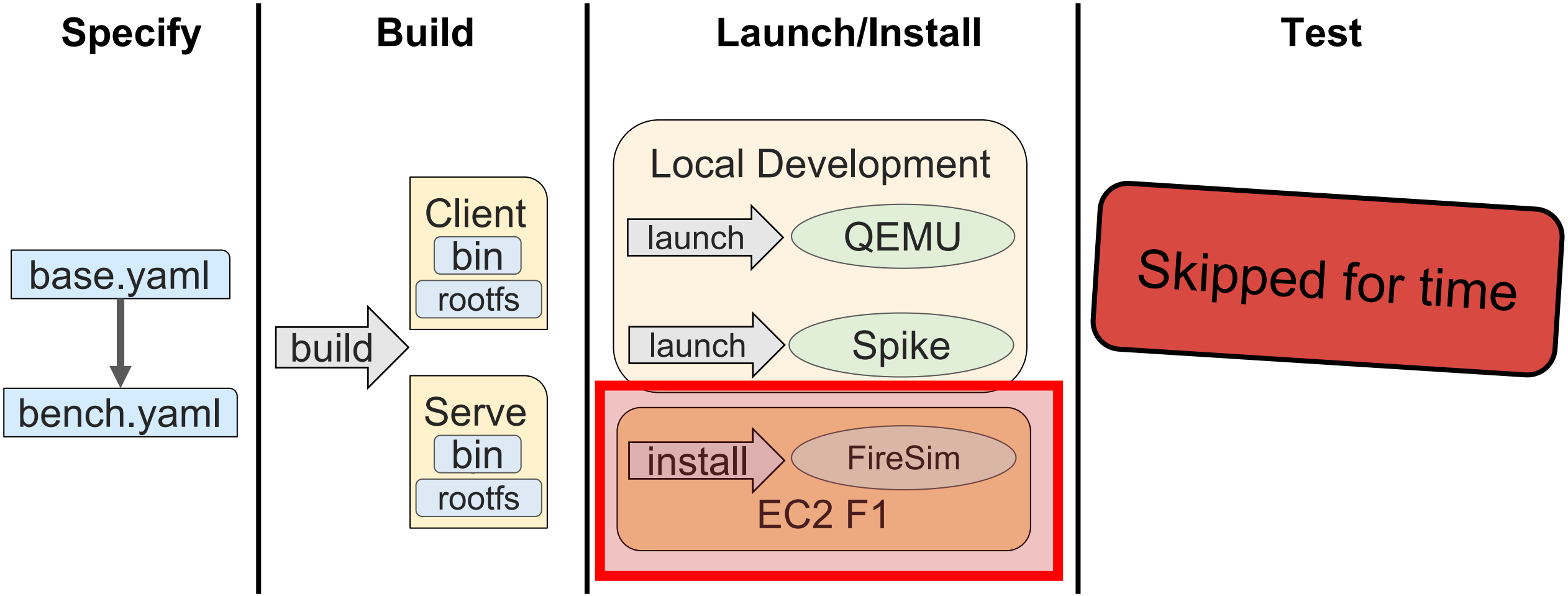
Path to the results directory passed to the script

Do anything you want with the results. For example, copy to a known location, or sanity check



Running Workloads on FireSim

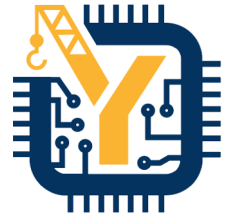
FireMarshal Workflow



Installing Workloads to FireSim

```
$ marshal install sha3*.yaml
$ cd ~/chipyard-afternoon
$ cd sims/firesim/deploy/workloads/
$ cat sha3-linux.json
```

```
rshal-configs]$ marshal install *.yaml
 check on progress, either call marshal with '-v' or see the
/data/repos/chipyard/software/firemarshal/logs/sha3-bare-rocc-i
Workload installed to FireSim at /data/repos/chipyard/sims/fire
To check on progress, either call marshal with '-v' or see the
/data/repos/chipyard/software/firemarshal/logs/sha3-bare-sw-ins
Workload installed to FireSim at /data/repos/chipyard/sims/fire
To check on progress, either call marshal with '-v' or see the
/data/repos/chipyard/software/firemarshal/logs/sha3-linux-jtr-c
Workload installed to FireSim at /data/repos/chipyard/sims/fire
To check on progress, either call marshal with '-v' or see the
/data/repos/chipyard/software/firemarshal/logs/sha3-linux-jtr-t
Workload installed to FireSim at /data/repos/chipyard/sims/fire
To check on progress, either call marshal with '-v' or see the
/data/repos/chipyard/software/firemarshal/logs/sha3-linux-jtr-i
Workload installed to FireSim at /data/repos/chipyard/sims/fire
To check on progress, either call marshal with '-v' or see the
/data/repos/chipyard/software/firemarshal/logs/sha3-linux-test-
Workload installed to FireSim at /data/repos/chipyard/sims/fire
To check on progress, either call marshal with '-v' or see the
/data/repos/chipyard/software/firemarshal/logs/sha3-linux-insta
Workload installed to FireSim at /data/repos/chipyard/sims/fire
Log available at: /data/repos/chipyard/software/firemarshal/log
[marshal-configs]$ cd ../../../../sims/firesim/deploy/workloads
[workloads]$ cat sha3-linux.json
{
        "benchmark_name": "sha3-linux",
        "common_simulation_outputs": [
                "uartlog"
        ],
        "common_bootbinary": "../../../../../software/firemarsh
        "common_rootfs": "../../../../../software/firemarshal/i
}[workloads]$
```



Lots More Features!

https://firemarshal.readthedocs.io/en/stable/