WLTests
A LISA-based evaluation tool

Chris Redpath
What is WLTests?

A tool to help evaluating the impact of kernel changes, built using LISA & Workload-Automation

• Available as part of LISA on Github
  – https://github.com/ARM-software/lisa
• Able to build and flash kernels to supported targets
• Performs multiple executions of workloads using v3 of Workload-Automation for test execution
• Collects energy measurements using devlib-supported meters
• iPython Notebooks provided to compare selected commits
• Supports Hikey620, Hikey960 & Pixel 2
• Started as a comparison tool for PELT vs WALT studies
Running WLTests

1. Make sure your device is configured the way you want and has a userspace matching the base kernel
   – This is really important, WLTests does not do device configuration (except for what WA3 can do)
2. Prepare your kernel branches to test
   – Kernel commits are identified in test results with a branch name, or a SHA1 if none is found
3. Provide a list of the commits to test
   – Use the output of git log –oneline, you can annotate (see later)
4. Provide an agenda to WA3
   – A collection is provided which corresponds to what we use
5. Run the tests using ‘lisa-wltest-series’
   – Example later
6. Point the provided notebooks at your test data
Selecting Commits to Test

Generate a series.sha1 file

At the simplest level, provide the output of `git log --oneline`

Can add order identifiers to the start of a line if desired:

- ‘00:’ skip this line
- ‘01:’ test this commit first
- ‘02:’ test this commit second
- ‘nn:’ test this commit nth
Choose an agenda
WA3 uses yaml agendas, much like WA2 did

Provided Agendas

- example-*.yaml
  - Simple examples, commented

- sched-evaluation-*.yaml
  - The agenda files we use for testing EAS patches
  - Use sched-evaluation-full.yaml if possible
  - Customize as necessary
Start the run
lisa-wltest-series

Help is available in the tool

Requires a lot of arguments:

- --platform
- --kernel_src
- --series
- --wa_agenda
- --device

Note: Some devices take longer than the default 180s to reboot from adb into fastboot mode – you can override it with --reboot-timeout
Let WLTests finish running tests...
Adding a new device to WLTests

You need to provide 3 scripts, in a platform directory, written in bash.

• Definitions
  – Specify locations of tools and build configuration
• build_images
  – With the kernel checked out at a specific commit, produce the input expected by ‘flash_images’
• flash_images
  – Take the output from ‘build_images’ and flash it to the specified device
Configurations Supplied with WLTests

Hikey620 & Hikey960

- Use android image builder to assemble ramdisk.img and cross-compiled kernel into a boot.img
- Use fastboot to flash boot.img
- Use adb to drive testing

Pixel2

- Pixel2 relies on binaries in the vendor partition which also contains modules
- For a successful flash, need to update modules on system partition after compilation
- Device must be rooted to allow adb remount of root dir

If you’re adding a new device, start by cloning one of the Hikey sets – it’s the least specific and least complicated.
Hikey620 Platform

‘definitions’ script

- Platform Name
- Boot Image Configuration
  - Boot Command line
  - Various variables used by our mkbootimg wrapper script
- Toolchain Configuration
  - Defconfig
  - Cross Compiler
Hikey620 Platform

‘build_images’ script

• Load helpers (common) and definitions

• Build a kernel and a boot image
  • Command line used in boot image creation taken from environment
  • Ramdisk location set in environment

• Build products are placed in:
  • tools/wltests/platforms/<platform name>/artefacts

```bash
#!/bin/bash

SCRIPT_DIR=$(dirname $(realpath -s $0))
BASE_DIR="$SCRIPT_DIR/../../",
source "$BASE_DIR/helpers"
source "$PLATFORM_PATH/definitions"

# Build all images required to test a new kernel

./build --image --ctbs --silent; ERROR=$?
./build --image --silent; ERROR=$?
[ $ERROR -eq 0 ] || exit $ERROR

./android/create_boot_img.sh; ERROR=$?
[ $ERROR -eq 0 ] || exit $ERROR
```
Hikey620 Platform

‘flash_images’ script

• Load helpers (common) and definitions

• Flash boot image with fastboot
  • Device will be in fastboot mode before script is called
  • Working directory is platform artefacts directory
  • Loop through every image file in the working directory
  • Assumes images are named after their partition – we strip .img to get partition name

```bash
#!/bin/bash

SCRIPT_DIR=$(
  dirname $(
    realpath -s $0)
)
BASE_DIR="$SCRIPT_DIR/../../..
"
source "$(BASE_DIR)/helpers"
source "$(PLATFORM_PATH)/definitions"

MODE=${1:-FASTBOOT}
if $MODE == FASTBOOT
  echo "Device will be in fastboot mode before script is called"
fi

# Build all images required to test a new kernel
ls *.img > /dev/null; ERROR=$?
if [ $ERROR -ne 0 ]; then
  echo "No images to flash in $PWD"
  exit SENDENT
fi

for IMAGE in $(ls *.img); do
  PARTITION=$(basename $IMAGE .img)
  echo $PARTITION
  c_info "Flashing $IMAGE on $PARTITION partition..."
  $FASTBOOT flash $PARTITION $IMAGE; ERROR=$?
  if [ $ERROR -eq 0 ]; then
    echo "Flash successful"
  fi
done
```