Distributed Cross Platform Test Automation

William Marone

Agenda

- **Goals and Problems**
- Solutions
- User Interface
- Future Enhancement
- Conclusion

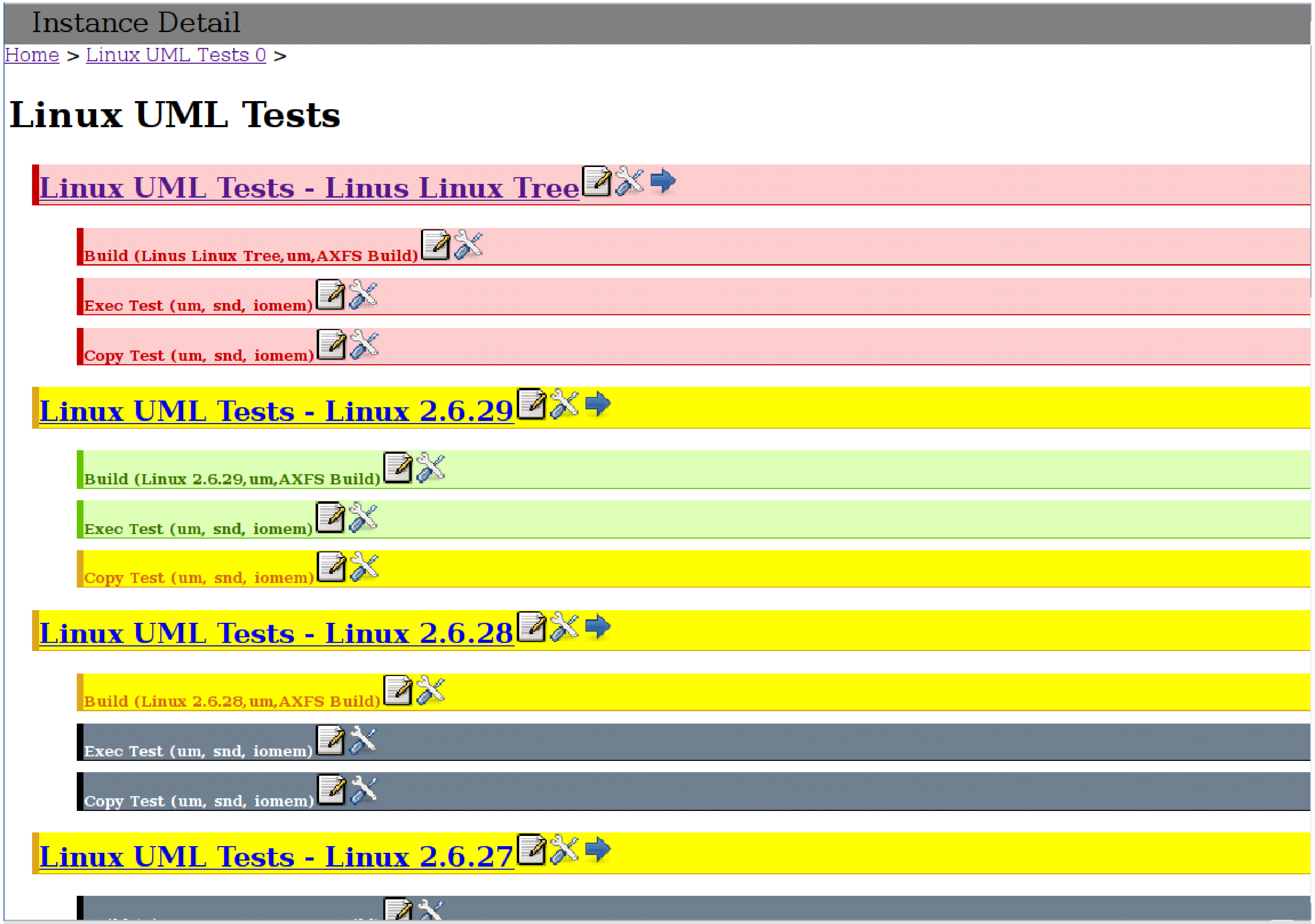
Goals

- Test AXFS functionality
- Quickly build and run tests
- Across multiple architectures and kernels
- Do so automatically and repeatably
- Display results clearly

Example Case

- Execute and Copy files (2)
- Different file system configurations (4)
- Different mount options (2)
- Kernel Config options (2)
- Multiple architectures (2)
- Numerous Kernels (20)

1280 Test Cases!



Instance Detail

Home > Linux UML Tests 0 >

Linux UML Tests

Linux UML Tests - Linus Linux Tree

Build (Linus Linux Tree,um,AXFS Build)

Exec Test (um, snd, iomem)

Copy Test (um, snd, iomem)

Linux UML Tests - Linux 2.6.29

Build (Linux 2.6.29,um,AXFS Build)

Exec Test (um, snd, iomem)

Copy Test (um, snd, iomem)

Linux UML Tests - Linux 2.6.28

Build (Linux 2.6.28,um,AXFS Build)

Exec Test (um, snd, iomem)

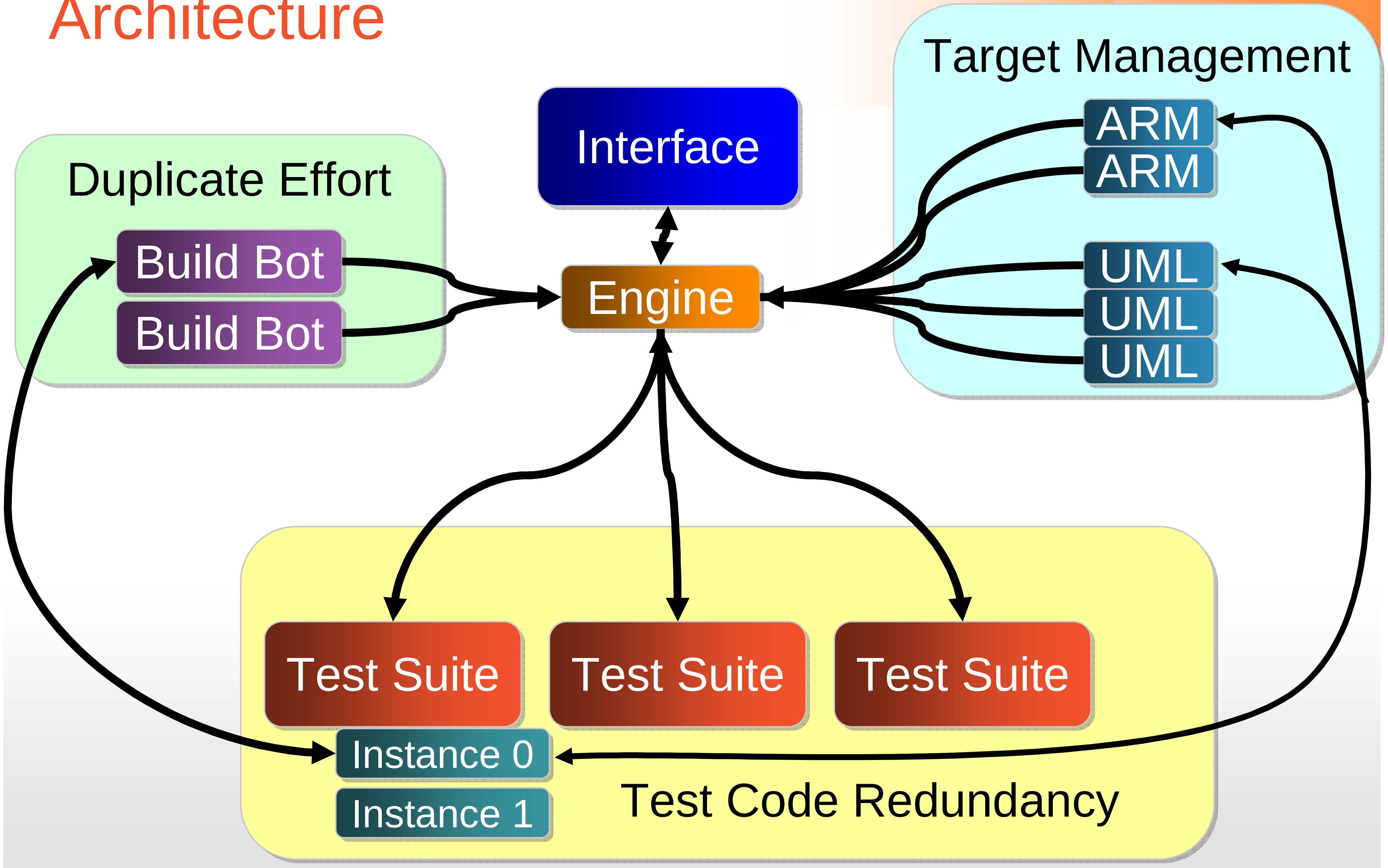
Copy Test (um, snd, iomem)

Linux UML Tests - Linux 2.6.27

Problems

- Internal redundancy in test code
  - Numerous kernels
  - Multiple tests
- Duplicated effort
  - Kernel builds
  - Checking out sources
  - Building file system images
- Targets need management
  - Redundant programming
  - Platform attributes may vary
  - Inconsistent interface

Architecture

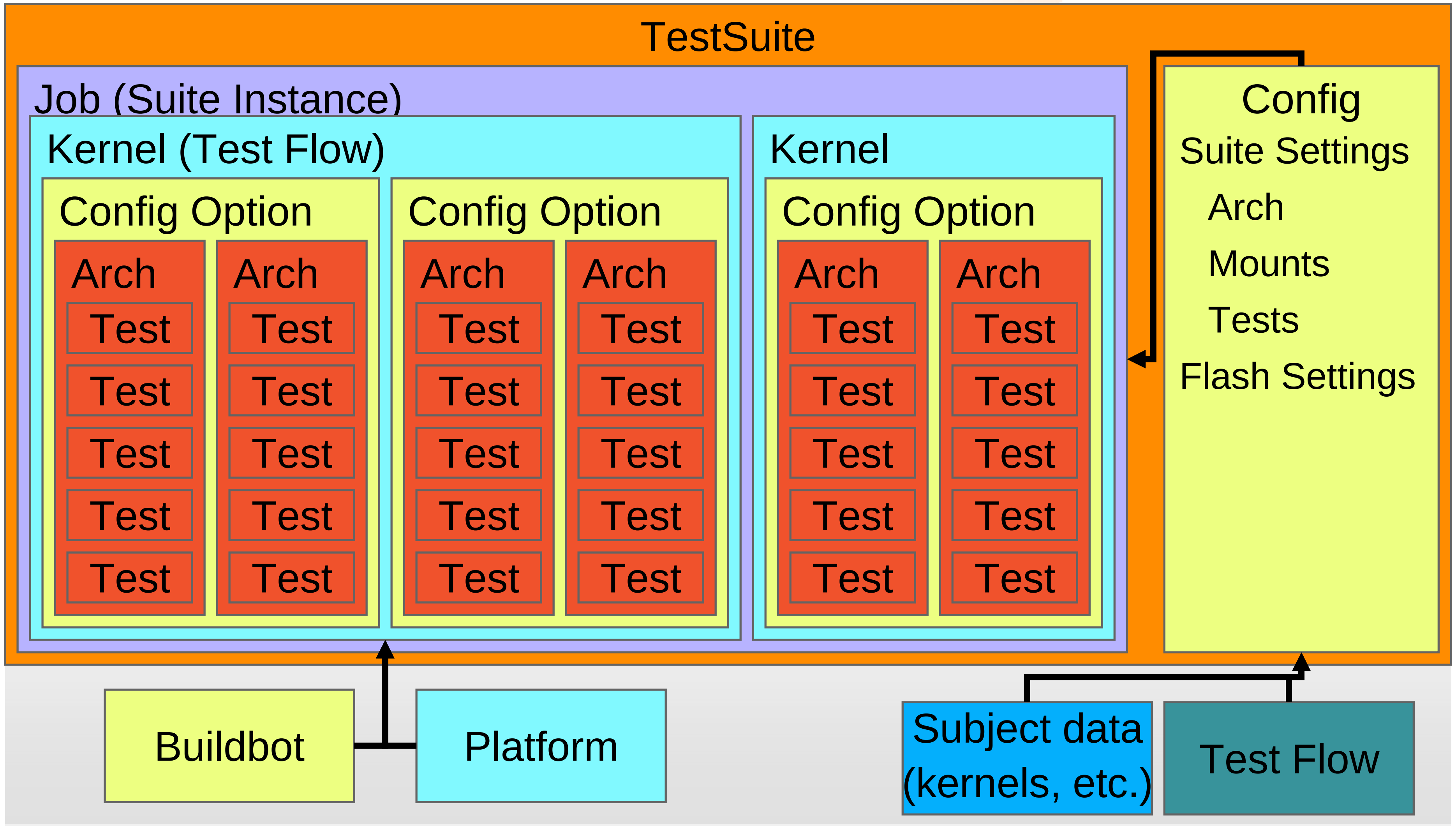


Technology

- Ruby + DRb (Distributed Ruby)
- WEBrick
- open4 library

Solving Code Redundancy

Test Suite Concept



Controlling Kernel Build

Kernel Config Options

```
axfs_mtd = Hash.new
axfs_mtd["description"] = "AXFS MTD only Build"
axfs_mtd["CONFIG_AXFS"] = "y"
axfs_mtd["CONFIG_AXFS_PROFILING"] = "y"
axfs_mtd["CONFIG_MTD"] = "y"
axfs_mtd["CONFIG_BLOCK"] = "n"
```

You can toggle any kernel config option

Exec Test Flow

Performs basic execution and response evaluation

```
For each command in command_list do
 result := do_command_and_wait(command, timeout)
 if result contains failure_condition
      test_fails()
 elsif result contains pass_condition
      test_pass()
 unknown or timeout
      test_fails()
 end
```

Exec Test Settings

Contained in config.rb

```
options["exec"]["um"]["commands"] = ["busybox"]
options["exec"]["um"]["busybox"] = Hash.new
options["exec"]["um"]["busybox"]["cmd"] =
   "/axfs/bin/busybox"
options["exec"]["um"]["busybox"]["pass"] =
    [ /busybox\ \[function\]\ \[arguments\]/ ]
options["exec"]["um"]["busybox"]["fail"] =
    [ /Segmentation\ fault/, /Killed/, /[O|o]ps/ ]
options["exec"]["um"]["busybox"]["timeout"] = 20
```

Test behavior is defined by the config file

Specifying Filesystems

Filesystem Image Requirements

```
 options["images"]["arm"]["snd"] = Hash.new
 options["images"]["arm"]["snd"]["cmd"] =
    “axfs_image_builder -i ./microfs/rootfs -o
exec_snd_arm.axfs"
 options["images"]["arm"]["snd"]["filename"] =
    "exec_snd_arm.axfs"
```

Tests are unconcerned with filesystem type, only with operation

Reducing Duplicated Effort

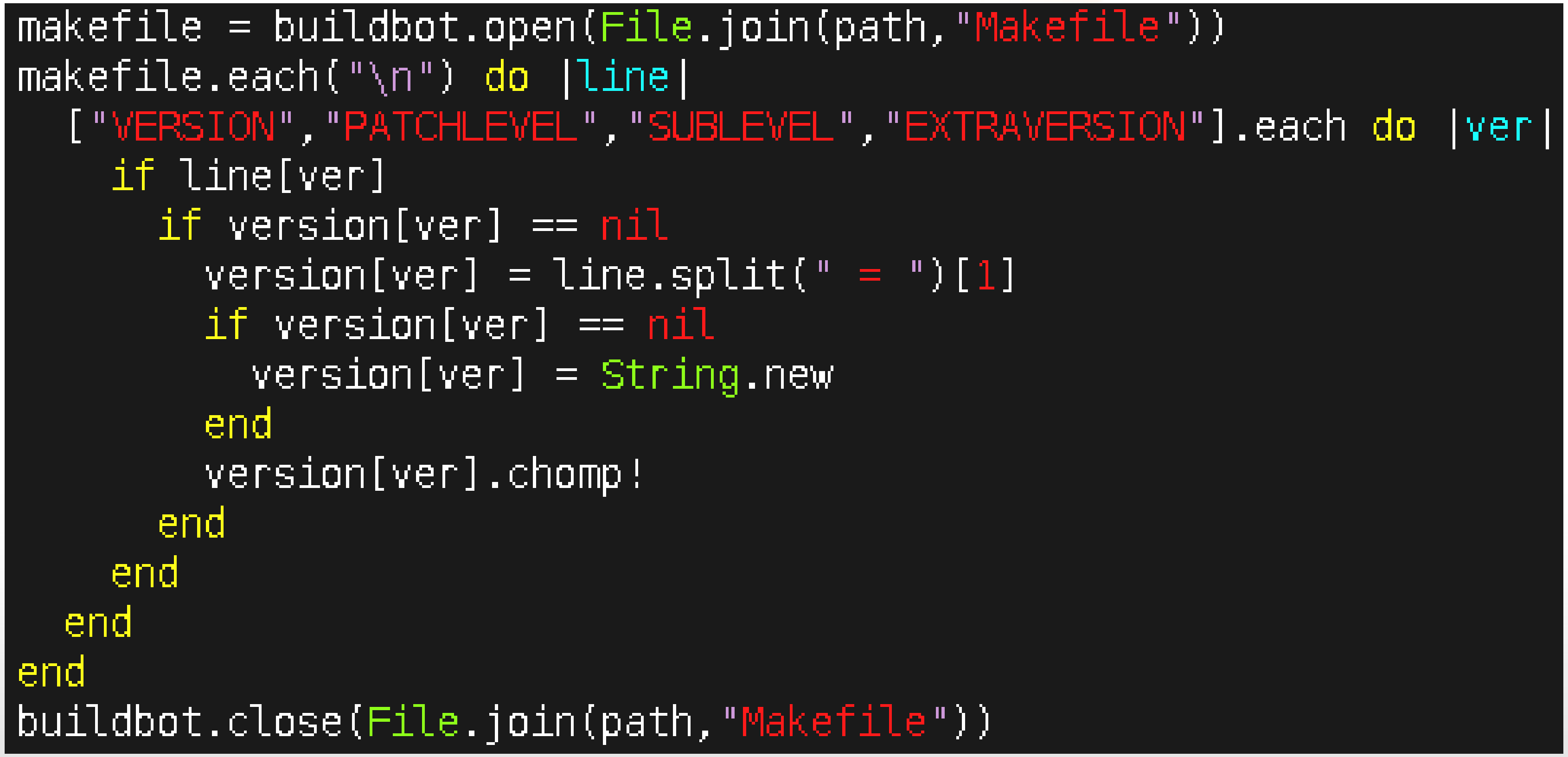
Buildbots

- Individual working directories
  - Cache-like effect on performance
- Handle all file interaction
  - Isolate tests from physical location of files
  - Protect tests from each other
  - Allow reuse of output (file system images, source repositories)
- Remote file interaction identical to local interaction
  - Distributed Ruby at work
- Allows creation of build clusters
  - Multiple build hosts to speed up tests

Buildbot API

- Command
  - *sys* – Execute a command in the buildbot
- File/Directory
  - *open, close, create, delete, copy, mkdir, link, ls*
- Evaluation
  - *is_dir?, exists?*
- System
  - *tftp_drop* – Move a file to the tftp directory for the associated platform.
  - *diff_files* – Perform a diff on two files visible to the Buildbot

Buildbot in Action



```
makefile = buildbot.open(File.join(path,"Makefile"))
makefile.each("\n") do |line|
  ["VERSION","PATCHLEVEL","SUBLEVEL","EXTRAVERSION"].each do |ver|
    if line[ver]
      if version[ver] == nil
        version[ver] = line.split(" = ")[1]
        if version[ver] == nil
          version[ver] = String.new
        end
        version[ver].chomp!
      end
    end
  end
end
buildbot.close(File.join(path,"Makefile"))
```

Helpers

- Leverage version control systems
  - GIT, SVN, Mercurial
- Can handle checkout, pulls, and update (no commits!)
- Influence subsequent operations (no change in file system repo -> no image rebuild)

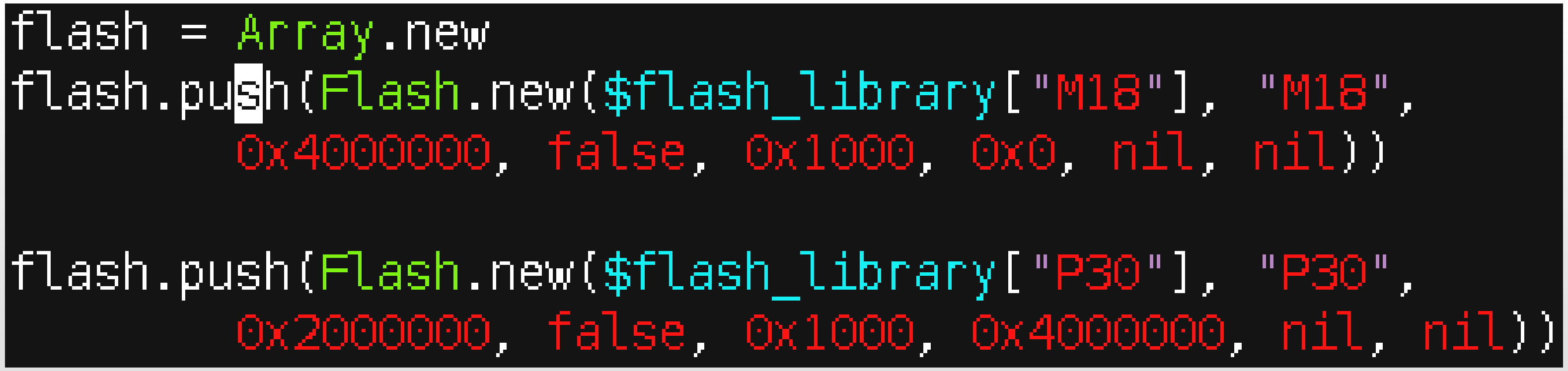
Managing Targets

Automatic MTD Partition Setup

Suite-level partition settings

```
parts = Array.new
parts.push(Partition.new("blob",0x80000))
parts.push(Partition.new("kern",0x200000))
parts.push(Partition.new("fs0",0x500000))
parts.push(Partition.new("fs1",0x500000))
parts.push(Partition.new("fs2",0x500000))
```

Target flash setup



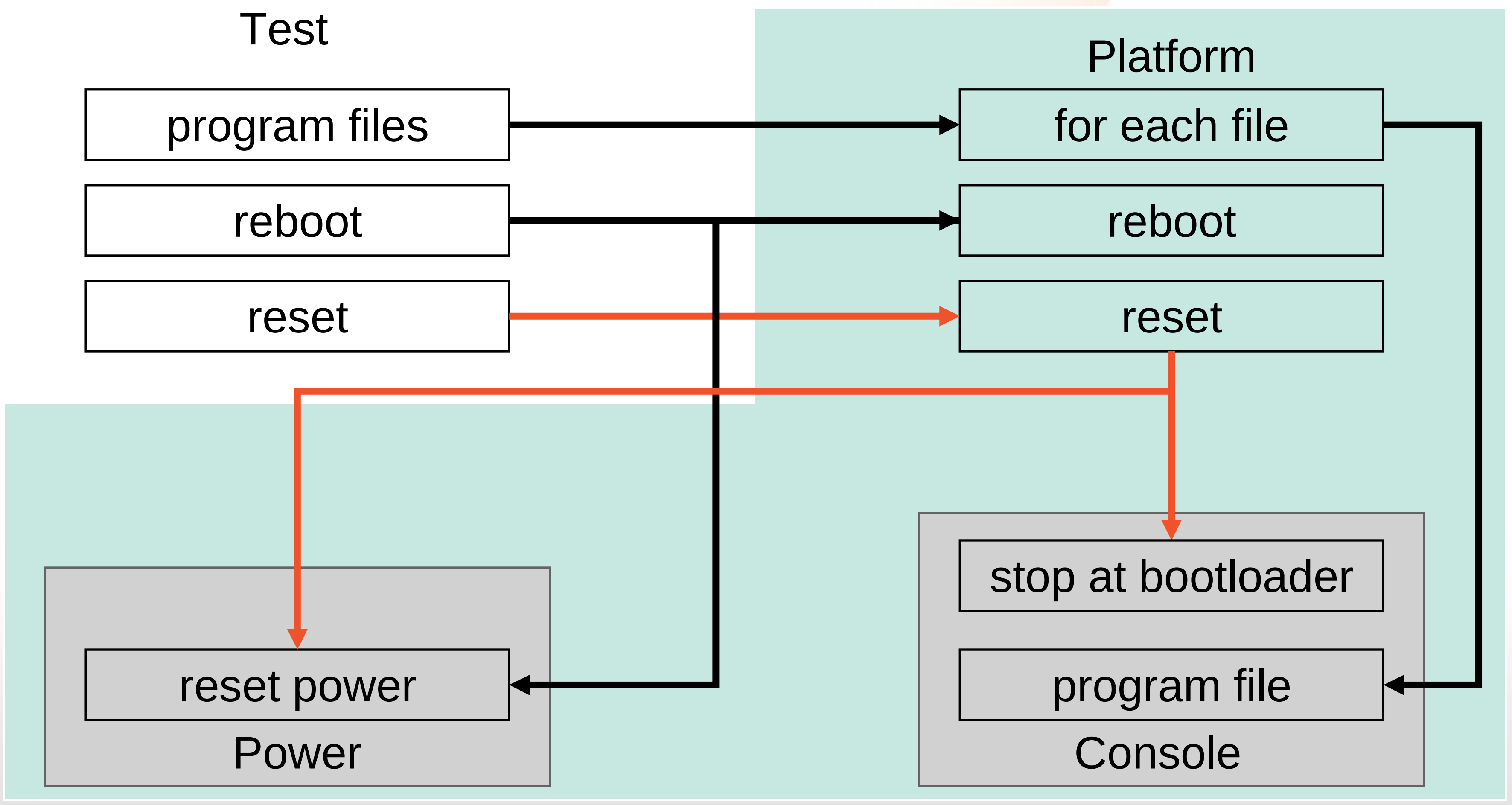
```
flash = Array.new
flash.push(Flash.new($flash_library["M18"], "M18",
        0x4000000, false, 0x1000, 0x0, nil, nil))

flash.push(Flash.new($flash_library["P30"], "P30",
        0x2000000, false, 0x1000, 0x4000000, nil, nil))
```

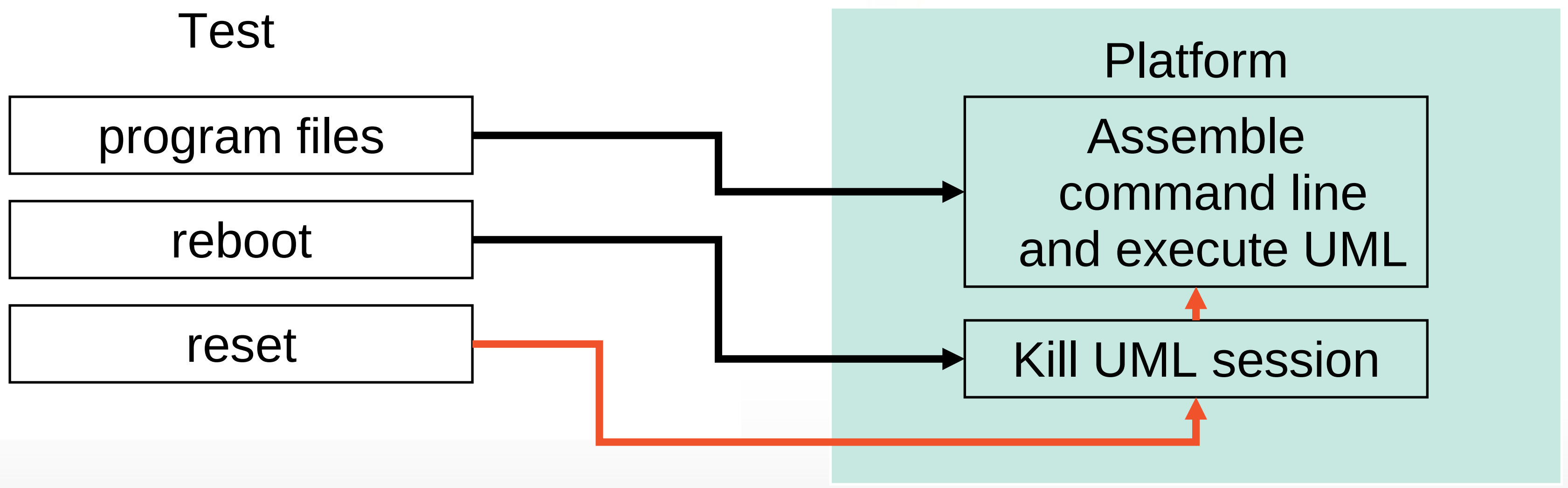
Platform API

- *print* – Write one line, no return of output
- *waitfor* – Wait for specific output, return output
- *waitcmd* – Execute a command and wait for specific output, return output
- *program_files* – Supply a list of files and wait until applied
- *reset* – Restart the system
- *reboot* – Restart the system and stop at the bootloader (if applicable)

ARM API Visualization



UML API Visualization



Prospective Enhancements

Prospective Future Improvements

- Convert to Ruby Gem
- Database Backing
- Convert core to Ruby-on-Rails?
- Actual name for this thing?
- Source code cleanup
- Test Suite -> Test Host
  - Better isolate tests from underlying system
  - Run tests on remote hosts
- Provide a real test API
  - Existing API is very basic
- Better interface
  - Shift resource management into Web Interface

Q&A