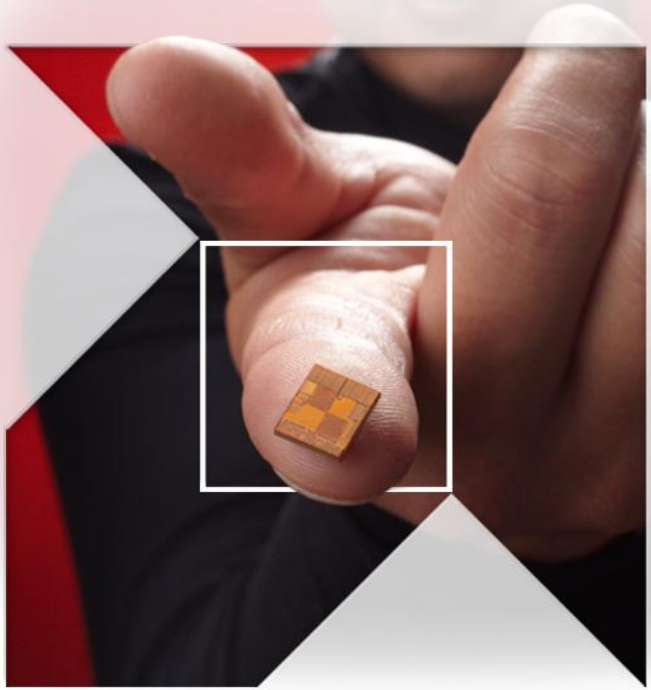


LINUX TESTING WITH TAPPER

Complexity in a nutshell

Steffen Schwigon, AMD Operating System Research Center
October 28, 2011
Public

OVERVIEW | Abstract

The Operating System Research Center (OSRC), a global AMD Research organisation headquartered in Dresden, Germany, acts as a bridge between the OS development community and the worldwide AMD processor design community.

At the OSRC we run a test infrastructure to test Linux in many orthogonal dimensions: hardware generations, software visible features, kernel branches, Linux-based distributions, virtualization with upstream or distro-specific Xen and KVM, multi-machine scenarios, and running in simulators. Inside of those dimensions we cover regression, functional, and stress tests, benchmarks, guest migration, and reboot and suspend/resume tests.

This talk will give an overview of our test infrastructure (codename "Tapper") and dive deeper into some interesting technical topics like the machine scheduler and the query interface, show the combination of open-source standard protocols and tools to glue everything together, and how we break down that complexity into easy but powerful, scriptable APIs with no client-side toolchain dependencies for the users.

OVERVIEW | Context

- Operating System Research Center (OSRC)
- We developed and run a test infrastructure called *Tapper*
  - Automated testing of operating systems and virtualization (Xen/KVM)
  - Published as open source in 2011
    - Overview: http://developer.amd.com/zones/opensource/AMDTapper
    - Mailing list: http://www.amd64.org/mailman/listinfo/tapper
    - Source: http://github.com/amd

OVERVIEW | Agenda

- Overview
  - Mission
  - Test approaches
  - Test infrastructure
  - Automation
  - Web GUI
- Testing
  - Understanding the test protocol
  - From visible simplicity to hidden complexity
- Automation
- Result evaluation
  - From hidden complexity back to visible simplicity

SCOPE | Mission

- OS distribution testing
  - Partner distributions: Novell (SLES), Red Hat (RHEL)
  - Community distributions: openSuse, Fedora, (Debian, Ubuntu)
  - Windows, as guest
- Linux kernel
  - OSRC contributions
  - Regressions (“Attack of the alien patches”)
- Virtualization
  - Xen
  - KVM
- AMD hardware

- → **Combinations** of all the above

TEST APPROACHES | 1/3: Functional testing

- Functional testing of Linux kernel
  - → **“Classic QA”**
  - For OSRC enablement work
  - Iterate Linux kernel + hardware + developer repositories

TEST APPROACHES | 2/3: Virtualization matrix

- Virtualization matrix
  - → **Find new problems**
  - Distro Xen/KVM vs. upstream releases
  - Huge matrix of host/guest combinations
  - Stress system (use benchmarks in guests)
  - Iterate Xen/KVM + hardware

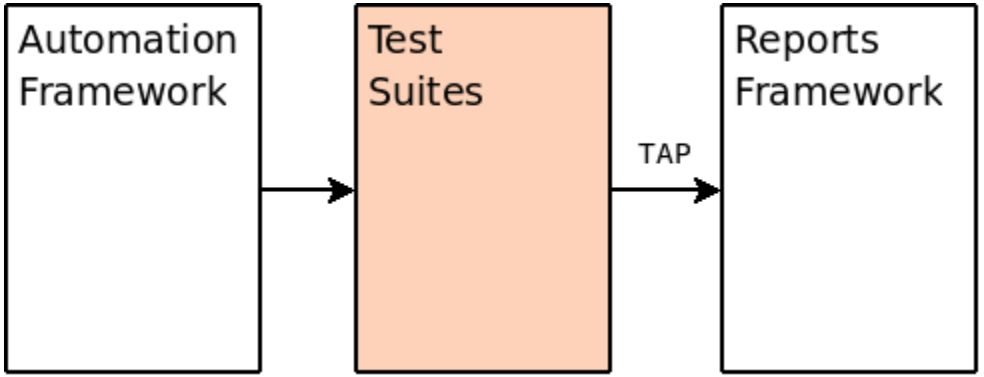
TEST APPROACHES | 3/3: Testplans

- Testplans
  - → **Ensure no regressions**
  - Dedicated scenarios for points of interest
  - Bridge to “TaskJuggler” planning software
    - Bi-directional: scheduling + reporting
  - On top of the other approaches

OVERVIEW | Test infrastructure

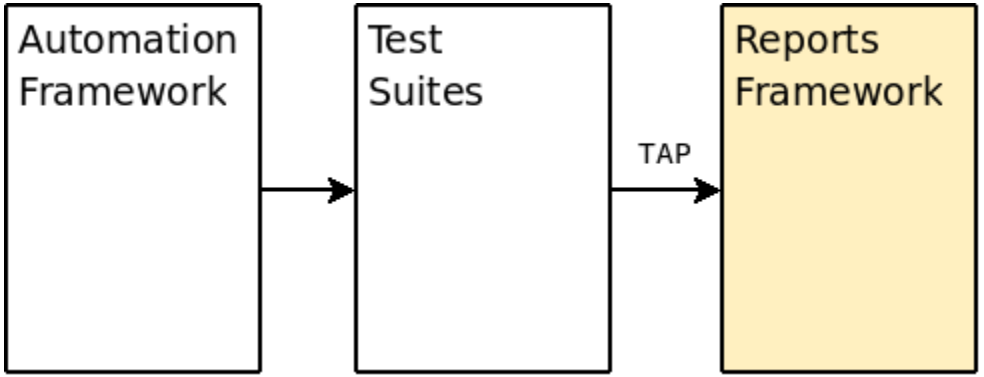
BASIC PRINCIPLES

- Test infrastructure "***Tapper***" -- basic principles:
  - **Zero overhead to write tests and report results**
  - Flexible evaluation: easy web GUI + scriptable API
  - Optional but advanced automation



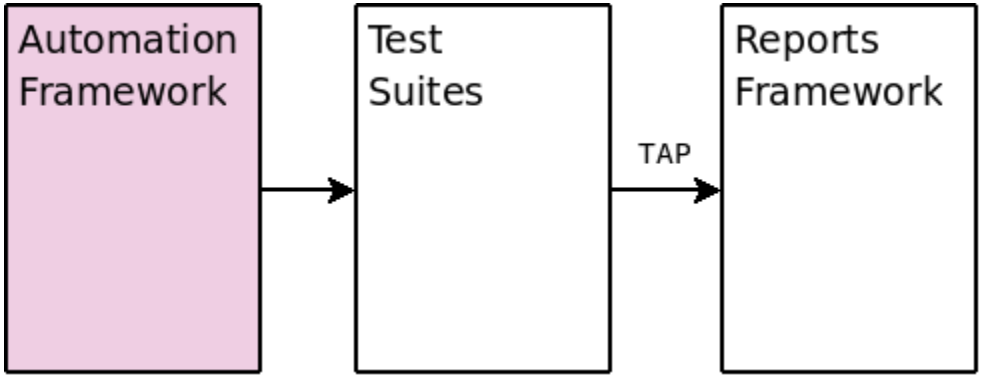
BASIC PRINCIPLES

- Test infrastructure "***Tapper***" -- basic principles:
  - Zero overhead to write tests and report results
  - **Flexible evaluation: easy web GUI + scriptable API**
  - Optional but advanced automation



BASIC PRINCIPLES

- Test infrastructure "***Tapper***" -- basic principles:
  - Zero overhead to write tests and report results
  - Flexible evaluation: easy web GUI + scriptable API
  - **Optional but advanced automation**



BASIC PRINCIPLES | Zero overhead

- Zero overhead to write tests
  - Just respect the test protocol (“TAP”)

```
1..3
ok - feature 'foo' available
ok - expected return value
not ok - memory cleaned up
```

- Zero overhead to report results
  - “Fire & forget” into socket

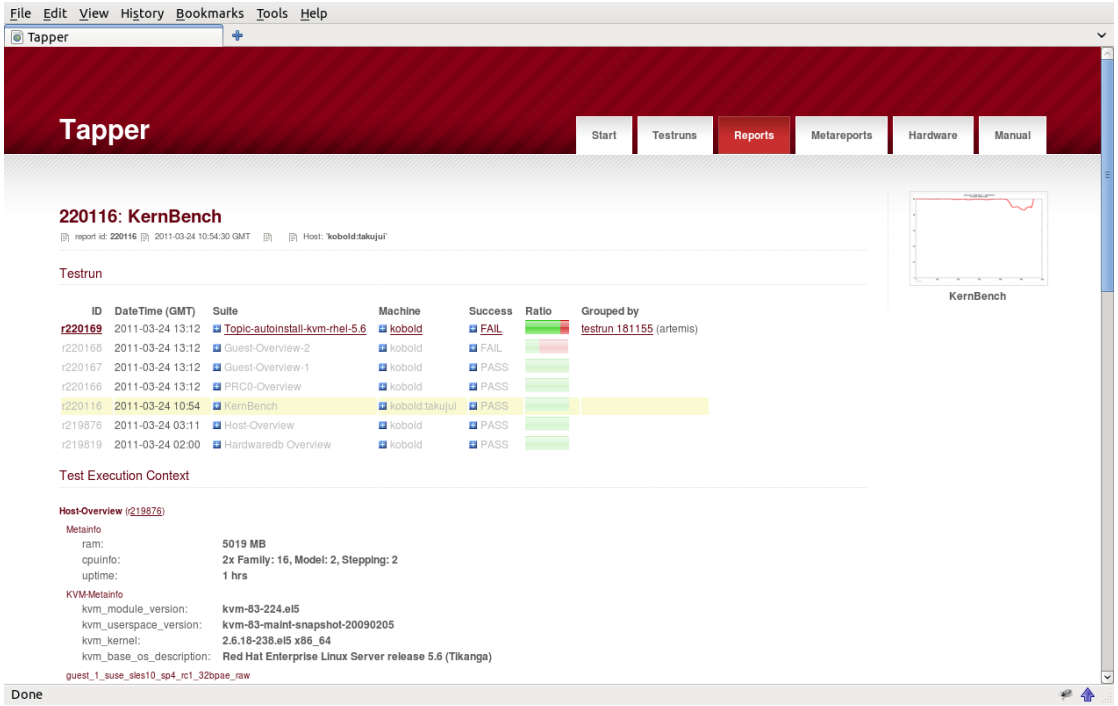
```
test_program.sh | netcat tapper 7357
```

BASIC PRINCIPLES | Advanced automation

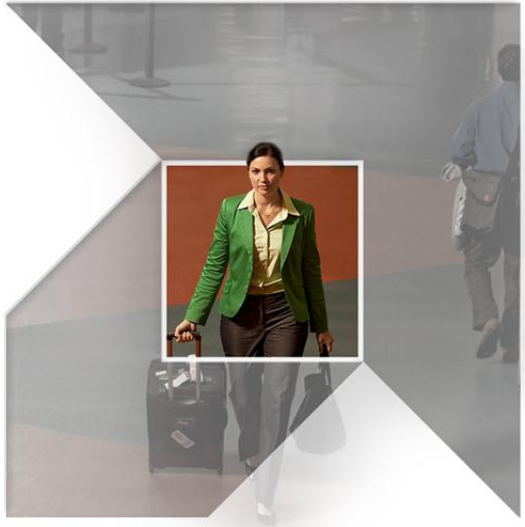
- Boot machines from network and set up from scratch
  - Optionally reuse machines via SSH
- Track serial console output
- Hardware reset on epic fails
- Virtualization setups
- Complex timeout handling
- Optimize machine utilisation
  - Bandwidth-driven multiplexing of “too many” use-cases on “not enough” machines
- Allow multi-machine scenarios
  - Network performance, guest migration
- Benchmarking infrastructure

BASIC PRINCIPLES | Evaluation

- Scriptable query interface
- Web interface



TESTING



The central idea is not a technology but a protocol.

The central idea is not a technology but a protocol.

(a standard one with already existing technology)

TAP INTRO | Philosophy

- No obligatory API how to **write test**
- But standard protocol to **declare test results**

- "Test Anything Protocol" (TAP)
  - Easy to generate
    - OS testing - be prepared to have nothing
    - Still easy when `printf/printk/echo` is everything you have
    - However, lots of toolchains optionally available
  - Scales from simplicity to complexity
  - http://testanything.org
  - http://amd64.org/fileadmin/user_upload/pub/yapc_eu_2011_tapjuggling.pdf

TAP INTRO | Basics

- Line-based
- Starts with a plan ("**1..3**") – how many test lines expected
- Some "`ok`" test lines
- Some "`not ok`" test lines
- Directives "`# TODO`" / "`# SKIP`" on test lines
- Comment lines starting with "`#`"
- Unrecognized lines are ignored

TAP INTRO | Synopsis

```
1..3
ok
ok
not ok
```

- **Plan and ok/not ok lines**

TAP INTRO | Synopsis

```
1..3
ok - established connection
ok - checksum
not ok - transfer completed
```

- Plan and ok/not ok lines
- **Test line descriptions**

TAP INTRO | Synopsis

```
1..3
ok - established connection
ok - checksum
not ok - transfer completed
# got error message "Bummer!"
```

- Plan and ok/not ok lines
- Test line descriptions
- **Comment lines**

TAP INTRO | Synopsis

```
1..3
ok - established connection
ok - checksum
not ok - transfer completed # TODO we know it fails
# got error message "Bummer!"
```

- Plan and ok/not ok lines
- Test line descriptions
- Comment lines
- Directives # **TODO**

TAP INTRO | Synopsis

```
1..3
ok - established connection
ok - checksum # SKIP no md5sum available
not ok - transfer completed # TODO we know it fails
# got error message "Bummer!"
```

- Plan and ok/not ok lines
- Test line descriptions
- Comment lines
- Directives # TODO / **# SKIP**

TAP INTRO | Synopsis

```
1..3
ok - established connection
ok - checksum # SKIP no md5sum available
not ok - transfer completed # TODO we know it fails
# got error message "Bummer!"
Hello? I am a statement lost in code, help me out!
```

- Plan and ok/not ok lines
- Test line descriptions
- Comment lines
- Directives # TODO / # SKIP
- **Unrecognized lines are ignored**

TAP INTRO | Synopsis

```
1..3
ok 1 - established connection
ok 2 - checksum # SKIP no md5sum available
not ok 3 - transfer completed # TODO we know it fails
# got error message "Bummer!"
Hello? I am a statement lost in code, help me out!
```

- Plan and ok/not ok lines – **optionally numbered**
- Test line descriptions
- Comment lines
- Directives # TODO / # SKIP
- Unrecognized lines are ignored

TAP INTRO | Embedded data

```
1..3
ok 1 - established connection
ok 2 - checksum # SKIP no md5sum available
not ok 3 - transfer completed # TODO we know it fails
# got error message "Bummer!"
Hello? I am a statement lost in code, help me out!
```

TAP INTRO | Embedded data

```
1..4
ok 1 - established connection
ok 2 - checksum # SKIP no md5sum available
not ok 3 - transfer completed # TODO we know it fails
# got error message "Bummer!"
Hello? I am a statement lost in code, help me out!
ok - transfer benchmarks
```

TAP INTRO | Embedded data in YAML

```
1..4
ok 1 - established connection
ok 2 - checksum # SKIP no md5sum available
not ok 3 - transfer completed # TODO we know it fails
# got error message "Bummer!"
Hello? I am a statement lost in code, help me out!
ok - transfer benchmarks
  ---
  benchmarks:
    pass1: 1234.56
    pass2: 999.99
  ...
```

TAP INTRO | Transport Tapper meta-information

TAP INTRO | Transport Tapper meta-information

```
1..4



ok 1 - established connection
ok 2 - checksum # SKIP no md5sum available
not ok 3 - transfer completed # TODO we know it fails
# got error message "Bummer!"
Hello? I am a statement lost in code, help me out!
ok - transfer benchmarks
  ---
  benchmarks:
    pass1: 1234.56
    pass2: 999.99
  ...
```

TAP INTRO | Transport Tapper meta-information

```
1..4
# Tapper-Suite-Name: hello-world

ok 1 - established connection
ok 2 - checksum # SKIP no md5sum available
not ok 3 - transfer completed # TODO we know it fails
# got error message "Bummer!"
Hello? I am a statement lost in code, help me out!
ok - transfer benchmarks
  ---
  benchmarks:
    pass1: 1234.56
    pass2: 999.99
  ...
```

TAP INTRO | Transport Tapper meta-information

```
1..4
# Tapper-Suite-Name: hello-world
# Tapper-Reportgroup-Testrun: 244122
ok 1 - established connection
ok 2 - checksum # SKIP no md5sum available
not ok 3 - transfer completed # TODO we know it fails
# got error message "Bummer!"
Hello? I am a statement lost in code, help me out!
ok - transfer benchmarks
  ---
  benchmarks:
    pass1: 1234.56
    pass2: 999.99
  ...
```

TAP INTRO | Run and evaluate

- Developer, locally

```
$ prove my_feature.sh                                  # run + evaluate
```

TAP INTRO | Run and evaluate

- Developer, locally

```
$ prove my_feature.sh                          # run + evaluate
my_feature.sh .. ok
All tests successful.
Files=1, Tests=1,  3 wallclock secs ( … )
Result: PASS
```

TAP INTRO | Run and evaluate

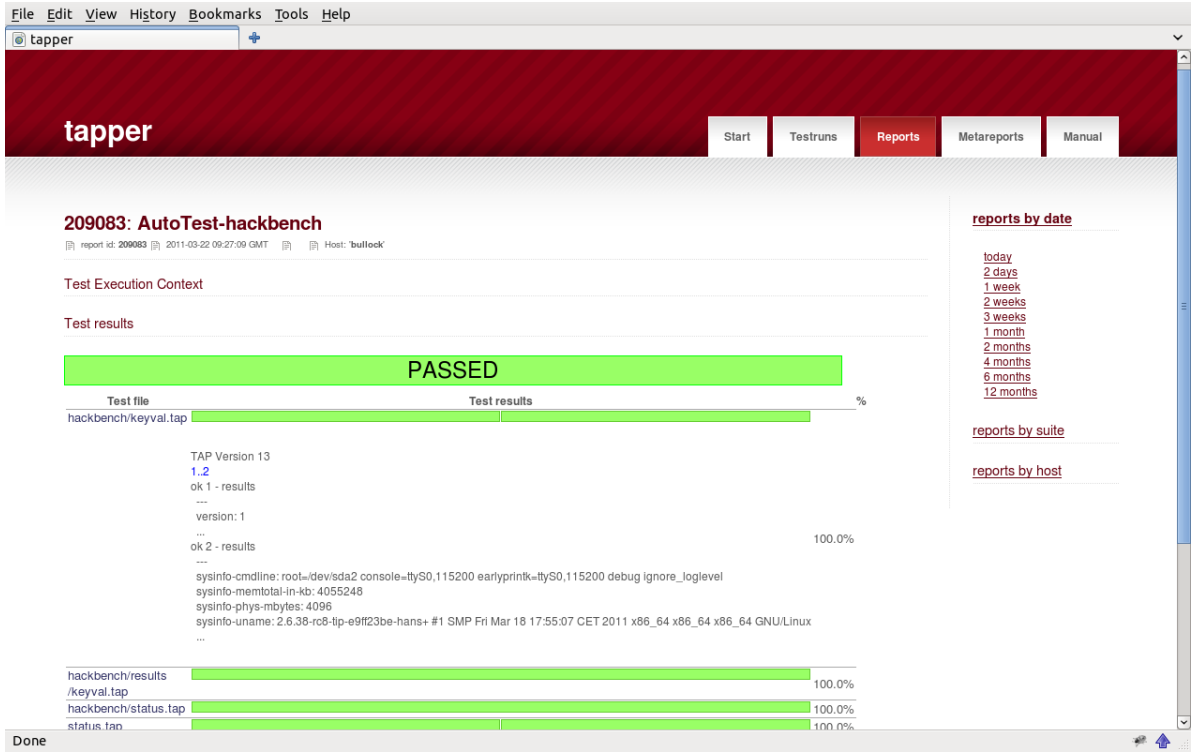
- Developer, locally

```
$ prove my_feature.sh                          # run + evaluate
my_feature.sh .. ok
All tests successful.
Files=1, Tests=1,  3 wallclock secs ( … )
Result: PASS
```

- Inside Tapper

```
$ prove -e cat static_tap_results.tap          # just evaluate
$ prove --formatter=TAP::Formatter::HTML …     # render to HTML
```

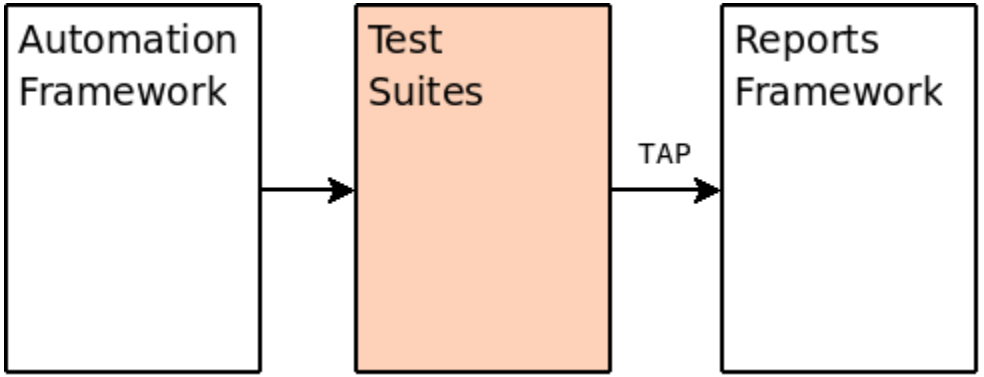
TAP INTRO | Rendering TAP



Everyone who can do TAP can participate.

Now put techology around it.

TESTING



TAP SUPPORT | Re-using and writing tests

- Using tests: **autotest**
- Writing tests: **Tapper-autoreport**

TAP SUPPORT | Using tests - autotest

- **Autotest** is a test project targeting the Linux kernel
- Wraps lots of existing test and benchmark suites
- AMD contributed TAP support as of autotest v0.13
  - Convert test results and data to TAP + embedded YAML
  - Bundle everything in TAP::Archive (.tar.gz of TAP + meta data)

```
autotest --tap tests/hackbench/control
```

- Generic wrapper Tapper::TestSuite::AutoTest
  - Downloads + installs autotest client from github or other URL
  - Run and upload TAP to Tapper server

```
tapper-testsuite-autotest --test hackbench
```

  - https://github.com/amd/Tapper-TestSuite-AutoTest

TAP SUPPORT | Writing tests - tapper-autoreport (2)

- **Tapper-autoreport**
- A shell “include” (`source`) file
  - You do what you normally do to test from shell
- Will make your script magically behave like a Tapper testsuite
  - Sends a TAP report to Tapper server
  - Includes meta-information
  - Uploads files
  - No send+upload when run via “`prove`” for local test development
- Lots of ways to influence behaviour
  - “Do What I Mean” parameters
  - Environment variables
  - Uses Tapper automation environment (e.g., Testrun-ID)
- https://github.com/amd/Tapper-autoreport

TAP SUPPORT | Writing tests - tapper-autoreport (3)

- SYNOPSIS – shortest usage

```
#! /bin/bash

# your testing here
. tapper-autoreport $? /tmp/my.log /tmp/results.dat
```

- “Do What I Mean” params
  - $? … integers are interpreted as success/fails – useful for one-liners
  - Existing filenames are attachments to be uploaded

TAP SUPPORT | Writing tests - tapper-autoreport (3)

- SYNOPSIS – shortest usage

```
#! /bin/bash

# your testing here
. tapper-autoreport $? /tmp/my.log /tmp/results.dat
```

- "Do What I Mean" params
  - **$? … integers are interpreted as success/fails – useful for one-liners**
  - Existing filenames are attachments to be uploaded

TAP SUPPORT | Writing tests - tapper-autoreport (3)

- SYNOPSIS – shortest usage

```
#! /bin/bash

# your testing here
. tapper-autoreport $? /tmp/my.log /tmp/results.dat
```

- "Do What I Mean" params
  - $? … integers are interpreted as success/fails – useful for one-liners
  - **Existing filenames are attachments to be uploaded**

TAP SUPPORT | Writing tests - tapper-autoreport (4)

- SYNOPSIS – use utility functions

```
#! /bin/bash
. tapper-autoreport --import-utils
# your testing here
. tapper-autoreport $? /tmp/my.log /tmp/results.dat
```

- Utility functions like
  - `ok $? "some description"`
  - `negate_ok $? "some description"`
  - `require_cpu_feature "cpb"`
  - `require_family_range 0x12 0x15`
  - `has_kernel_config CONFIG_SENSORS_FAM15H_POWER`

TAP SUPPORT | tapper-autoreport example

- “Vendor ID” – the issue
  - Some data structure overflowed into the vendor ID in `/proc/cpuinfo`
  - Sloppily check we are on AMD and skip all if not
  - Check whether the full vendor string is correct

TAP SUPPORT | tapper-autoreport example

- “Vendor ID” – the test

```
#! /bin/bash  # vendor-id.sh

. tapper-autoreport --import-utils

TICKETURL='https://osrc/bugs/show_bug.cgi?id=901'

require_vendor_amd
grep -q 'vendor.*AuthenticAMD' /proc/cpuinfo
ok $? ”vendor string in /proc/cpuinfo”

. tapper-autoreport
```

TAP SUPPORT | tapper-autoreport example

- “Vendor ID” – the report

```
1..6
# Tapper-suite-name:                vendor-id
# Tapper-machine-name:              bascha
# Tapper-ticket-url:                https://osrc/bugs/show_bug.cgi?id=901
# Tapper-uname:                     Linux bascha 2.6.35 #59-Ubuntu SMP Tue Aug 30 19:00:03 UTC 2011 x86_64 GNU/Linux
# Tapper-osname:                    Ubuntu 10.10
# Tapper-kernel:                    2.6.35
# Tapper-changeset:                 Linux version 2.6.35-30-generic (buildd@allspice) (gcc version 4.4.5 (Ubuntu…
# Tapper-flags:                     root=UUID=6990cb5e-1a77-40b8-ba05-919f6c928607 ro quiet splash
# Tapper-cpuinfo:                   2 cores [AMD Athlon(tm) 64 X2 Dual Core Processor 6000+]
# Tapper-ram:                       2007
# Tapper-starttime-test-program:    Tue, 11 Oct 2011 14:48:04 +0200
ok - autoreport
ok - exitcode
  ---
  exitcode: 0
  ...
ok - success
ok - require_vendor_amd
ok - vendor string in /proc/cpuinfo
# File upload: '/boot/config-2.6.35-30-generic'
# File upload: 'vendor-id.sh'
# File upload: '/proc/cpuinfo'
# File upload: '/proc/devices'
# File upload: '/proc/version'
```

Zero overhead to submit test results.

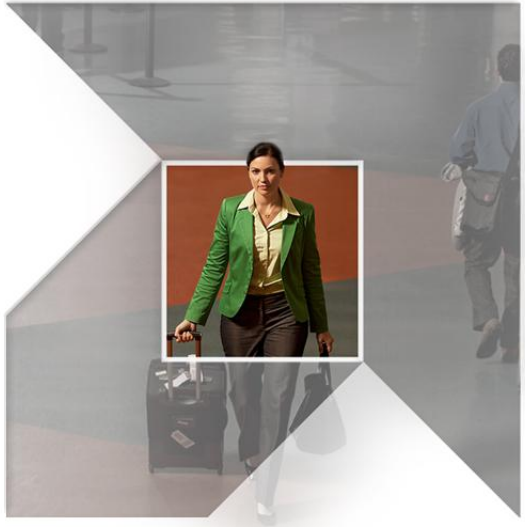
REPORT RESULTS | “Fire & forget”

- Report via “fire & forget” into socket

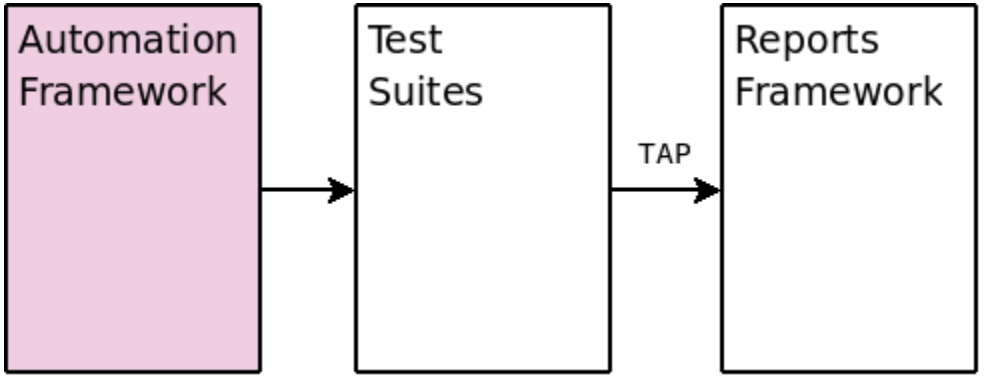
```
$ test_program.sh | netcat tapper 7357
```

  - Easy
  - On crashes you get as much as possible
  - Still recognize the crash (planned vs. counted test lines)

AUTOMATION



AUTOMATION



AUTOMATION | Overview (1)

- Optional
- Set up machines from scratch, over network
  - Unpack prepared images (.iso, .tgz)
  - Or run kickstart/autoyast/d-i distro installers
  - Inject any other requirements
- Allow virtualization setups
  - Xen, KVM
  - Inject into guests
- Optionall via SSH
  - Already prepared machines
  - E.g., simnow, just inject kernel

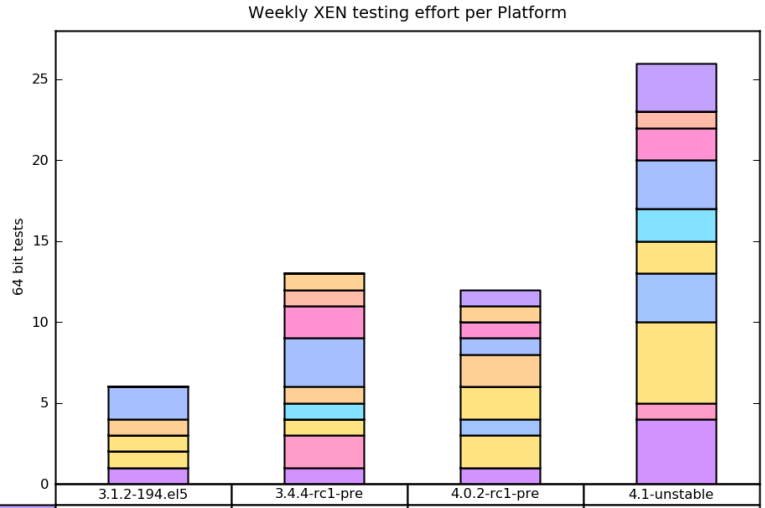
AUTOMATION | Overview (2)

- Track serial console output, cover early boot problems
- Hardware reset on epic fails
- Time-out handling
  - Virtualization-aware
- Suspend/Resume support
- Benchmarking infrastructure

AUTOMATION | Overview (3)

- Advanced scheduling
  - Optimize machine utilisation
    - “Too many” use-cases on “not enough” machines
    - Use-case queues with bandwidths/priorities
    - Different types of bandwidths (“official” vs. “non-official” to fill under-used machines)
    - Choose host by complex feature expressions
      - `mem >= 4096 and vendor eq “AMD”`
    - Bind hosts to queues
  - Allow multi-machine scenarios
    - Network performance
    - Guest migration

AUTOMATION | Use-case bandwidths



	3.1.2-194.el5	3.4.4-rc1-pre	4.0.2-rc1-pre	4.1-unstable
CHEETAH (DR-B2I)	0	0	1	3
CHEETAH (DR-B3N)	0	1	1	0
DINAR (HY-D0A)	0	1	0	1
INTEL-VT (UNKNOWN)	0	2	1	2
PC-WARE (BH-F2I)	2	3	1	3
PC-WARE (BH-G1B)	1	1	2	0
PC-WARE (BL-C2A)	0	1	0	2
PC-WARE (JH-F2G)	1	0	2	2
PENCE2 (HY-D1C)	0	0	1	3
TILAPIA (PH-E0A)	1	1	2	5
TOONIE (RB-C2G)	0	2	0	1
WARTHOG (JH-F2G)	1	1	1	4

AUTOMATION | Synopsis (1)

- SYNOPSIS

```
$ tapper-testrun newhost --name grizzly --active --queue simnow

$ tapper-testrun listhost --verbose

   ID |      Name |      Active | Testrun ID |                Comment | Queues
  ==================================================================================
   30 |     blibb |      active |     249876 |                        |
   31 |     blobb |      active |     249862 |                        | xen-unstable-pvops-64
   27 |     blubb |      active |     249529 | testplan experimenting | AdHoc
   22 |   grizzly |      active |       free |                        | simnow

$ tapper-testrun listqueue

      ID | Name                              | Bandwidth
      10 | AdHoc                             | 1000
      98 | autoinstall-bare-rhel-6.2-64      | 200
      71 | autoinstall-bare-sles-11.2-64     | 200
      21 | xen-4.0-testing-32                | 50
      22 | xen-4.0-testing-64                | 50
      75 | xen-4.1-testing-pvops-32          | 300
      76 | xen-4.1-testing-pvops-64          | 300
```

AUTOMATION | Synopsis (2)

- SYNOPSIS

```
$ tapper-testrun listqueue --name AdHoc

  Id: 10
  Name: AdHoc
  Priority: 1000
  Active: yes
  Bound hosts: blubb, affe, zomtec
  Queued testruns (ids): 238772, 238773, 238774, 238785, 238786, 238787


$ tapper-testrun list --id 249532

  id: 249532
  topic: track-workload-stress-opensuse_11.4_32
  state: schedule
  queue: AdHoc
  requested hosts: blubb
  auto rerun: no
  precondition_ids: 224057, 224058, 224059, 224060, 224061
```

AUTOMATION | Synopsis (3)

- SYNOPSIS

```
$ tapper-testrun freehost --name grizzly --desc "known Xen hang, don't wait for timeout"


$ tapper-testrun newtestplan -v \
                             --file topic/osrc/kernel/track-workload/track-workload_autoinstall \
                             -Ddistros=rhel_6.1_64,sles_11.2_32 \
                             -Dtests=hackbench,dbench
  Plan created
    id:   241
    url:  http://tapper/tapper/testplan/id/241
    path: topic/osrc/kernel/track-workload/track-workload_autoinstall
    file: /data/tapper/live/testplan/topic/osrc/kernel/track-workload/track-workload_autoinstall
```

AUTOMATION | Screenshots (1)

Testruns of last 2 days - Mozilla Firefox

Firefox | tapper | Testruns of last 2 days | Testruns of last 2 days

tapper

Start | Testruns | Reports | Testplans | Metareports | Manual

Automated testruns of last 2 days

This list shows automated **testruns**. Links in columns *ID* show details of single report; on column *Suite* show all reports of this suite; on column *Machine* show all reports on this machine. If you look for more finegrained results not limited to the automation system try Reports.

Fri Oct 14, 2011

ID	DateTime (GMT)	Topic	Machine	state	Ratio	Owner
tr250344	2011-10-14 15:02	Misc	No host assigned	schedule		tapper
tr250343	2011-10-14 14:54	track-workload-hackbench-rhel6.1_64	No host assigned	schedule		tapper
tr250342	2011-10-14 14:54	track-workload-stress-rhel6.1_64	No host assigned	schedule		tapper
tr250341	2011-10-14 14:54	track-workload-hackbench-sles11.2_64	No host assigned	schedule		tapper
tr250340	2011-10-14 12:54	track-workload-stress-sles11.2_64	athene	running		tapper
tr250339	2011-10-14 14:52	Misc	No host assigned	schedule		tapper
tr250338	2011-10-14 12:51	Debug	athene	finished		tapper
tr250337	2011-10-14 14:41	autoinstall-bare-rhel-6.2-32	No host assigned	schedule		farnold
tr250336	2011-10-14 14:38	autoinstall-bare-sles-11.2-64	No host assigned	schedule		root
tr250335	2011-10-14 14:04	autoinstall-bare-sles-11.2-32	No host assigned	schedule		root
tr250334	2011-10-14 14:03	autoinstall-bare-rhel-6.2-64	No host assigned	schedule		farnold
tr250333	2011-10-14 14:02	track-workload-hackbench-rhel6.1_64	No host assigned	schedule		tapper
tr250332	2011-10-14 14:02	track-workload-stress-rhel6.1_64	No host assigned	schedule		tapper
tr250331	2011-10-14 14:02	track-workload-hackbench-sles11.2_64	No host assigned	schedule		tapper
tr250330	2011-10-14 14:02	track-workload-stress-sles11.2_64	No host assigned	schedule		tapper
tr250329	2011-10-14 12:52	xen-unstable-pvops	gawaine	running		tapper
tr250328	2011-10-14 13:56	Misc	No host assigned	schedule		tapper
tr250327	2011-10-14 12:04	xen-unstable-pvops	king	finished		tapper
tr250326	2011-10-14 13:49	track-workload-hackbench-rhel6.1_64	No host assigned	schedule		tapper
tr250325	2011-10-14 13:49	track-workload-stress-rhel6.1_64	No host assigned	schedule		tapper
tr250324	2011-10-14 13:49	track-workload-hackbench-sles11.2_64	No host assigned	schedule		tapper
tr250323	2011-10-14 13:49	track-workload-stress-sles11.2_64	No host assigned	schedule		tapper
tr250322	2011-10-14 11:58	xen-unstable-pvops	king	finished		tapper
tr250321	2011-10-14 13:41	Misc	No host assigned	schedule		root
tr250320	2011-10-14 13:34	Misc	No host assigned	schedule		root
tr250319	2011-10-14 11:51	xen-unstable-pvops	king	finished		tapper
tr250318	2011-10-14 13:18	Misc	No host assigned	schedule		root

Testruns by date

today
1 week
2 weeks
3 weeks
1 month
2 months

Control

Create new Testrun

Active Filters

days: 2

AUTOMATION | Control

Here be dragons.

AUTOMATION CONTROL | Overview

- “Preconditions”
  - YAML files describing the machine setup
  - Automatically producible
  - Human-readable
  - Verifiable
  - Tweakable
    - “Do The Right Thing” internally, like always handle root image first
- We auto-generate them from a database
  - Test matrix of host/guest/workload/config combinations

AUTOMATION CONTROL | Synopsis

```
precondition_type: virt
name: automatically generated KVM test
host:
  root:
    precondition_type: autoinstall
    grub_text: "timeout 2\n\ntitle RedHat Testing\nkernel […] ks=[…] $TAPPER_OPTIONS […]"
    name: autoinstall-kvm-fedora-14
    timeout: 10000
  preconditions:
    - precondition_type: package
      filename: xen-pvops-d7e0e9f3.tar.gz
  testprogram_list:
    - execname: /opt/tapper/bin/tapper-testsuite-ctcs
      timeout_testprogram: 300
guests:
  - root:
      precondition_type: image
      …
  - root:
      precondition_type: image
      …
```

AUTOMATION CONTROL | Preconditions

- Different types
  - “Copy a file”, “unpack a package”
  - “Base OS from image file”, “base OS from kickstart/autoyast”
  - “Virtualization environment”
  - “Execute test program”
  - Lazy preconditions (“producer” plugins)
    - “Use latest Xen package file by the time of scheduling”
    - Combine that with “auto-rerun”
- Different granularity for different needs
  - Single preconditions
  - Macro-preconditions (+ precompile pass with template language)
  - Testplans (+ multiple machines, several testruns)

AUTOMATION CONTROL | Testplan philosophy

- Testplan developer vs. end user
- Internal power vs. external easiness
- The testplan developer combines **complexity** to provide **simplicity** to the end user
  - One single front-end file for the use-case
  - Optional parameters
  - Sensible defaults
  - Self-documentation

```
$ tapper-testrun newtestplan --verbose \
                             --file track-workload_autoinstall \
                             -Ddistros=rhel_6.1_64,sles_11.2_32 \
                             -Dtests=hackbench,dbench
```

AUTOMATION CONTROL | Testplan example

- The use-case:
  - Schedule matrix of workloads over distros and machines
  - Track perf counters to investigate the workload
  - Upload perf logs
  - Organize results to ease later evaluation

AUTOMATION CONTROL | Testplan 1/8 – prepare params

```
[%#- -*- mode: tt -*- %]
# tapper-description:      Track performance counters over a workload
# tapper-mandatory-fields:
# tapper-optional-fields:  tests, distros, machine
[%- PROCESS 'osrc/includes' -%]
[%- IF tests   == '' %][% tests   = 'hackbench'    %][% END -%]
[%- IF distros == '' %][% distros = 'sles_11.2_64' %][% END -%]
[%- IF machine == '' %][% machine = 'grizzly'      %][% END -%]
[%- IF title   == '' %][% title   = BLOCK %]\
                        [% IF (tests.match(',')) %]MULTI\
                        [% ELSE %][% tests %]\
                        [% END %]\
                        -\
                        [% IF (distros.match(',')) %]MULTI\
                        [% ELSE %][% distros %]\
                        [% END %][% END %][% END -%]
[% AllTests   = tests.split(',') %]
[% AllDistros = distros.split(',') %]
```

AUTOMATION CONTROL | Testplan 2/8 – self-documentation

```
### Track performance counters over several workloads.
###
### Name : track-workload-[% title %]
###
### Optional params:
###
###  -Dtests=<testname>  Workload names, comma separated; default: hackbench
###  -Ddistros=<distro>  Distro names, comma separated;   default: sles_11.2_64
###  -Dmachine=<machine> Machine name;                    default: grizzly
###
### Available values:
###
###   distros: [% FOREACH d = distro_list -%][% d %], [% END %]
###   tests:   [% FOREACH t = useful_autotest_tests -%][% t %], [% END %]
```

AUTOMATION CONTROL | Testplan 3/8 – open loops (distros + tests)

```
[%- FOREACH distro = AllDistros %]
[%- FOREACH test = AllTests %]
[% testrunsuffix = BLOCK %][% test %]-[% distro %][% END %]
[% Timeout = Timeout+10800 -%]
```

AUTOMATION CONTROL | Testplan 4/8 – prepare distro details

```
[% IF distro == 'sles_11.2_64' %]
 [% install_file = 'autoyast=http://tapper/autoinstall/sles/11.2/x86_64/bare.xml' %]
 [% install_repo = 'install=ftp://osko/testing/sles/11.2/x86_64' %]
 [% install_opts = 'textmode=1' %]
 [% kernel  = '/tftpboot/testing/sles/11.2/x86_64/linux' %]
 [% initrd  = '/tftpboot/testing/sles/11.2/x86_64/initrd' %]
[% END %]

[% IF distro == 'rhel_6.1_64' %]
 [% install_file = 'ks=http://tapper/autoinstall/rhel/6.1/x86_64/bare.ks' %]
 [% install_repo = 'repo=ftp://osko/rhel/6.1/x86_64/os' %]
 [% install_opts = 'ksdevice=link' %]
 [% kernel  = '/tftpboot/stable/rhel/6.1/x86_64/vmlinuz' %]
 [% initrd  = '/tftpboot/stable/rhel/6.1/x86_64/initrd.img' %]
[% END %]
```

AUTOMATION CONTROL | Testplan 5/8 – the actual spec begins

```
---
type: multitest
description:
  topic: track-workload-[% testrunsuffix %]
  requested_hosts_all:
  - [% machine %]
  preconditions:
```

AUTOMATION CONTROL | Testplan 6/8 – setup OS and copy test files

```
-
  precondition_type: autoinstall
  name: autoinstall-[% distro %]
  grub_text: "timeout 2\n\n\
              title [% distro %] Testing\n\
              kernel [% kernel %] \
                     [% install_file %] \
                     [% install_repo %] \
                     [% install_opts %] \
                     console=ttyS0,115200 $TAPPER_OPTIONS\n\
              initrd [% initrd %]\n"
-
  precondition_type: copyfile
  protocol: local
  name: /data/tapper/testprograms/track-workload/*
  dest: /
```

AUTOMATION CONTROL | Testplan 7/8 – test programs

```
-
  precondition_type: testprogram
  program: /track-workload-pmc.sh
-
  precondition_type: testprogram
  program: /opt/tapper/bin/tapper-testsuite-autotest
  parameters:
    - --source_url
    - file:///data/tapper/packages/autotest/osrc-autotest-snapshot.tar.gz
    - --test
    - [% test %]
  timeout: [% Timeout %]
-
  precondition_type: testprogram
  program: /track-workload-upload-results.sh
```

AUTOMATION CONTROL | Testplan 8/8 – close loops

```
[% END %][%# FOREACH AllTests %]
[% END %][%# FOREACH AllDistros %]
```

AUTOMATION CONTROL | Testplan self-documentation

```
$ tapper-testrun newtestplan --guide --file track-workload_autoinstall
```

AUTOMATION CONTROL | Testplan self-documentation

```
$ tapper-testrun newtestplan --guide --file track-workload_autoinstall

  Track performance counters over several workloads.

  Name : track-workload-hackbench-sles_11.2_64

  Optional params:

   -Dtests=<testname>  Workload names, comma separated; default: hackbench
   -Ddistros=<distro>  Distro names, comma separated;   default: sles_11.2_64
   -Dmachine=<machine> Machine name;                    default: grizzly

  Available values:

    distros: rhel_6.1_64, sles_11.2_64, …
    tests:   hackbench, dbench, tiobench, …
```

AUTOMATION CONTROL | Testplan execution

```
$ tapper-testrun newtestplan --verbose \
                             --file track-workload_autoinstall \
                             -Ddistros=rhel_6.1_64,sles_11.2_32 \
                             -Dtests=hackbench,dbench
```

AUTOMATION CONTROL | Testplan execution

```
$ tapper-testrun newtestplan --verbose \
                             --file track-workload_autoinstall \
                             -Ddistros=rhel_6.1_64,sles_11.2_32 \
                             -Dtests=hackbench,dbench

  Plan created
     id:   241
     url:  http://tapper/tapper/testplan/id/241
     path: topic/osrc/kernel/track-workload/track-workload_autoinstall
     file: /data/tapper/[…]topic/osrc/kernel/[…]/track-workload_autoinstall
```

AUTOMATION CONTROL | Screenshots (1)

Testplan list - Mozilla Firefox

Firefox | Testplan list | Testplan id 244, t... | Testplan id 241, t... | Testrun id 24988... | Report ID 309790... | Report ID 309790... | Report ID 309790...

tapper

Start | Testruns | Reports | Testplans | Metareports | Manual

Fri Oct 14, 2011

ID	Name	Path	Success	Testruns (success/pending/fail)
tp244	track-workload-MULTI-MULTI	topic/osrc/kernel/track-workload/track-workload_autoinstall		(0/4/0)

Wed Oct 12, 2011

ID	Name	Path	Success	Testruns (success/pending/fail)
tp241	track-workload-stress-rhel_6.1_64	topic/osrc/kernel/track-workload/track-workload_autoinstall		(1/0/0)

Matrix Overview

Testplan by date

today
2 days
1 week
2 weeks
3 weeks
1 month
2 months
4 months
6 months
12 months

Active Filters

days: 7

Copyright © 2008-2011 AMD Operating System Research Center.

AUTOMATION CONTROL | Screenshots (2)

Testplan id 244, track-workload-MULTI-MULTI - Mozilla Firefox

Firefox | Testplan list | Testplan id 244, t... | Testplan id 241, t... | Testrun id 24988... | Report ID 309790... | Report ID 309790... | Report ID 309790...

tapper

Start | Testruns | Reports | Testplans | Metareports | Manual

Testplan 244: track-workload-MULTI-MULTI

Testruns

ID	DateTime (GMT)	Topic	Machine	state	Ratio	Owner
tr250340	2011-10-14 12:54	track-workload-stress-sles11.2_64	athene	running		tapper
tr250341	2011-10-14 14:54	track-workload-hackbench-sles11.2_64	No host assigned	schedule		tapper
tr250342	2011-10-14 14:54	track-workload-stress-rhel6.1_64	No host assigned	schedule		tapper
tr250343	2011-10-14 14:54	track-workload-hackbench-rhel6.1_64	No host assigned	schedule		tapper

Testplan specification

Path: topic/osrc/kernel/track-workload/track-workload_autoinstall

- Element: track-workload-stress-sles11.2_64

Kernel
Root image
Test track-workload-pmc.sh
tapper-testsuite-autotest
track-workload-upload-results.sh

- Element: track-workload-hackbench-sles11.2_64

Kernel
Root image
Test track-workload-pmc.sh
tapper-testsuite-autotest
track-workload-upload-results.sh

- Element: track-workload-stress-rhel6.1_64

Kernel
Root image
Test track-workload-pmc.sh
tapper-testsuite-autotest
track-workload-upload-results.sh

Matrix Overview

Testplan by date

today
2 days
1 week
2 weeks
3 weeks
1 month
2 months
4 months
6 months
12 months

Active Filters

244:

AUTOMATION CONTROL | Screenshots (5)

Report ID 309790, Topic-track-workload-stress-rhel_6.1_64 - Mozilla Firefox

Firefox | Testplan list | Testplan id 244, t... | Testplan id 241, t... | Testrun id 24988... | Report ID 309790... | Report ID 309790... | Report ID 309790...

tapper

Start | Testruns | Reports | Testplans | Metareports | Manual

Report 309790: Topic-track-workload-stress-rhel_6.1_64

report id: 309790 | 2011-10-11 18:17:09 GMT | Host: 'arges'

Reports

ID	DateTime (GMT)	Suite	Machine	Success	Ratio	Grouped by	Owner
r309790	2011-10-11 18:17	Topic-track-workload-stress-rhel_6.1_64	arges	PASS		testrun 249886	tapper
r309791	2011-10-11 18:17	PRC0-Overview	arges	PASS			
r309789	2011-10-11 18:17	track-workload-results-stress-rhel_6.1_64	arges	PASS			
r309788	2011-10-11 18:16	Autotest-stress	arges	PASS			
r309787	2011-10-11 18:15	track-workload-pmc	arges	PASS			
r309781	2011-10-11 18:05	Hardwaredb Overview	arges	PASS			

reports by date

today
2 days
1 week
2 weeks
3 weeks
1 month
2 months
4 months
6 months
12 months

reports by suite

reports by host

Test Execution Context

track-workload-pmc (r309787)

section-000

ram: 48396

cpuinfo: 24 cores [AMD Engineering Sample]

uname: Linux arges 2.6.32-131.0.15.el6.x86_64 #1 SMP Tue May 10 15:42:40 EDT 2011 x86_64 x86_64 x86_64 GNU/Linux

osname: Red Hat Enterprise Linux Server release 6.1 (Santiago)

flags: ro root=UUID=bfa3526e-0b4d-4306-b6cf-217d063c2158 rd_NO_LUKS rd_NO_LVM rd_NO_MD rd_NO_DM LANG=en_US.UTF-8 SYSFONT=latarcyrheb-sun16 KEYBOARDTYPE=pc KEYTABLE=us crashkernel=129M@0M tapper_host=plutonium tapper_port=1337 console=tty0 console=ttyS0,115200

kernel: 2.6.32-131.0.15.el6.x86_64

changeset: Linux version 2.6.32-131.0.15.el6.x86_64 (mockbuild@x86-007.build.bos.redhat.com) (gcc version 4.4.4 20100726 (Red Hat 4.4.4-13) (GCC)) #1 SMP Tue May 10 15:42:40 EDT 2011

Testrun Specification

Name

Host

Architecture

Root image

Test

AUTOMATION CONTROL | Screenshots (6)

Report ID 309790, Topic-track-workload-stress-rhel_6.1_64 - Mozilla Firefox

Firefox | Testplan list | Testplan id 244, t... | Testplan id 241, t... | Testrun id 24988... | Report ID 309790... | Report ID 309790... | Report ID 309790...

track-workload-pmc (r309787)

section-000

ram:	48396
cpuinfo:	24 cores [AMD Engineering Sample]
uname:	Linux arges 2.6.32-131.0.15.el6.x86_64 #1 SMP Tue May 10 15:42:40 EDT 2011 x86_64 x86_64 x86_64 GNU/Linux
osname:	Red Hat Enterprise Linux Server release 6.1 (Santiago)
flags:	ro root=UUID=bfa3526e-0b4d-4306-b6cf-217d063c2158 rd_NO_LUKS rd_NO_LVM rd_NO_MD rd_NO_DM LANG=en_US.UTF-8 SYSFONT=latarcyrheb-sun16 KEYBOARDTYPE=pc KEYTABLE=us crashkernel=129M@0M tapper_host=plutonium tapper_port=1337 console=tty0 console=ttyS0,115200
kernel:	2.6.32-131.0.15.el6.x86_64
changeset:	Linux version 2.6.32-131.0.15.el6.x86_64 (mockbuild@x86-007.build.bos.redhat.com) (gcc version 4.4.4 20100726 (Red Hat 4.4.4-13) (GCC)) #1 SMP Tue May 10 15:42:40 EDT 2011

Testrun Specification

Name

Host

Architecture

Root image

Test

Test results

PASSED

Test file	Test results	%
MCP-overview		100.0%
1 files	2 tests, 2 ok, 0 failed, 0 todo, 0 skipped, 0 parse errors exit status: 0, wait status: 0 elapsed time: 0 wallclock secs (0.05 usr + 0.00 sys = 0.05 CPU)	100.0%

raw TAP report

Attachments

console	158612 Bytes	view	inline | ansi-colored	plain	2011-10-11 18:17:10 GMT
test__opt_tapper_bin_tapper-testsuite-autotest_stderr	452 Bytes	view	inline | ansi-colored	plain	2011-10-11 18:17:10 GMT
test__opt_tapper_bin_tapper-testsuite-autotest_stdout	0 Bytes	view	inline | ansi-colored	plain	2011-10-11 18:17:10 GMT
test__track-workload-pmc_sh_stderr	265 Bytes	view	inline | ansi-colored	plain	2011-10-11 18:17:10 GMT
test__track-workload-pmc_sh_stdout	0 Bytes	view	inline | ansi-colored	plain	2011-10-11 18:17:10 GMT
test__track-workload-upload-results_sh_stderr	136 Bytes	view	inline | ansi-colored	plain	2011-10-11 18:17:10 GMT
test__track-workload-upload-results_sh_stdout	0 Bytes	view	inline | ansi-colored	plain	2011-10-11 18:17:10 GMT

AUTOMATION CONTROL | Screenshots (7)

Report ID 309790, Topic-track-workload-stress-rhel_6.1_64 - Mozilla Firefox

Firefox | Testplan list | Testplan id 244, t... | Testplan id 241, t... | Testrun id 24988... | Report ID 309790... | Report ID 309790... | Report ID 309790...

15:42:40 EDT 2011

Testrun Specification

Name

Host

Architecture

Root image

Test

Test results

PASSED

Test file	Test results	%
MCP-overview	TAP Version 13 1..2 # Tapper-reportgroup-testrun: 249886 # Tapper-suite-name: Topic-track-workload-stress-rhel_6.1_64 # Tapper-suite-version: 3.000010 # Tapper-machine-name: arges # Tapper-section: MCP overview # Tapper-reportgroup-primary: 1 ok 1 - Installation finished ok 2 - Testing finished in PRC 0	100.0%
1 files	2 tests, 2 ok, 0 failed, 0 todo, 0 skipped, 0 parse errors exit status: 0, wait status: 0 elapsed time: 0 wallclock secs (0.05 usr + 0.00 sys = 0.05 CPU)	100.0%

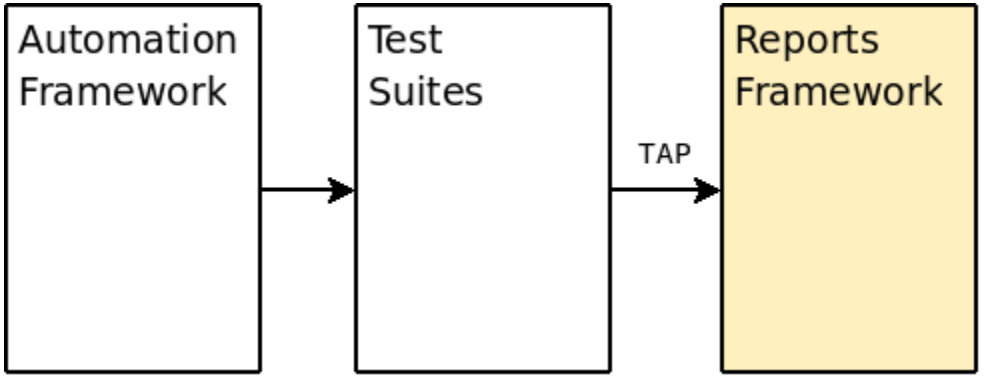
raw TAP report

Attachments

console	158612 Bytes	view	inline | ansi-colored	plain	2011-10-11 18:17:10 GMT
test__opt_tapper_bin_tapper-testsuite-autotest_stderr	452 Bytes	view	inline | ansi-colored	plain	2011-10-11 18:17:10 GMT
test__opt_tapper_bin_tapper-testsuite-autotest_stdout	0 Bytes	view	inline | ansi-colored	plain	2011-10-11 18:17:10 GMT
test__track-workload-pmc_sh_stderr	265 Bytes	view	inline | ansi-colored	plain	2011-10-11 18:17:10 GMT
test__track-workload-pmc_sh_stdout	0 Bytes	view	inline | ansi-colored	plain	2011-10-11 18:17:10 GMT
test__track-workload-upload-results_sh_stderr	136 Bytes	view	inline | ansi-colored	plain	2011-10-11 18:17:10 GMT
test__track-workload-upload-results_sh_stdout	0 Bytes	view	inline | ansi-colored	plain	2011-10-11 18:17:10 GMT

EVALUATE RESULTS

RESULTS



EVALUATE RESULTS | What do we have so far?

- Remember: we dropped TAP into Tapper with “fire & forget” (`netcat`)
  - Hide internal complexity
    - Actual success status
    - Aggregated results
    - Report groups
    - Meta-information
    - Embedded YAML data
    - Any sufficiently advanced technology
      - TAP::Parser
      - TAP::DOM
      - TAP::Formatter::HTML
      - Databases
      - Etc.
- How to trivially access results?

EVALUATE RESULTS | The no-problem

- Web application for “end users”
  - RED / YELLOW / GREEN
  - Cautious but useful Javascript
  - Overviews, details, attachments
  - List, filters, RSS feeds

EVALUATE RESULTS | The no-problem - screenshots (2)

File Edit View History Bookmarks Tools Help

Tapper

Tapper

Start | Testruns | Reports | Metareports | Hardware | Manual

220116: KernBench

report id: 220116 2011-03-24 10:54:30 GMT Host: 'kobold:takujui'

Testrun

ID	DateTime (GMT)	Suite	Machine	Success	Ratio	Grouped by
r220169	2011-03-24 13:12	Topic-autoinstall-kvm-rhel-5.6	kobold	FAIL		testrun 181155 (artemis)
r220168	2011-03-24 13:12	Guest-Overview-2	kobold	FAIL		
r220167	2011-03-24 13:12	Guest-Overview-1	kobold	PASS		
r220166	2011-03-24 13:12	PRC0-Overview	kobold	PASS		
r220116	2011-03-24 10:54	KernBench	kobold:takujui	PASS		
r219876	2011-03-24 03:11	Host-Overview	kobold	PASS		
r219819	2011-03-24 02:00	Hardwaredb Overview	kobold	PASS		

KernBench

Test Execution Context

Host-Overview (r219876)

Metainfo

ram:	5019 MB
cpuinfo:	2x Family: 16, Model: 2, Stepping: 2
uptime:	1 hrs

KVM-Metainfo

kvm_module_version:	kvm-83-224.el5
kvm_userspace_version:	kvm-83-maint-snapshot-20090205
kvm_kernel:	2.6.18-238.el5 x86_64
kvm_base_os_description:	Red Hat Enterprise Linux Server release 5.6 (Tikanga)

guest_1_suse_sles10_sp4_rc1_32bpae_raw

Done

EVALUATION | Query API

QUERY API | The query gap

- Scriptable querying
  - The same ease as reporting
  - Again: shell level, `netcat`
- Use-cases
  - Generally access our own reports
    - Data + attachments
  - Track test success over time
  - Track benchmark results
  - Custom-visualize the data
- Challenges
  - Test suites change over time → fuzzy find
  - Hide the toolchain

QUERY API | The solution

- Provide template mechanism
- With embedded query language “`DPath`”
- Dialog-oriented protocol
  - HERE-doc style
  - → Send template with “`netcat`”
  - ← Receive processed content

QUERY API | Example 1 – Get simple values

- Command

```
$ cat report.mas | netcat tapper 7358 > result.txt
```

- Template

```
#! tt <<EOTEMPLATE
Planned tests:
[% FOREACH x IN reportdata('{ "suite.name" => "power_msr" } :: //tap/tests_planned') -%]
   [% x %]
[% END %]
EOTEMPLATE
```

- Result

```
Planned tests:
   3
   4
   17
```

QUERY API | Example 2 – Fill a GNUPLOT file

- Command

```
$ cat CTCS_ratio.gnuplot | netcat tapper 7358 | gnuplot
```

- Template

```
#! tt <<EOTEMPLATE
TITLE = "success ratio: CTCS"
set output "CTCS_ratio.png"
plot '-' using 0:2 with linespoints
[% time  = reportdata('...') %]
[% ratio = reportdata('...') %]
[% FOREACH i IN … %]
  [% time.$i %] [% ratio.$i %]
[% END %]
EOTEMPLATE
```

- Result
  - Generated file "CTCS_ratio.png"

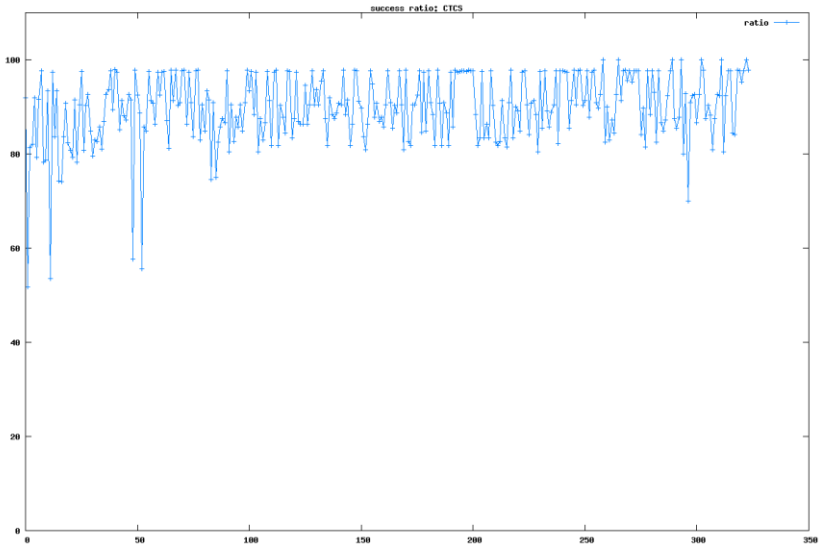
QUERY API | Example 2 – Fill a GNUPLOT file

- Command

```
$ cat CTCS_ratio.gnuplot | netcat tapper 7358 | gnuplot
```

- Template

```
#! tt <<EOTEMPLATE
TITLE = "success ratio: CTCS"
set output "CTCS_ratio.png"
plot '-' using 0:2 with linespoints
[% time  = reportdata('...') %]
[% ratio = reportdata('...') %]
[% FOREACH i IN … %]
  [% time.$i %] [% ratio.$i %]
[% END %]
EOTEMPLATE
```



- Result
  - Generated file "CTCS_ratio.png"

Did you notice? No client-side toolchain dependencies!

QUERY API | How does it work?

- TAP::DOM
  - A data structure (DOM) out of TAP

QUERY API | How does it work?

- TAP::DOM
  - A data structure (DOM) out of TAP

```
{ 'tests_planned' => 6
  'tests_run'     => 8,
  # […]
  'summary' => {
      'status'          => 'FAIL',
      'total'           => 8,
      'passed'          => 6,
      'failed'          => 2,
      'skipped'         => 1,
      'todo'            => 4,
      'todo_passed'     => 2,
      # […]
  },
  'lines' => [
     { 'number'       => '1',
       'is_ok'        =>  1,
       'description'  => '- connection established',
       '_children'    => [ # subsequent comments/yaml
             { 'is_yaml' => 1,
               'data' => [ {'pass1' => '1234.56',
                            'pass2' => '999.99' } ] }
    # […… lines ……]
  ] } }
```

QUERY API | How does it work?

- TAP::DOM
  - A data structure (DOM) out of TAP
- DPath to fuzzy navigate data
  - XPath-like

```
{ 'tests_planned' => 6
  'tests_run'     => 8,
  # […]
  'summary' => {
      'status'          => 'FAIL',
      'total'           => 8,
      'passed'          => 6,
      'failed'          => 2,
      'skipped'         => 1,
      'todo'            => 4,
      'todo_passed'     => 2,
      # […]
  },
  'lines' => [
     { 'number'       => '1',
       'is_ok'        =>  1,
       'description'  => '- connection established',
       '_children'    => [ # subsequent comments/yaml
             { 'is_yaml' => 1,
               'data' => [ {'pass1' => '1234.56',
                            'pass2' => '999.99' } ] }
    # […… lines ……]
  ] } }
```

QUERY API | How does it work?

- TAP::DOM
  - A data structure (DOM) out of TAP
- DPath to fuzzy navigate data
  - XPath-like

```
'/tests_planned'
```

```
{ 'tests_planned' => 6
  'tests_run'     => 8,
  # […]
  'summary' => {
      'status'          => 'FAIL',
      'total'           => 8,
      'passed'          => 6,
      'failed'          => 2,
      'skipped'         => 1,
      'todo'            => 4,
      'todo_passed'     => 2,
      # […]
  },
  'lines' => [
     { 'number'       => '1',
       'is_ok'        =>  1,
       'description'  => '- connection established',
       '_children'    => [ # subsequent comments/yaml
              { 'is_yaml' => 1,
                'data' => [ {'pass1' => '1234.56',
                             'pass2' => '999.99' } ] }
    # […… lines ……]
  ] } }
```

QUERY API | How does it work?

- TAP::DOM
  - A data structure (DOM) out of TAP
- DPath to fuzzy navigate data
  - XPath-like

```
'/tests_planned'
'//tests_planned'
```

```
{ 'tests_planned' => 6
  'tests_run'     => 8,
  # […]
  'summary' => {
      'status'          => 'FAIL',
      'total'           => 8,
      'passed'          => 6,
      'failed'          => 2,
      'skipped'         => 1,
      'todo'            => 4,
      'todo_passed'     => 2,
      # […]
  },
  'lines' => [
     { 'number'       => '1',
       'is_ok'        =>  1,
       'description'  => '- connection established',
       '_children'    => [ # subsequent comments/yaml
              { 'is_yaml' => 1,
                'data' => [ {'pass1' => '1234.56',
                             'pass2' => '999.99' } ] }
    # […… lines ……]
  ] } }
```

QUERY API | How does it work?

- TAP::DOM
  - A data structure (DOM) out of TAP
- DPath to fuzzy navigate data
  - XPath-like

```
'/tests_planned'
'//tests_planned'
'//todo_passed'
```

```
{ 'tests_planned' => 6
  'tests_run'     => 8,
  # […]
  'summary' => {
      'status'          => 'FAIL',
      'total'           => 8,
      'passed'          => 6,
      'failed'          => 2,
      'skipped'         => 1,
      'todo'            => 4,
      'todo_passed'     => 2,
      # […]
  },
  'lines' => [
     { 'number'       => '1',
       'is_ok'        =>  1,
       'description'  => '- connection established',
       '_children'    => [ # subsequent comments/yaml
             { 'is_yaml' => 1,
               'data' => [ {'pass1' => '1234.56',
                            'pass2' => '999.99' } ] }
    # […… lines ……]
  ] } }
```

QUERY API | How does it work?

- TAP::DOM
  - A data structure (DOM) out of TAP
- DPath to fuzzy navigate data
  - XPath-like

```
'/tests_planned'
'//tests_planned'
'//todo_passed'
'//data//pass2'
```

```
{ 'tests_planned' => 6
  'tests_run'     => 8,
  # […]
  'summary' => {
      'status'          => 'FAIL',
      'total'           => 8,
      'passed'          => 6,
      'failed'          => 2,
      'skipped'         => 1,
      'todo'            => 4,
      'todo_passed'     => 2,
      # […]
  },
  'lines' => [
     { 'number'       => '1',
       'is_ok'        =>  1,
       'description'  => '- connection established',
       '_children'    => [ # subsequent comments/yaml
            { 'is_yaml' => 1,
              'data'  => [ {'pass1' => '1234.56',
                            'pass2' => '999.99' } ] }
    # [… lines …]
  ] } }
```

QUERY API | How does it work?

- TAP::DOM
  - A data structure (DOM) out of TAP
- DPath to fuzzy navigate data
  - XPath-like

```
'/tests_planned'
'//tests_planned'
'//todo_passed'
'//data//pass2'
'//lines/*/is_ok'
```

```
{ 'tests_planned' => 6
  'tests_run'     => 8,
  # […]
  'summary' => {
      'status'          => 'FAIL',
      'total'           => 8,
      'passed'          => 6,
      'failed'          => 2,
      'skipped'         => 1,
      'todo'            => 4,
      'todo_passed'     => 2,
      # […]
  },
  'lines' =>
      { 'number'        => '1',
        'is_ok'         =>  1,
        'description'   => '- connection established',
        '_children'     => [ # subsequent comments/yaml
              { 'is_yaml' => 1,
                'data' => [ {'pass1' => '1234.56',
                             'pass2' => '999.99' } ] }
    # […… lines ……]
  ] } }
```

QUERY API | Anatomy of a Tapper::DPath

```
{ suite_name => "CTCS" } :: //tests_planned[value > 10]/../summary/passed
```

- Virtual DOM of the TAP database
- Two orthogonal concepts
  - Database axis: provide but hide relational access
    - SQL::Abstract
    - The “history of reports”
  - Report axis: inside single reports data structure
    - TAP::DOM
    - Data::DPath
    - Inside “one point in history”

MORE

MORE | Topics

- Benchmark sub-infrastructure
  - Subscribe to incoming data with DPaths (`//data//codespeed/*`)
  - Pass-through to benchmark rendering application → Codespeed

- Integration with TaskJuggler
  - Map task IDs to filesystem hierarchy of testplan files
    - `osrc.productfoo.xen.power_msr.xen4_3` → `osrc/productfoo/xen/power_msr/xen4_3`
    - Schedule testruns by taskjuggler task dates
    - Report back results per E-Mail as TaskJuggler timesheets
    - TaskJuggler renders project status from that

- Deployment
  - Bootstrap your own infrastructure
  - Create utility images, client packages

SUMMARY

SUMMARY | Framework

- Complete testing environment fitting several parties' needs
  - Test team
    - Automation to run machine pool
  - Developer / Tester
    - Support on developing and running tests
    - Locally and/or automated
  - Manager / Tester
    - Visual presentation of test results
    - QA lifecycle, driven by planning software
- Built on top of open source standards
- Provide complexity – allow simplicity

- The End – Thank You!

http://amd64.org/mailman/listinfo/tapper

irc.freenode.net / #tapper

github.com/amd