Benchmark::Perl::Formance for the masses

Steffen "renormalist" Schwigon

Dresden Perl Mongers

16 August 2011

Abstract

- **Abstract**

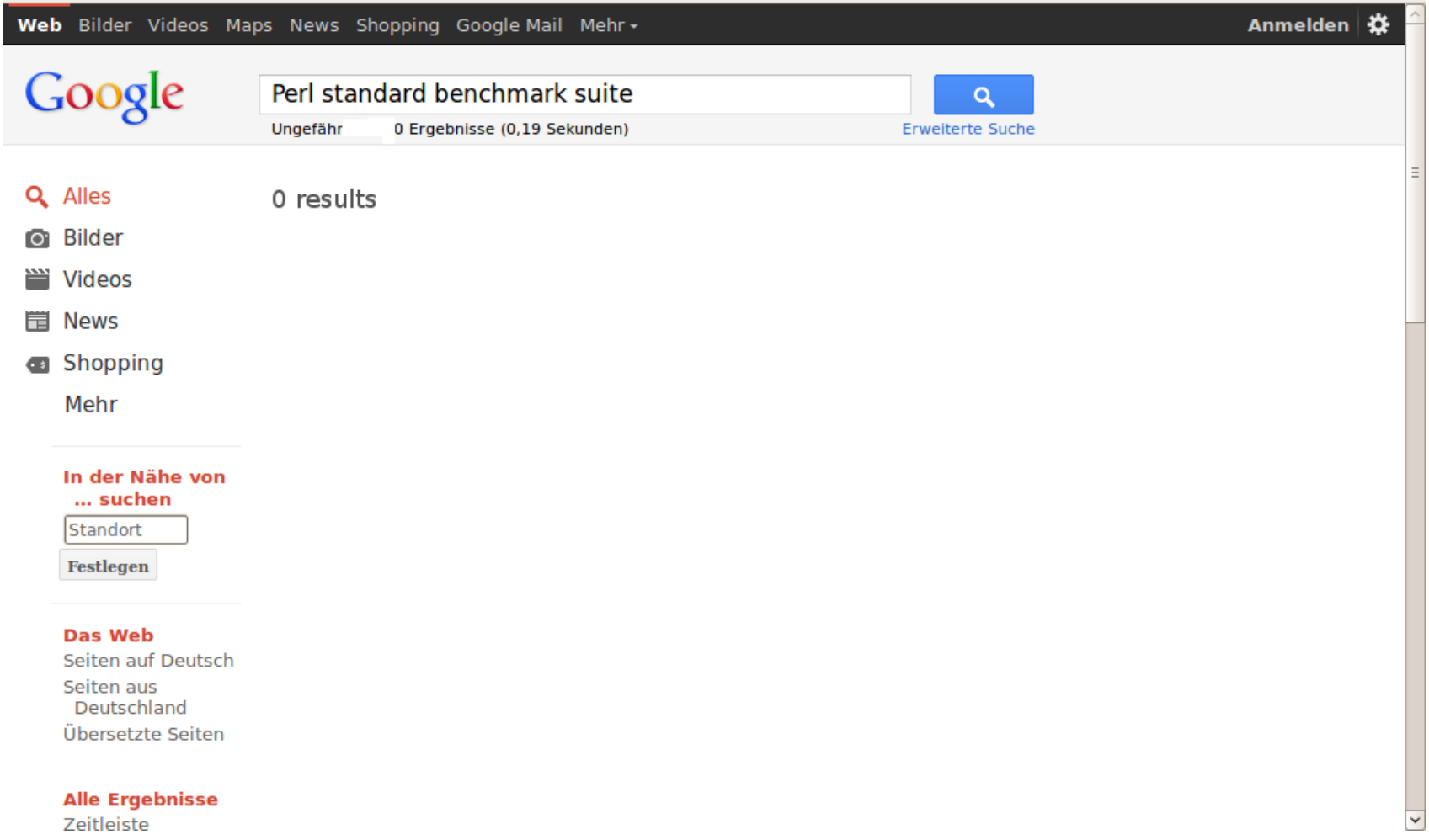
Abstract

- Perl lacks benchmarking
- I work on a full-coverage solution
  - benchmark implementation
  - complete "build - execute - evaluate" lifecycle
  - benchmark result database
  - graphically present numbers and details
- Present the vision and state

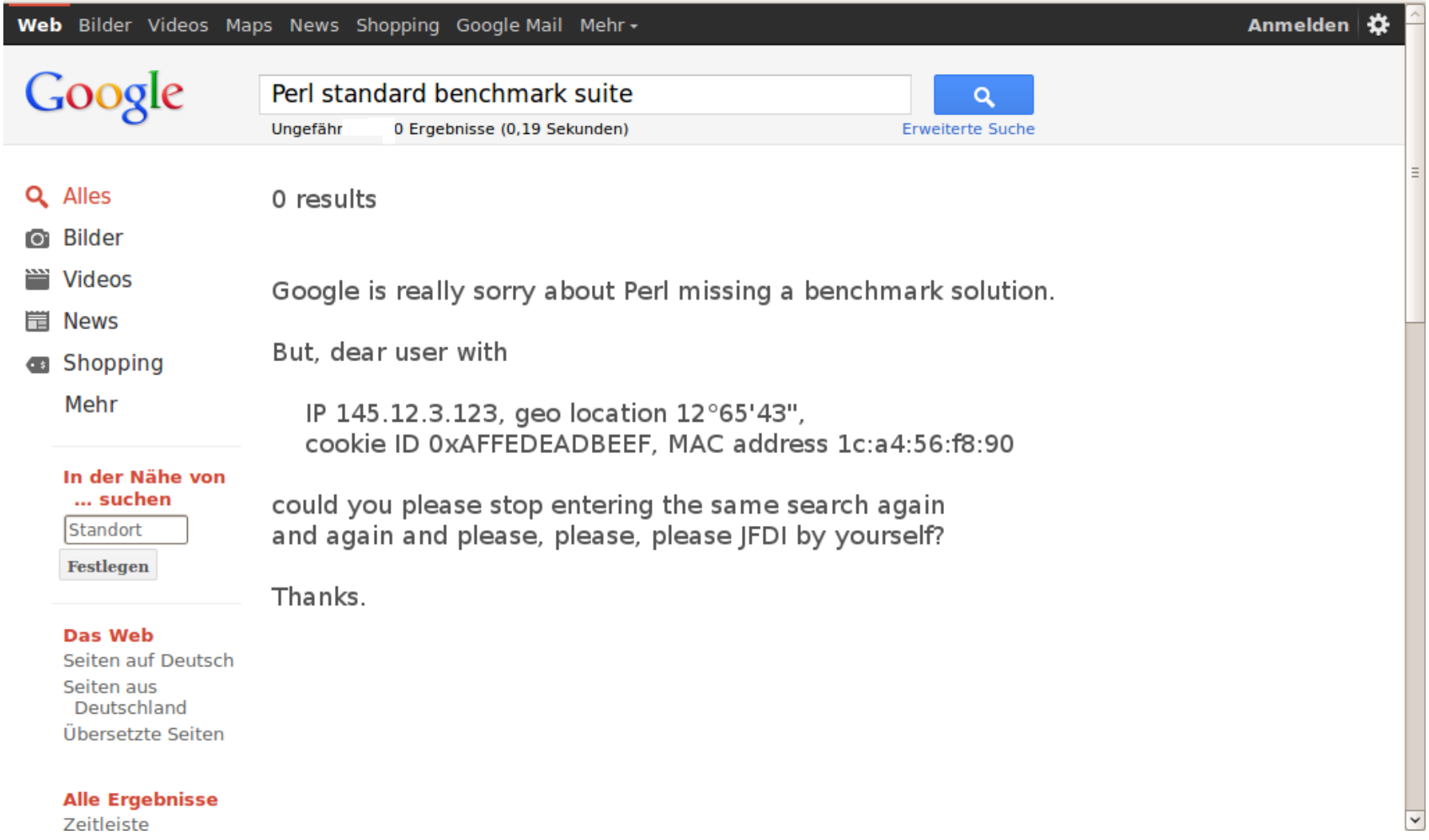
Motivation

- **Motivation**

Perl standard benchmark suite?



Perl standard benchmark suite?



Perl standard benchmark suite?

Web Bilder Videos Maps News Shopping Google Mail Mehr Anmelden

Google

Perl standard benchmark suite

Ungefähr 0 Ergebnisse (0,19 Sekunden) Erweiterte Suche

Alles
Bilder
Videos
News
Shopping
Mehr

In der Nähe von ... suchen
Standort
Festlegen

Das Web
Seiten auf Deutsch
Seiten aus Deutschland
Übersetzte Seiten

Alle Ergebnisse
Zeitleiste

0 results

Google is really sorry about Perl missing a benchmark solution.

But, dear user with

IP 145.12.3.123, geo location 12°65'43",
cookie ID 0xAFFEDEADBEEF, MAC address 1c:a4:56:f8:90

could you please stop entering the same search again and again and please, please, please JFDI by yourself?

Thanks.

Measuring

- **Measuring**

The most important benchmark question

- **What do you want to measure?**
- Primary goal: perl5, the interpreter
  - real world workloads
  - micro benchmarks
  - usethreads vs. no usethreads
  - use64bitall vs. no use64bitall
- Secondary goal: misc Perl5 topics
  - subs vs. methods
  - Moose
  - regex engines

Workloads

- **Workloads**

Where is Perl?

- Everywhere and nowhere
- Benchmarks are a corporate topic
- Perl's usual corporate visibility problem

"The One Workload"

- No dedicated single heavy applications
- Except SpamAssassin
  - important app
  - heavy for a reason
  - hit by 5.12 deprecations
  - handholding on new CPAN release

Self-referential - Perl for Perl

- Refer to Perl itself
  - CPAN.pm
  - Perl::Critic
  - Perl6 STD.pm (gimme5, viv)

Nothing else!

- Not much more

Writing my own benchmark suite

- Goals
  - Have benchmark workloads
    - "server" workloads (MEM! + TIME!)
    - "desktop" workloads (small + fast)
    - "micro" benchmarks (language features)
    - "multi purpose" workloads (average, not fitting in others)
  - Provide general benchmarking umbrella

Perl::Formance

- **Perl::Formance**

The Perl::Formance benchmark suite

- Collect existing heavy code
- Collect existing benchmark snippets
- Write new benchmark code from scratch
- Measure execution time ("smaller is better")
- http://xrl.us/cpanperlformance

Perl::Formance features (1)

- Heavy workloads
  - The one: SpamAssassin
    - Type: "server"
    - Example corpus from spamassassin.org
    - Run sa-learn

Perl::Formance features (2)

- Wrap existing benchmarks
  - "Language Shootout" on alioth.debian.org
    - Type: "multi purpose"
    - binarytrees
    - fasta
    - nbody
    - pidigits
    - regexdna
    - revcomp
    - spectralnorm
    - fannkuch (!)
    - knucleotide (!)
    - mandelbrot (!)
  - Mostly copy'n'paste into Perl::Formance plugins

Perl::Formance features (3)

- Known heavy code from CPAN
  - Heavy test suites
    - Type: "desktop"
    - Moose test suite
    - Regexp::Common
  - Perl6 STD.pm
    - Type: "desktop"
    - gimme5 STD.pm6
    - viv STD.pm6
  - Data::DPath

Perl::Formance features (4)

- Stress of language features
  - Stress recursion (use Fibonacci numbers as vehicle)
    - Type: "micro"
    - subs - plain Perl
    - methods - plain Perl
    - methods - Moose
    - methods - MooseX::Declare

Perl::Formance features (5)

- Pathological Regular Expressions
  - Type: "micro"
  - Known pathological issues
  - my $re = ("a?" x $n) . ("a" x $n);
  - See

    `http://swtch.com/~rsc/regexp/regexp1.html`

Perl::Formance features (6)

- Compare different regex engines
  - Pluggable regex engines since Perl 5.10
  - Again pathological regular expressions
  - Type: "micro"
- Regex engines
  - native
  - POSIX::Regex
  - re::**engine**::Plan9
  - re::**engine**::PCRE
  - re::**engine**::Lua
  - re::**engine**::LPeg (confusion bonus)
  - re::**engine**::Oniguruma

Different output styles (1)

- Human readable

```
$ benchmark-perlformance --fastmode [...]
Rx.regexes.fieldsplit1   : 1.267907
Rx.regexes.fieldsplit2   : 2.106220
Rx.regexes.pathological  : 1.000129
Shootout.binarytrees     : 1.046751
Shootout.fasta           : 2.270553
Shootout.nbody           : 1.685537
Shootout.spectralnorm    : 1.855935
```

Different output styles (2)

- Evaluation friendly (–outstyle=yaml)

```
---
perlformance:
  config:
    fastmode: 1
results:
  Shootout:
    fasta:
      Benchmark:
        - 0.263123989105225
        - 0.25
        - 0
        - 0
        - 1
      count: 1
      goal: 5000
...
```

Output style

- Augment YAML with surrounding TAP

```
--tapdescription="some description"
```

- Additional Codespeed data structure

```
--codespeed
```

- Why TAP?
  - TAP is my hammer
  - Later embed into Tapper infrastructure

Plugin API

- Very lax, just namespace + sub main()

```
Benchmark::Perl::Formance::Plugin::Skeleton
Benchmark::Perl::Formance::Plugin::*
sub main($options) {
                        ...
                        return { Benchmark => $t,
                                 ...
                               };
                    }
```

Challenges

- **Challenges**

Automatic building

Automatic building

- **sigh**

Automatic building

Automatic building

- Working throughout Perl's git history (5.8..blead)
- Cherry-pick fixes from the future (5.8.x)
- Inconsistent version tags
- perlbrew?
  - No git
  - No bootstrapping CPAN, distroprefs (ANDK++), etc.

The Art of Benchmarking (1)

- Stable benchmark environment
  - Perl <-> OS interaction
    - Flush Caches
    - Address Space Layout Randomisation (ASLR)
    - Disable Core Performance Boost
  - Provide stable set of CPAN dependencies
    - Own CPAN mirror
    - Only sync once in a time
  - Stable execution environment
    - Machine, Memory, Harddisk
    - Operating System
    - Compile toolchain
  - Lazily load benchmark plugins+dependencies after fork

The Art of Benchmarking (2)

- I/O everywhere
  - depends on the workload
  - SPECcpu 2006 (Perl 5.8.7) avoids I/O
  - I do **not** fight that battle

Infrastructure

- CPAN module obviously not enough
- I should solve all other challenges, too
- Set up complete infrastructure

Infrastructure

- **Infrastructure**

Infrastructure

- perl64.org - **Benchmark** machine
  - Rent dedicated machine
  - 6-core AMD Opteron 4180
  - Without any running services
  - No private data, for relaxed access
- perlformance.net - "**Tapper**" application
- speed.perlformance.net - "**Codespeed**" application

Single Run

- Builds Perl
  - from git
  - inject CPAN toolchain
  - Perl 5.8 .. blead
- Installs Benchmark::**Perl:**:Formance
- Installs Tapper::**TestSuite:**:Benchmark::Perl::Formance
- Runs tapper-testsuite-benchmark-perl-formance
  - runs benchmark-perlformance
    - –outstyle=yaml
    - –tapdescription="perlformance"
    - –codespeed
  - adds meta information for Tapper
  - sends report to Tapper server

Multiple runs

- Tapper

Tapper

- Test infrastructure, open sourced by AMD in 2011
  - Overview:
    - http://developer.amd.com/zones/opensource/AMDTapper
  - Source:
    - http://github.com/amd
  - Productized:
    - Starterkit/Deployment, Docs, Wiki

Tapper

- Why Tapper?
  - Tapper == TAP database + automation + scheduler
  - Query API for detailed data forensics
    - see my YAPC::EU 2009 presentation
    - "Cinderella 'TAP - The lazy evaluation sisters of TAP::Parser"
    - http://xrl.us/lazytap (PDF)

Tapper

- Why Tapper?
  - Schedule / intermix different use-cases
    - auto-rerun for blead/threads
    - auto-rerun for blead/nothreads
    - specific commits
    - completely different stuff I add later (Perl6 benchmarks?, smoke tests?)
  - Have bandwidth ratio for each,
    - eg. more non-threaded than threaded
  - Advanced automation support, timeout handling, etc.
  - Frontends to start single runs
    - cmdline + webGUI
    - with requested options (–commitid, –threads, –64bit)
  - Pass-through data chunks to Codespeed application

Codespeed application

- http://github.com/tobami/codespeed
- Web application
- Render benchmark graphs
- Show commit/meta information
- Comparison graphs, baselines, etc.
- Can handle git repos

Own CPAN mirror

- http://perlformance.net/CPAN
- To guarantee no unexpected changes
  - slower/faster/break

Summary

- **Summary**

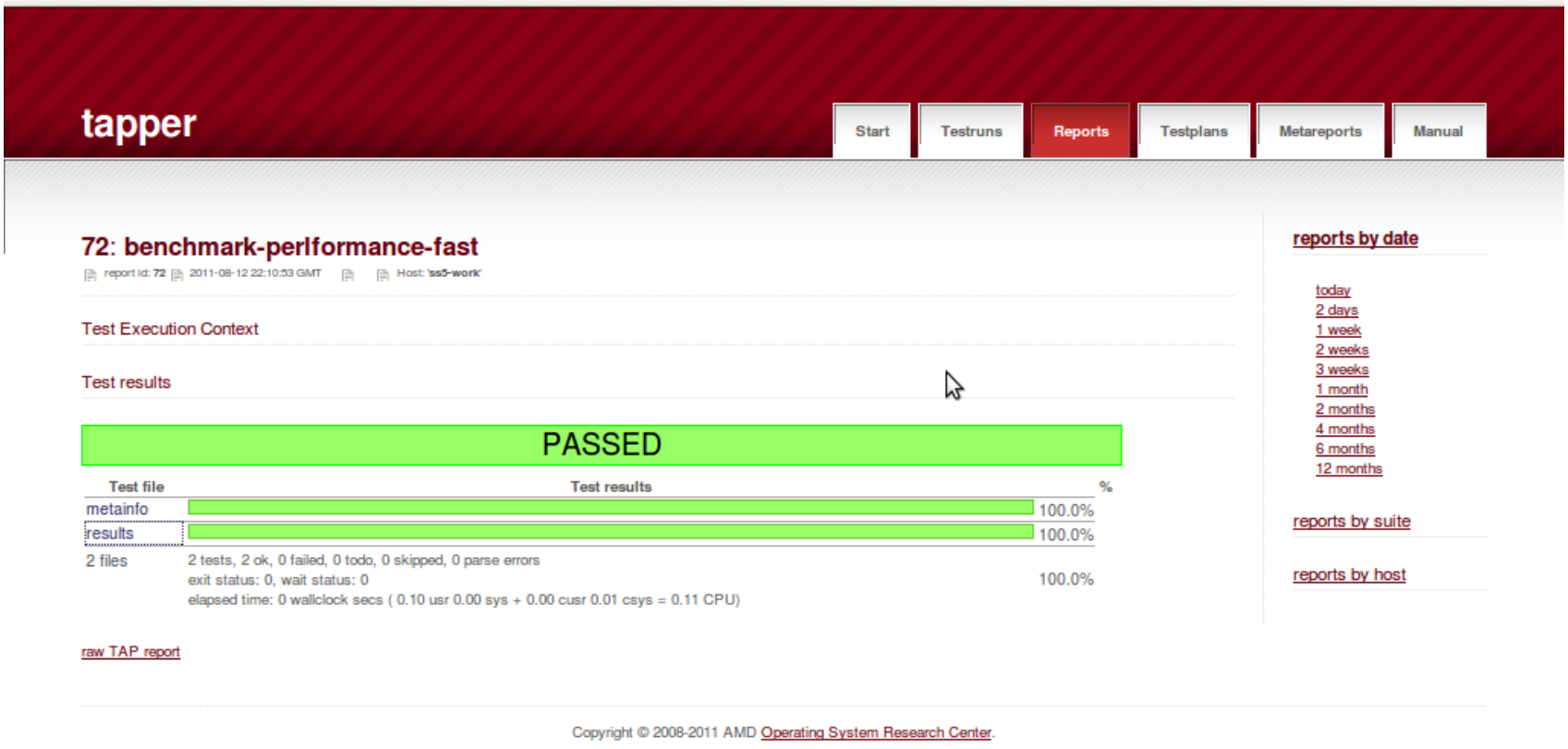
Summary

- I have benchmarks
- I have automatic Perl + CPAN setup
- I have applications for complete infrastructure
- I have dedicated benchmarking server
- I have dedicated CPAN mirror
- Polished in many details
- **TODO** :
  - 1. Re-setup after server went mental
  - 2. Allow users to request benchmark runs

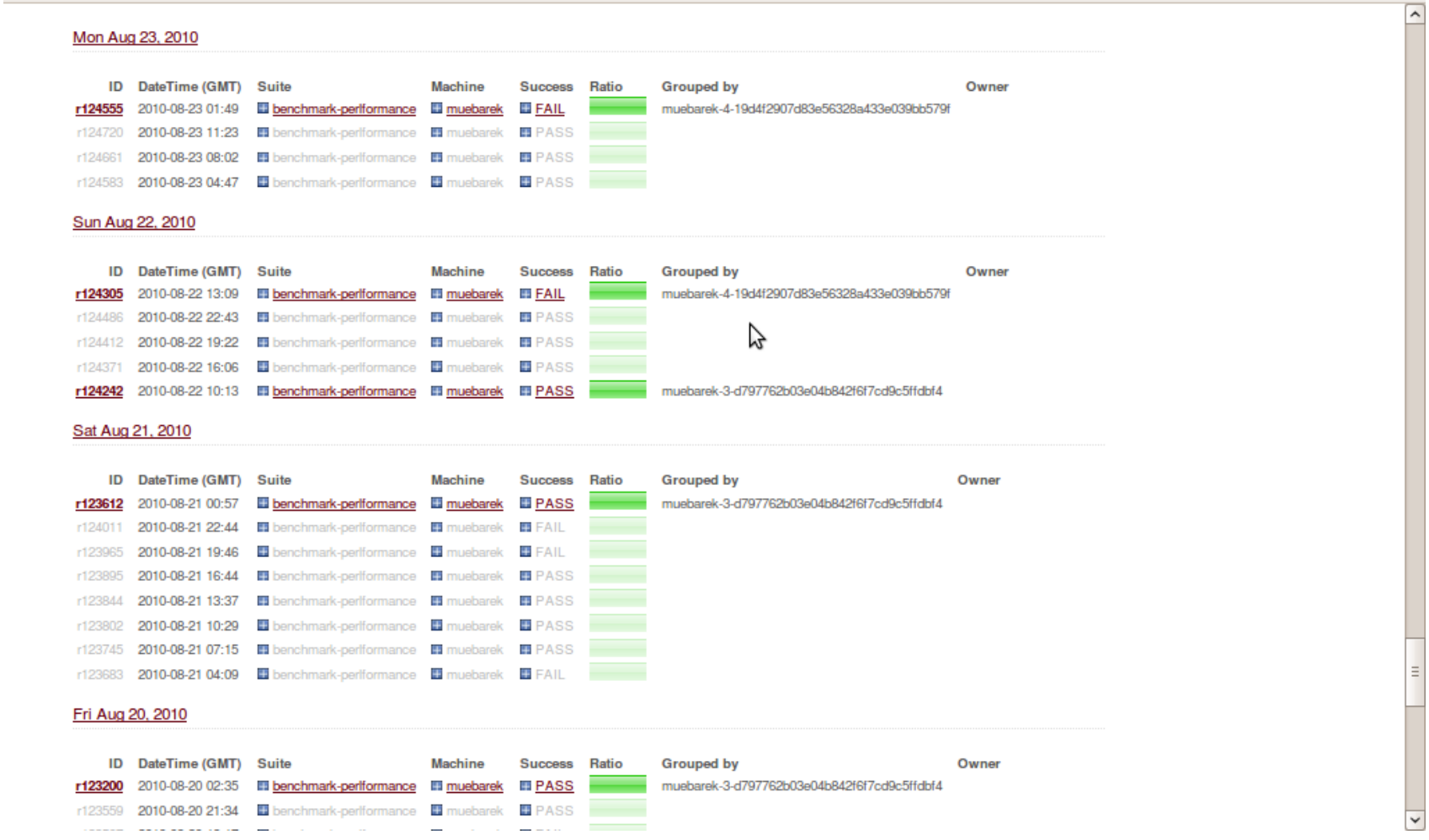
Bonus Screenshots

- **Bonus Screenshots**

Tapper at http://perlformance.net



Tapper - Overview lists



Mon Aug 23, 2010

ID	DateTime (GMT)	Suite	Machine	Success	Ratio	Grouped by	Owner
r124555	2010-08-23 01:49	benchmark-perlformance	muebarek	FAIL		muebarek-4-19d4f2907d83e56328a433e039bb579f	
r124720	2010-08-23 11:23	benchmark-perlformance	muebarek	PASS			
r124661	2010-08-23 08:02	benchmark-perlformance	muebarek	PASS			
r124583	2010-08-23 04:47	benchmark-perlformance	muebarek	PASS			

Sun Aug 22, 2010

ID	DateTime (GMT)	Suite	Machine	Success	Ratio	Grouped by	Owner
r124305	2010-08-22 13:09	benchmark-perlformance	muebarek	FAIL		muebarek-4-19d4f2907d83e56328a433e039bb579f	
r124486	2010-08-22 22:43	benchmark-perlformance	muebarek	PASS			
r124412	2010-08-22 19:22	benchmark-perlformance	muebarek	PASS			
r124371	2010-08-22 16:06	benchmark-perlformance	muebarek	PASS			
r124242	2010-08-22 10:13	benchmark-perlformance	muebarek	PASS		muebarek-3-d797762b03e04b842f6f7cd9c5ffdbf4	

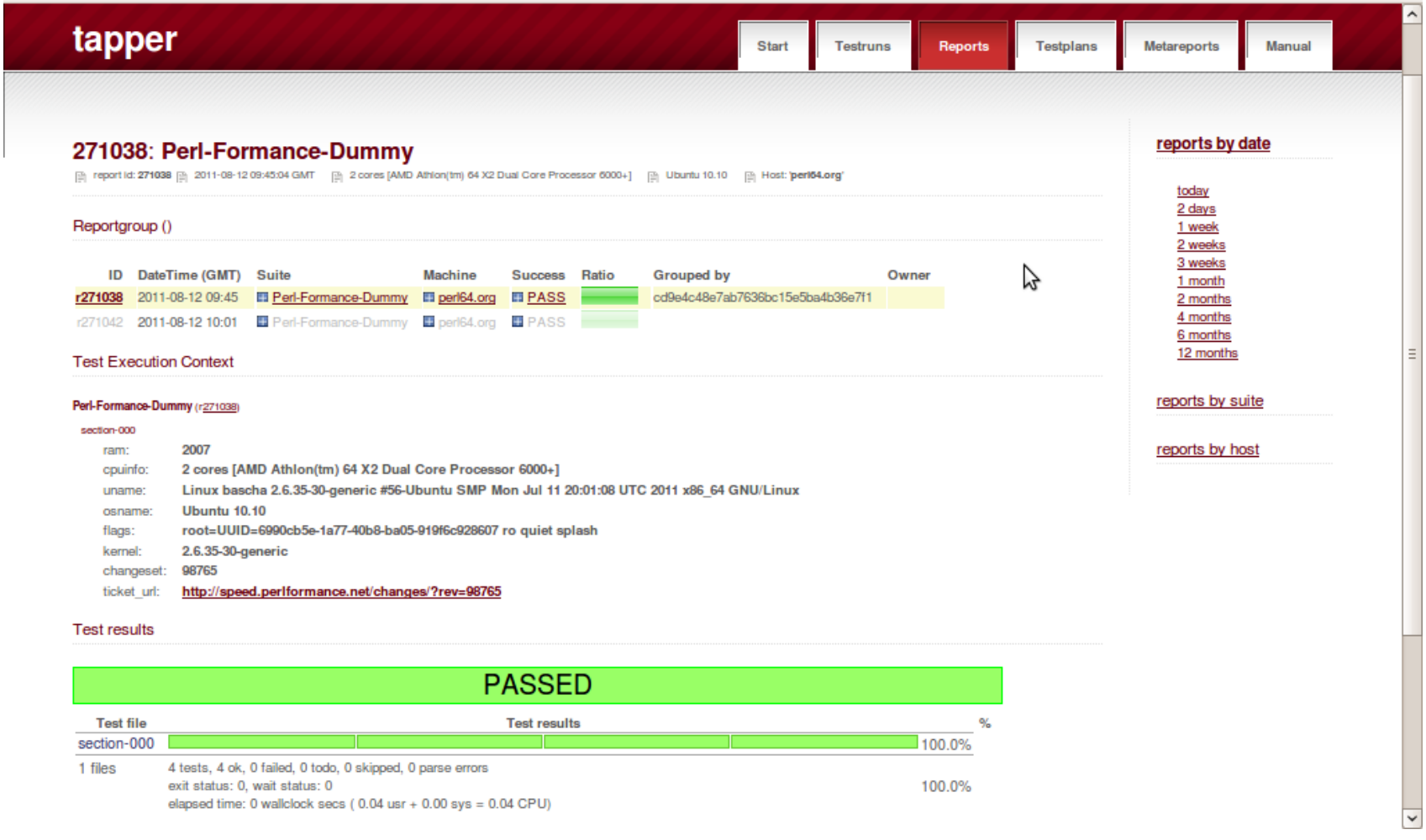
Sat Aug 21, 2010

ID	DateTime (GMT)	Suite	Machine	Success	Ratio	Grouped by	Owner
r123612	2010-08-21 00:57	benchmark-perlformance	muebarek	PASS		muebarek-3-d797762b03e04b842f6f7cd9c5ffdbf4	
r124011	2010-08-21 22:44	benchmark-perlformance	muebarek	FAIL			
r123965	2010-08-21 19:46	benchmark-perlformance	muebarek	FAIL			
r123895	2010-08-21 16:44	benchmark-perlformance	muebarek	PASS			
r123844	2010-08-21 13:37	benchmark-perlformance	muebarek	PASS			
r123802	2010-08-21 10:29	benchmark-perlformance	muebarek	PASS			
r123745	2010-08-21 07:15	benchmark-perlformance	muebarek	PASS			
r123683	2010-08-21 04:09	benchmark-perlformance	muebarek	FAIL			

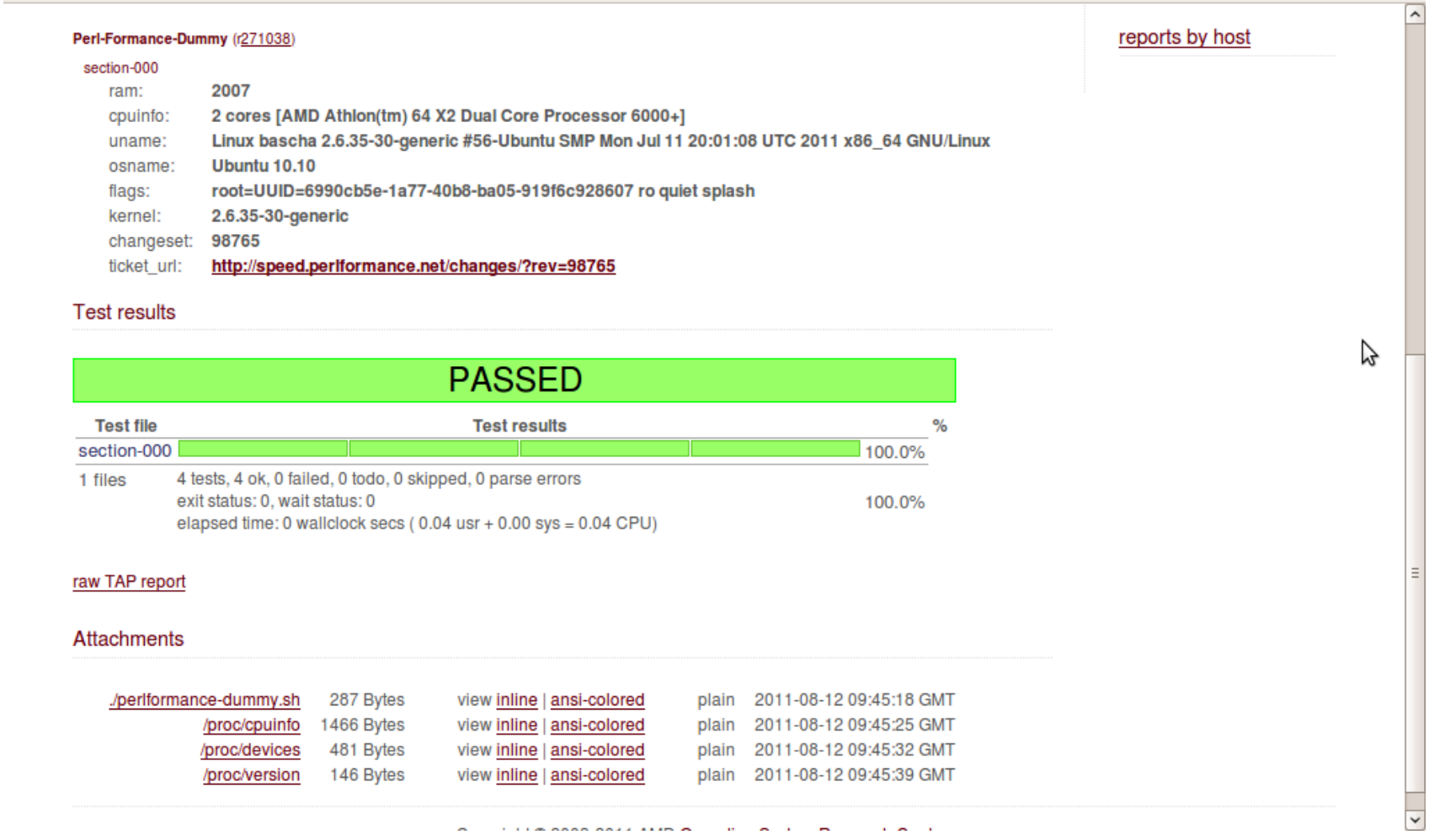
Fri Aug 20, 2010

ID	DateTime (GMT)	Suite	Machine	Success	Ratio	Grouped by	Owner
r123200	2010-08-20 02:35	benchmark-perlformance	muebarek	PASS		muebarek-3-d797762b03e04b842f6f7cd9c5ffdbf4	
r123559	2010-08-20 21:34	benchmark-perlformance	muebarek	PASS			

Tapper - Metainfo



Tapper - Metainfo



Perl-Formance-Dummy (r271038)

reports by host

section-000

ram:	**2007**
cpuinfo:	**2 cores [AMD Athlon(tm) 64 X2 Dual Core Processor 6000+]**
uname:	**Linux bascha 2.6.35-30-generic #56-Ubuntu SMP Mon Jul 11 20:01:08 UTC 2011 x86_64 GNU/Linux**
osname:	**Ubuntu 10.10**
flags:	**root=UUID=6990cb5e-1a77-40b8-ba05-919f6c928607 ro quiet splash**
kernel:	**2.6.35-30-generic**
changeset:	**98765**
ticket_url:	**http://speed.perlformance.net/changes/?rev=98765**

Test results

PASSED

Test file	Test results	%
section-000		100.0%

1 files — 4 tests, 4 ok, 0 failed, 0 todo, 0 skipped, 0 parse errors
exit status: 0, wait status: 0 — 100.0%
elapsed time: 0 wallclock secs (0.04 usr + 0.00 sys = 0.04 CPU)

raw TAP report

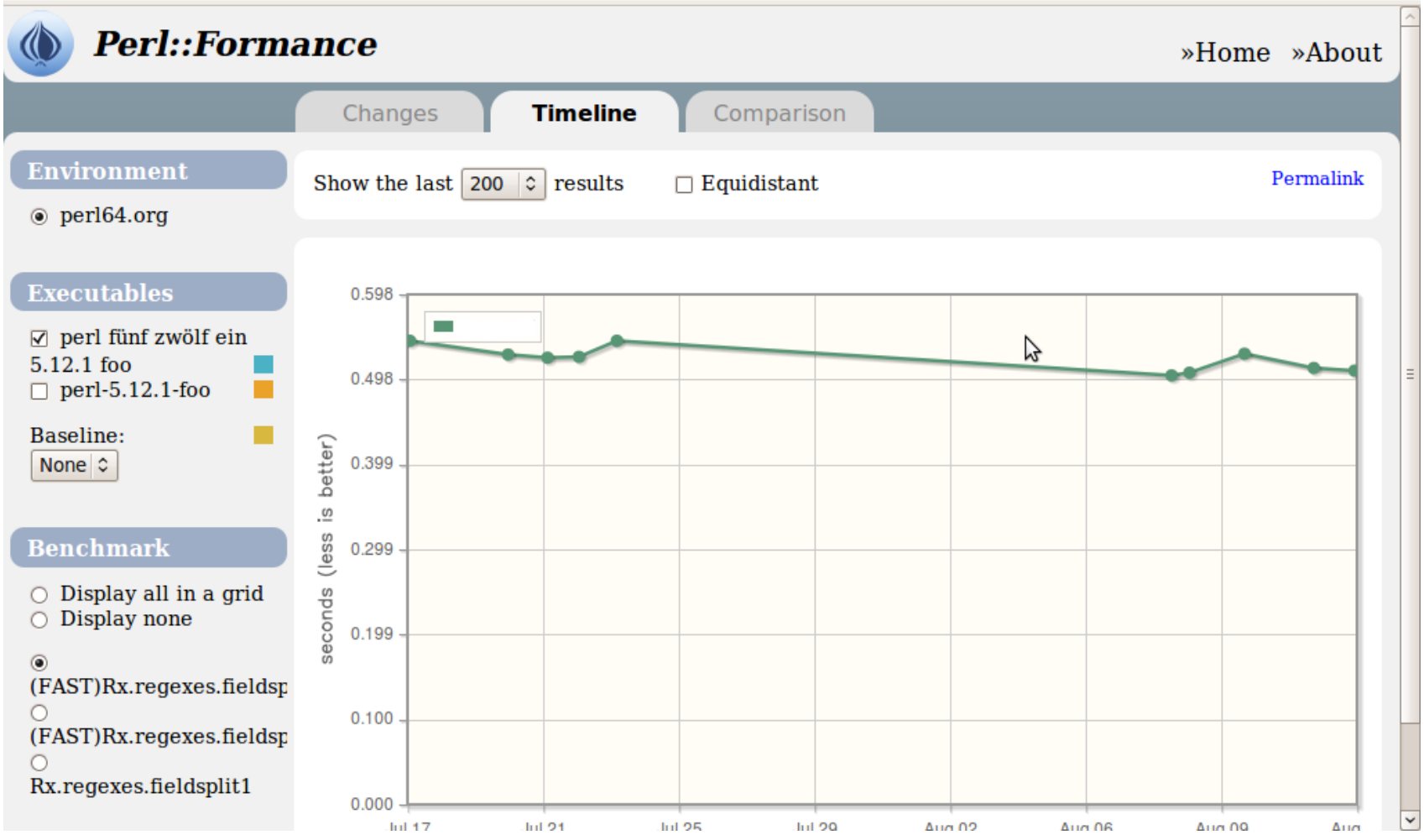
Attachments

./perlformance-dummy.sh	287 Bytes	view inline | ansi-colored	plain	2011-08-12 09:45:18 GMT
/proc/cpuinfo	1466 Bytes	view inline | ansi-colored	plain	2011-08-12 09:45:25 GMT
/proc/devices	481 Bytes	view inline | ansi-colored	plain	2011-08-12 09:45:32 GMT
/proc/version	146 Bytes	view inline | ansi-colored	plain	2011-08-12 09:45:39 GMT

Tapper - Tracked data


```
    - 0
    - 0
    - 0.03
    - 5
  goal: STD.pm6
 viv:
  Benchmark:
   - 0.0236821174621582
   - 0
   - 0
   - 0
   - 0
   - 5
  goal: STD.pm6
PerlCritic:
 PLUGIN_VERSION: 0.001
 bundled:
  Benchmark:
   - 29.1310901641846
   - 0
   - 0
   - 28.96
   - 0.16
   - 2
  count: 2
  perl_critic_version: 1.108
 upstream:
  Benchmark:
   - 30.2070469856262
   - 0
   - 0
   - 29.71
   - 0.13
   - 2
  count: 2
  perl_critic_version: 1.108
```

Codespeed at http://speed.perlformance.net



Thanks.

- **Thanks.**