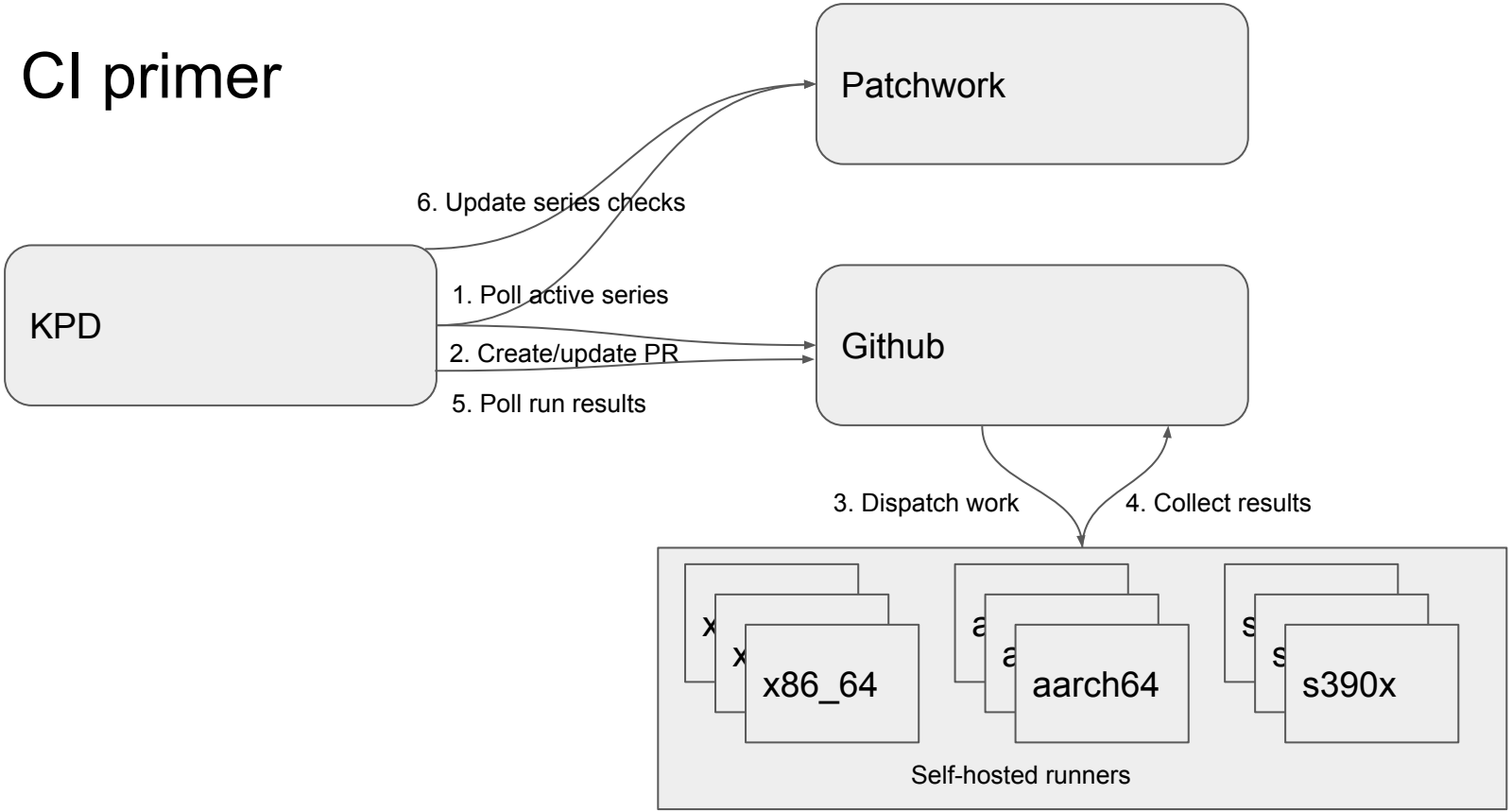
BPF CI

One year later

Manu Bretelle, Daniel Müller, Mykola Lysenko - Meta - May 2023

BPF CI primer



LSFMMBPF 2022: https://docs.google.com/presentation/d/1RQZjLkbXmSFOr_4Sj5BdQsXbUh_vMshXi7w09pUpWsY/

Using CI before sending to ML [0]

- Fork kernel-patches/bpf
- Create a branch off bpf/bpf-next branch
- Push branch to your fork
- Create PR against kernel-patches/bpf@bpf{-next,}_base

Note: First-time contributors need approval [1]

[0] https://tinyurl.com/bpf-ci-test
[1] https://docs.github.com/en/actions/managing-workflow-runs/approving-workflow-runs-from-public-forks

CI coverage

- 2022:
  - Architectures: x86_64 and s390x
  - Kernel Compilation: gcc
  - Build kernel, selftests, and run selftests in VM

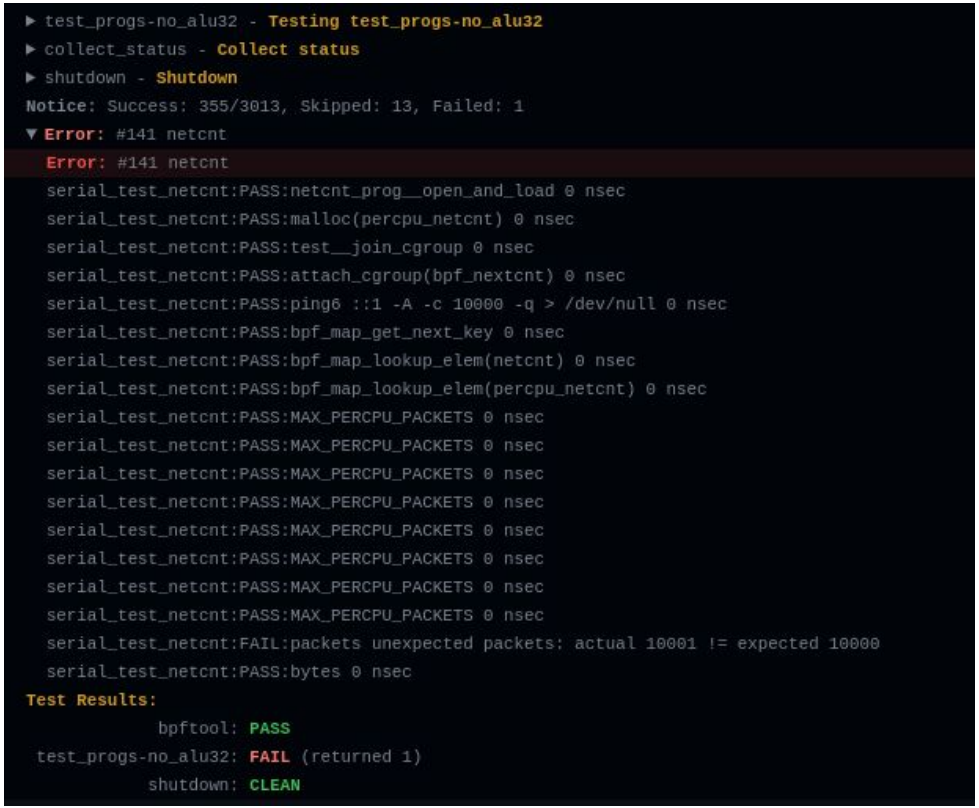
- Now:
  - Architectures: x86_64, s390x, and aarch64
  - Kernel compilation: gcc and llvm for x86_64/aarch64, gcc only for s390x
  - Build kernel, selftests, and run selftests in VM
  - Run veristat on x86_64 to catch regressions
  - 75% of kernel/bpf covered per LCOV

Infra/Testing changes

- x86_64/aarch64 running on bare metal
  - More cores to share
  - Faster VM runs (KVM not available on AWS VMs)
  - Build + test time from 14 min to 8 min
- test_* run in parallel
  - Overhead of building VM image for each test runs
  - Parallelized test time
  - More resources traded for faster turn around/clearer fault isolation
- Incremental kernel build
  - Beneficial for GH hosted runners
  - Shave few minutes off build time on s390x

UI improvement

- Failed test/error logs directly accessible from GH UI
- Accessible from Workflow summary



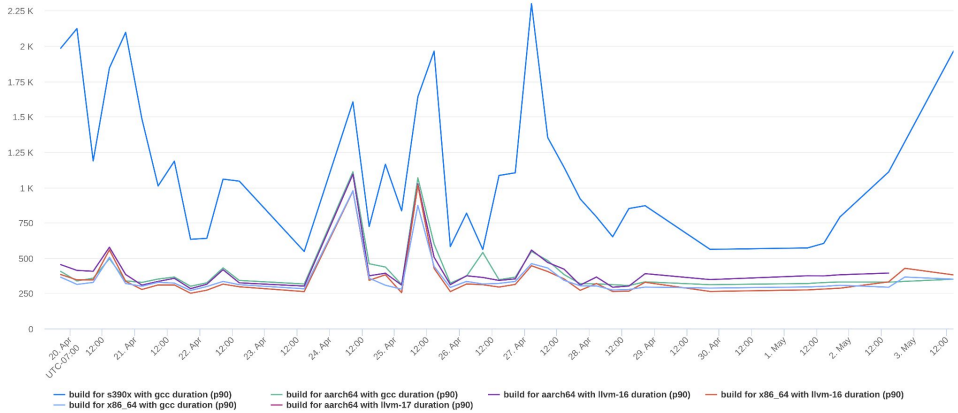
Observability

- GH actions UI is limited
  - success/failure/cancelled
  - historical list of runs
  - Some limited filters
  - Generic to cater for wide audience
- REST API gives access to run results
  - Run, jobs, steps name, durations, results
  - Base commit, PR commit….
- Can poll API to collect and save data for later analysis/10,000ft overview
- Can answer:
  - “when did this test regress”
  - “how long does it take to build X, test X, per compiler, per arch…”
  - “rate of successes/failure, number of cancelled jobs”

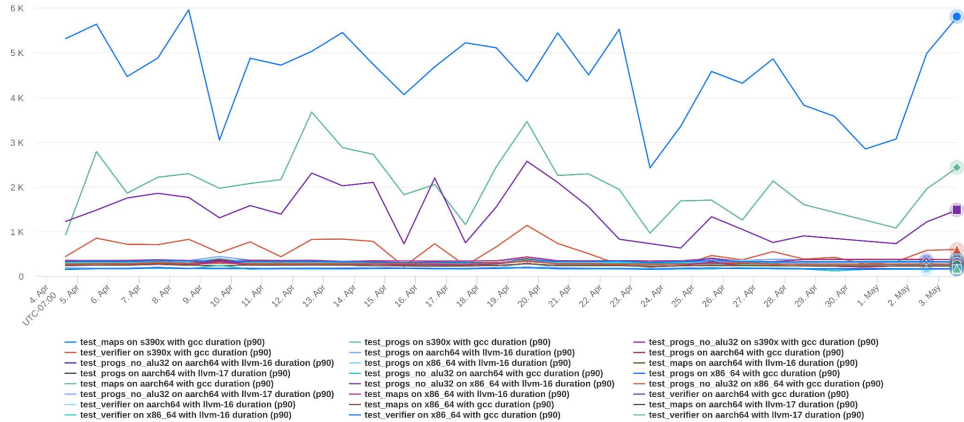
Observability

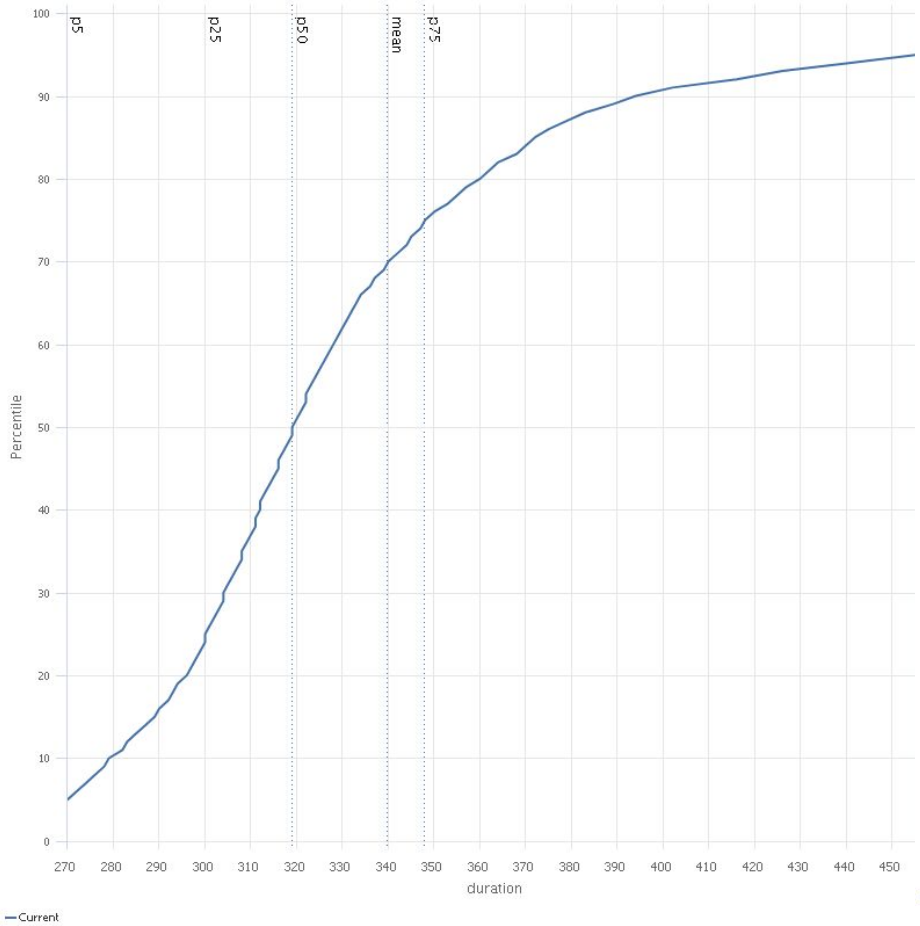
Job Annotations	arch	compiler	Hits	Samples
failure: #24 btf_dump:failure: #24/14 btf_dump/btf_dump: struct_data	aarch64	gcc	12 (13.6%)	12 (13.6%)
failure: #24 btf_dump:failure: #24/14 btf_dump/btf_dump: struct_data	x86_64	llvm-16	12 (13.6%)	12 (13.6%)
failure: #24 btf_dump:failure: #24/14 btf_dump/btf_dump: struct_data	aarch64	llvm-16	12 (13.6%)	12 (13.6%)
failure: #24 btf_dump:failure: #24/14 btf_dump/btf_dump: struct_data	x86_64	gcc	11 (12.5%)	11 (12.5%)
failure: #24 btf_dump:failure: #24/14 btf_dump/btf_dump: struct_data	s390x	gcc	10 (11.4%)	10 (11.4%)
failure: #196 sockopt:failure: #196/31 sockopt/setsockopt: ignore >PAGE_SIZE optle	s390x	gcc	6 (6.8%)	6 (6.8%)
failure: #227 test_global_funcs:failure: #227/1 test_global_funcs/global_func1	aarch64	gcc	4 (4.5%)	4 (4.5%)
failure: #227 test_global_funcs:failure: #227/1 test_global_funcs/global_func1	aarch64	llvm-17	4 (4.5%)	4 (4.5%)
failure: #141 netcnt	aarch64	gcc	3 (3.4%)	3 (3.4%)
failure: #195 sockmap_listen:failure: #195/77 sockmap_listen/sockmap VSOCK test_vs	s390x	gcc	2 (2.3%)	2 (2.3%)
failure: #195 sockmap_listen:failure: #195/77 sockmap_listen/sockmap VSOCK test_vs	x86_64	llvm-16	1 (1.1%)	1 (1.1%)
failure: #24 btf_dump:failure: #24/14 btf_dump/btf_dump: struct_data:failure: #356	x86_64	gcc	1 (1.1%)	1 (1.1%)
failure: #224 tc_redirect:failure: #224/5 tc_redirect/tc_redirect_dtime	x86_64	llvm-16	1 (1.1%)	1 (1.1%)

Build time (in sec)

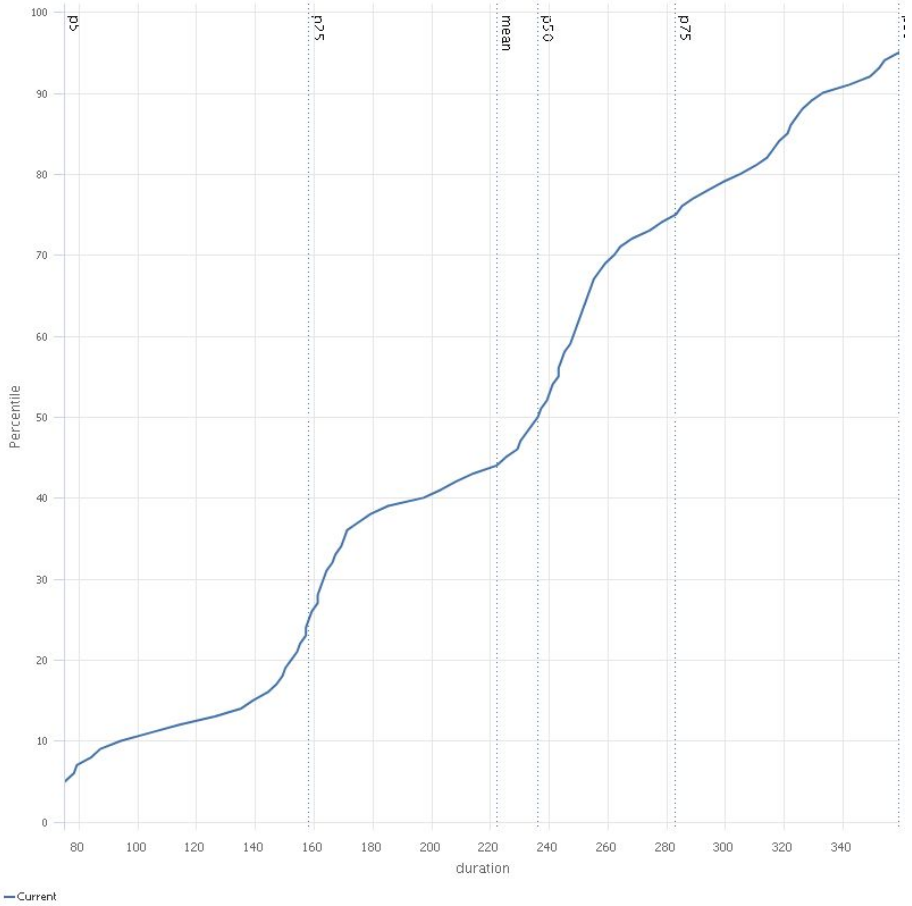


Successful test runs duration p90 (in sec)





Build x86_64/arm64, p90 ~400s

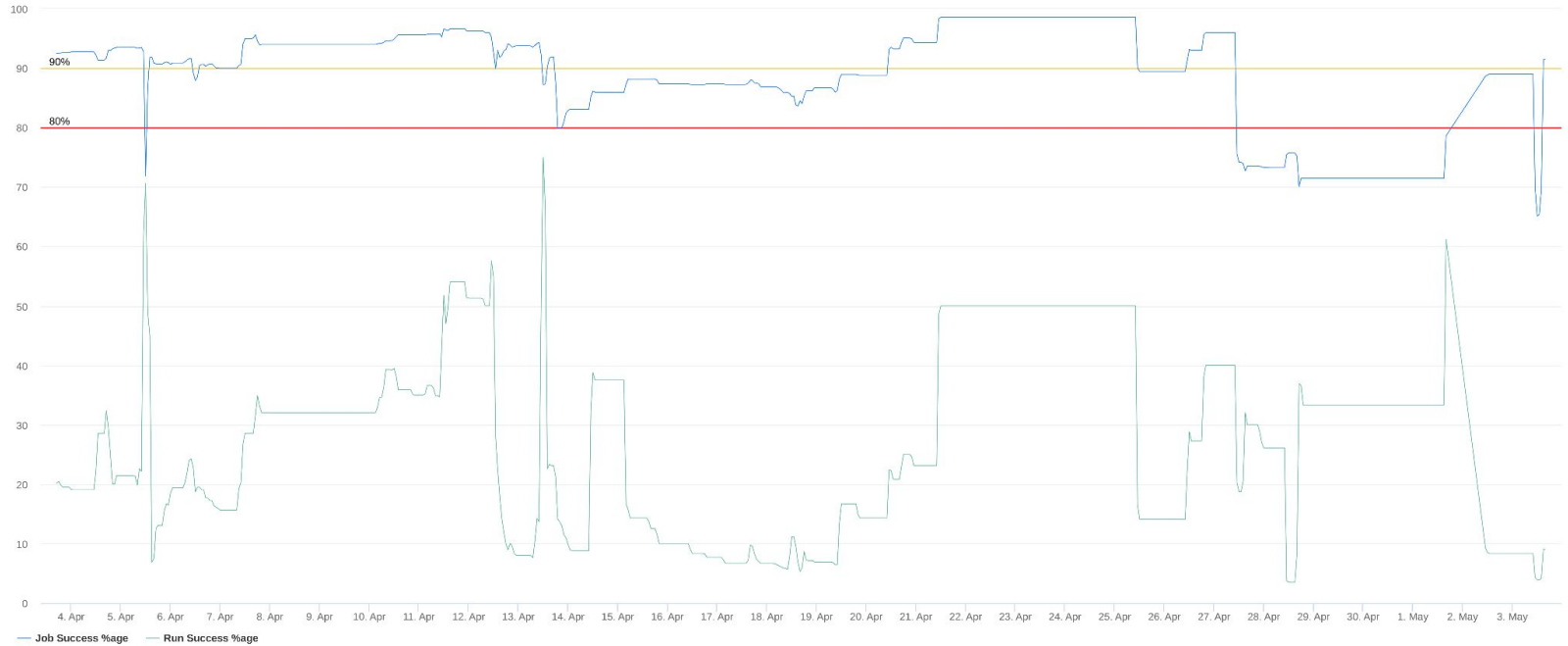


Tests x86_64/arm64, p90 ~330s

Challenges: Flaky tests

1 flaky test turns CI red

“High” job success rate, yet low run success rate



Challenges: Flaky tests

- Relying on exact count + network
  - netcnt: serial_test_netcnt:FAIL:packets unexpected packets: actual 10001 != expected 10000
  - migrate_reuseport: count_requests:FAIL:count in BPF prog unexpected count in BPF prog: actual 22 != expected 25
- Probably exacerbated when running in a VM
- More investigation needed to understand source of flakiness

Discussion

- How to make community more aware of/invested in CI health?
  - Could export telemetry to external datasink/grafana
  - Preferably no extra infra to maintain
- Get reliable base
  - to be more aggressive on enforcing a green CI
  - Possibly sending test results to series author on failure
- flaky tests
  - How to detect early, minimize their noise (FLAKYLIST turning them into warning? Multiple retries)
- Requiring coverage for new functionalities
- Increasing community participation
- Repro error environment is complicated, lot going on within GH actions